ملاحظات للعاملين بلغة CSS

تأليف
مساهمون من Stack Overflow
ترجمة
أيمن طارق القاضي

أكاديمية حسوب

ملاحظات للعاملين بلغة CSS

أتقن لغة CSS بأمثلة تطبيقية وملاحظات عملية متقدمة

اسم الكتاب: ملاحظات للعاملين بلغة CSS

المؤلف: مساهمون من Stackoverflow

المترجم: أيمن طارق القاضي

المحرر: جميل بيلوني

تصميم الغلاف: محمد زاهر شلار

سنة النشر: 2023

رقم الإصدار: 1.0

Book Title: CSS Notes for Professionals

Author: StackOverflow Contributors

Translator: Ayman Tarig Algadi

Editor: Jamil Bailony

Cover Design: Mohamed Zaher Shallar

Publication Year: 2023

Edition: 1.0

https://academy.hsoub.com

academy@hsoub.com

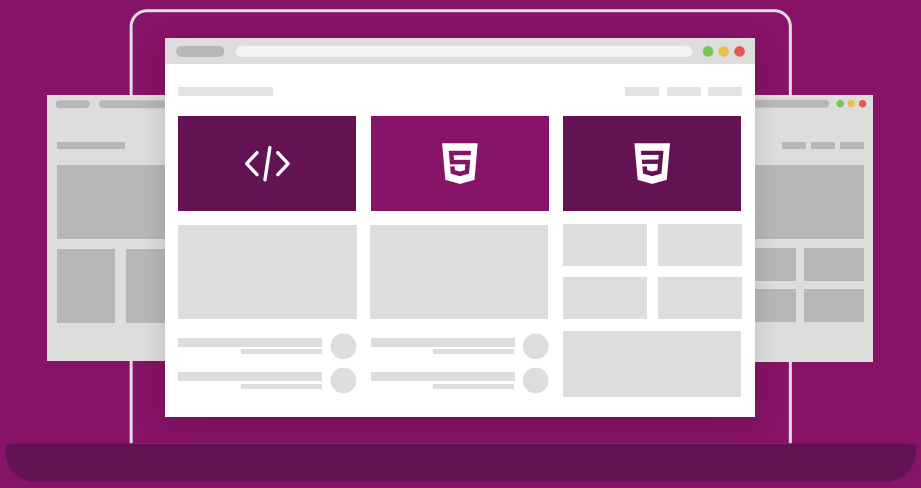
عن الناشـر

أُنتج هذا الكتاب برعاية شركة حسوب وأكاديمية حسوب.

تهدف أكاديمية حسوب إلى توفير دروس وكتب عالية الجودة في مختلف المجالات وتقدم دورات شاملة لتعلُّم البرمجة بأحدث تقنياتها معتمدةً على التطبيق العملي الذي يؤهل الطالب لدخول سوق العمل بثقة.

حسوب شركة تقنية في مهمة لتطوير العالم العربي. تبني حسوب منتجات تركِّز على تحسين مستقبل العمل، والتعليم، والتواصل. تدير حسوب أكبر منصتي عمل حر في العالم العربي، مستقل وخمسات ويعمل في فيها فريق شاب وشغوف من مختلف الدول العربية.

دورة تطوير واجهات المستخدم



مميزات الدورة

- بناء معرض أعمال قوي بمشاريع حقيقية
- وصول مدى الحياة لمحتويات الدورة
- تحديثات مستمرة على الدورة مجانًا
- شهادة معتمدة من أكاديمية حسوب
- إرشادات من المدربين على مدار الساعة
- من الصفر دون الحاجة لخبرة مسبقة

اشترك الآن

المحتويات باختصار

جدول المحتويات

تمهيد

لغة CSS هي لغة توصيف شهيرة ذائعة الصيت تُستخدم في تنسيق صفحات HTML لمواقع الويب وإضفاء رونق جميل عليها وهي الركيزة الثانية الأساسية من ركائز بناء صفحات ومواقع الويب بعد لغة HTML ولغة جافاسكربت JavaScript فلا تخفى على أي مطور ويب أو مطلع على مجال الويب عمومًا.

تعد لغة CSS بسيطة سهلة التعلم لكن في الوقت نفسه احترافها ليس بالسهل ويحتاج إلى خبرة وتعامل طويل معها، إذ تنسيق صندوق أو نص أو صفحة بسيطة بلغة CSS أمر شديد السهولة لكن عندما تطول الصفحة وتكثر عناصرها وتتشابك مع بعضها تبدأ الصعوبة والخروج عن النمط المألوف وحتى ظهور تعارضات وتداخلات بالتنسيق قد يأخذ حلها الدقائق وحتى الساعات.

أمرٌ إضافيٌ وهو أن لغة CSS تحتاج إلى بيئة لتنفيذها وعادة ما تكون البيئة هي متصفح الويب، ومتصفحات الويب تعمل بمحركات داخلية مسؤولة عن تفسير اللغات وتنفيذها ومنها لغة CSS وهنا تظهر مشكلة التوافقية مع تلك المحركات والحاجة إلى توحيد التنسيق على جميع المتصفحات وهذا أمر آخر يحتاج مطور الويب إلى تعلمه ومعالجته.

انطلاقًا من ذلك، جاء هذا الكتاب والذي يحمل عنوان "نصائح للعاملين بلغة CSS" وذلك لينقل مطور الويب من مرحلة المعرفة بأساسيات لغة CSS إلى الاحتراف ويختصر عليك الوقت الطويل ويغنيك عن التجريب والوقوع في الخطأ نفسه والتعلم منه، ويأتي الكتاب على شكل نصائح ومقتطفات من مطورين خبروا اللغة واستعملوها مرارًا وتكرارًا في مختلف المشاريع.

حول الكتاب

هذا الكتاب مترجم عن كتاب "CSS Notes For Professionals" المبني على توثيق موقع StackOverflow وقد ساهم في إعداده عدد كبير من المساهمين على شبكة StackOverflow الشهيرة لحل

المشاكل البرمجية. وإن أردت الاطلاع على قائمة المساهمين الكاملة، ارجع إلى قسم "Credits" في نهاية الكتاب الأصلي.

انتبه إلى أن هذا الكتاب ليس مثل غيره من الكتب والشروحات التي تشرح لغة CSS من البداية شرحًا مُبسَّطًا ومتسلسلًا وإنَّما يعتمد على مبدأ خير الكلام ما قل ودل في الشرح وترك الشيفرة تشرح نفسها بنفسها، فيحوي على كم كبير من الشيفرات بالموازنة مع الشرح. وُجِّه هذا الكتاب لمن لديه معرفة بلغة CSS، لذا يفضَّل أن تمتلك خبرة بلغة SQL لتستفيد أكبر استفادة من هذا الكتاب وتقرأ الشيفرات وتفهمها وتتعلَّم منها. في هذه الحالة، سيساهم هذا الكتاب في رفع مستواك في لغة CSS وسيُملِّكك مهارات متقدمة في استعمالها بالإضافة إلى بعض الخدع والالتفافات المتقدمة أيضًا.

قد تسأل نفسك، هل ينفع أن اقرأ الكتاب دون معرفة مسبقة بلغة CSS؟ سأقول، نعم، ولكن يجب أن تتحلى بالصبر في قراءة الشيفرة وتحليلها وفهمها والبحث عن أي موضوع لم تفهمه والسؤال عن شرح لأي شيفرة غامضة، إذ لن تجد كلامًا وشرحًا كبيرًا للمواضيع التي يتحدث عنها الكتاب، كما أن تسلسل المواضيع في الكتاب لا تراعي عدم امتلاك القارئ معرفة بلغة CSS.

بذكر ترتيب عناوين ومواضيع الكتاب، حاولت ترتيب عناوين الكتاب بأنسب شكل لتكون متدرِّجة في الصعوبة وحاولت جمع المواضيع المتشابهة في فصل واحد رغم تشرذمها وتفرقها في الكتاب الأصلي فلا تشبه النسخة العربية النسخة الأجنبية مطلقًا، إذ بذلت جهدًا لتكون أفضل منها. فإن كنت على معرفة بأحد المواضيع، فلا تتخطاها بل اقرأها، فقد تمر معك إشارة لموضوع متقدم أو ملاحظة مهمة لم تكن تعرفها (تذكر أنَّ اسم الكتاب ملاحظات متقدمة). يمكنك أيضًا أن تقرأ الكتاب من أي قسم تريد فهو من الأساس غير مُرتَّب ترتيبًا متدرجًا ومتسلسلًا كما أشرت رغم محاولتي في ترتيبه لك أنسب ترتيب من البداية للنهاية.

أنشئ العمل الأصلي من هذا الكتاب لأغراض تعليمية ولا يتبع إلى أي شركة أو مجموعة رسمية متعلقة بلغة CSS ولا حتى شبكة StackOverflow، كما أن جميع العلامات التجارية المذكورة في هذا الكتاب تتبع إلى الشركات المالكة لها.

ماذا بعد هذا الكتاب

عند الانتهاء من هذا الكتاب، يمكنك الاطلاع على العديد من المقالات العملية في أكاديمية حسوب. أثناء ذلك، يمكنك التنقل بين توثيق لغة CSS في موسوعة حسوب وفصول هذا الكتاب.

يمكن لأيِّ شخص ملم بالبرمجة أن يساهم في المشاريع مفتوحة المصدر. البرامج مفتوحة المصدر هي برامج متاحة للاستخدام وإعادة التوزيع والتعديل دون قيود. تساعد المساهمة في المشاريع مفتوحة المصدر على تحسين البرامج، عبر ضمان تمثيلها لقاعدة عريضة من المستخدمين. عندما يساهم المستخدمون في المشاريع مفتوحة المصدر، سواء عبر كتابة الشيفرة، أو التوثيق، أو صيانة المجلدات، فإنَّهم يوفرون قيمة مضافة للمشروع، ومجتمع المطورين على العموم.

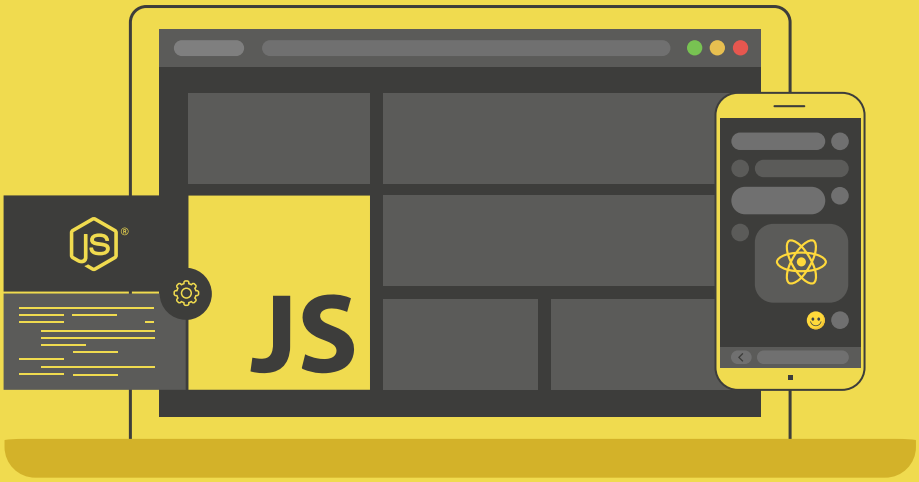
للحصول على مراجع إضافية عن CSS، أو للمشاركة في نقاشات مع الآخرين، يمكنك الاطلاع على المقـالات والأسئلة والدروس عن CSS في أكاديمية حسوب.

إذا كنت مهتمًا بتعلم تطوير تطبيقات الويب، أو تعلم لغات محـدَّدة مثـل PHP وجافاسـكربت، فـاطلع على قسم الدورات في الأكاديمية، كما يمكنك تصفح موسـوعة حسـوب لأجـل قـراءة توثيقـات عـدد كبـير من لغـات البرمجة باللغة العربية.

جميل بيلوني

16/01/2023

دورة تطوير التطبيقات
باستخدام لغة JavaScript



احترف تطوير التطبيقات بلغة جافا سكريبت
انطلاقًا من أبسط المفاهيم وحتى بناء تطبيقات حقيقية

التحق بالدورة الآن

1. أساسيات لغة CSS

يوضح الجدول التالي إصدارات CSS:

الإصدار	تاريخ الصدور
CSS1	17-12-1996
CSS2	12-5-1998
CSS3	13-10-2015

1.1 إضافة تنسيقات CSS للصفحات

1.1.1 التنسيقات الخارجية External Styles

تُضاف ملفات CSS الخارجية إلى صفحات HTML عن طريق الوسم `<link>`، حيث تحدد الخاصية `href` فيه إلى مسار الملف، وتأخذ الخاصية `rel` القيمة `stylesheet`، أما الخاصية `type` فيمكن تجاهلها أو إعطائها القيمة `text/css`. ومن الأفضل إضافة هذا الوسم داخل الوسم `<head>` لضمان تحميل ملف التنسيقات قبل تحميل العناصر مما يؤدي لظهورها مُنسَّقة فورًا، وفيما عدا ذلك ستظهر العناصر غير مُنسَّقة لفترة قصيرة حتى يكتمل تحميل ملف التنسيقات.

تتكون قواعد CSS من مُحدِّد selector، وكتلة التعريفات declaration block:

```
h1 {}
/* h1: المحدد
{}: كتلة التعريفات
*/
```

مثال:

- ملف HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <h1>Hello World!</h1>
        <p>I    CSS</p>
    </body>
</html>
```

- ملف style.css

```
h1 {
    color: green;
    text-decoration: underline;
}

p {
    font-size: 25px;
    font-family: 'Trebuchet MS', sans-serif;
}
```

انتبه، تأكد دائمًا أن الخاصية `href` تشير للمسار الصحيح لملف CSS، وإذا كان الملف داخل مجلد آخر ينبغي تحديد ذلك المجلد كما هو موضح بالشيفرة التالية:

```
<link rel="stylesheet" type="text/css" href="foldername/style.css" />
```

وتعتبر إضافة التنسيقات عن طريق ملف CSS خارجي هي الطريقة الأفضل ولذلك لأنها تجمع كل التنسيقات في ملف مركزي واحد مما يُسهِّل الرجوع إليه وتعديله في المستقبل.

ومن الممكن أيضًا إضافة عدد من ملفات CSS الخارجية، مثلًا:

```
<link rel="stylesheet" type="text/css" href="main.css" />
```

```
<link rel="stylesheet" type="text/css" href="override.css" />
```

وفي هذه الحالة ينبغي مراعاة ترتيب الملفات عند كتابة التنسيقات، فعلى سبيل المثال إذا كان الملف main.css يحوي القاعدة التالية والتي تجعل جميع الفقرات ذات الصنف green تظهر باللون الأخضر الفاتح.

```
p.green { color: #00FF00; }
```

وكان الملف override.css يحوي القاعدة الموضحة أدناه والتي تجعل لون الفقرات أخضر غامقًا

```
p.green { color: #006600; }
```

فإن النتيجة النهائية ستكون ظهور الفقرات باللون الأخضر الغامق بسبب أن الملف override.css يُحمَّل إلى المتصفح بعد الملف main.css مما يُجبر المتصفح على تطبيق القواعد المذكورة فيه وستطغى القواعد الموجودة فيه مع أي قواعد سابقة تتعارض معها وتلغيها.

> هناك بعض القواعد والمميزات مثل القاعدة !important والوراثة وأعماق التحديد والتي تمكنك من التحكم في كيفية تطبيق الخواص، وستُناقش هذه القواعد لاحقًا.

> تُحمَّل ملفات CSS الخارجية مرة واحدة فقط عند تحميل الصفحة للمرة الأولى، وتبقى محفوظة في ذاكرة المتصفح عند التنقل بين الصفحات مما يؤدي لتحميلها بشكل أسرع.

1.1.2 التنسيقات الداخلية Internal Styles

يمكن إضافة تنسيقات CSS داخل ملف HTML مباشرة، وذلك عن طريق كتابتها بين الوسمين:

```
<style> ... </style>
```

ويجب وضعها داخل الوسم <head>. إليك مثال:

- ملف HTML

```
<head>
    <style>
        h1 {
            color: green;
            text-decoration: underline;
        }
        p {
```

```
            font-size: 25px;
            font-family: 'Trebuchet MS', sans-serif;
        }
    </style>
</head>

<body>
    <h1>Hello World</h1>
    <p>I    CSS</p>
</body>
```

1.1.3 القاعدة @import

تُستعمل القاعدة `@import` لاستيراد قواعد CSS من ملفات أخرى، ويجب أن تسبق هذه القاعدة جميع القواعد الأخرى عدا القاعدة `@charset`، ولا يمكن استعمالها داخل قواعد `@` مثل قواعد استعلامات الوسائط وغيرها.

يمكن استخدام القاعدة `@import` في التنسيقات الداخلية بالشكل التالي:

```
<style>
    @import url('/css/style.css');
</style>
```

حيث تحدد الخاصية `url` مسار الملف المُراد استيراده.

يمكن أيضًا استعمالها في ملفات التنسيقات الخارجية، حيث تستورد الشيفرة أدناه الملف additional-styles.css إلى ملف CSS الرئيسي:

```
@import '/additional-styles.css';
```

ومن الشائع استعمال القاعدة `@import` لاستيراد الخطوط من الويب كما هو موضح في الشيفرة التالية:

```
@import 'https://fonts.googleapis.com/css?family=Lato';
```

ويمكن إضافة استعلامات الوسائط كمعاملات اختيارية للقاعدة `@import`، وفي هذه الحالية تُستورد الملفات المحددة فقط في حالة استيفاء شروط استعلامات الوسائط.

```
@import '/print-styles.css' print;
@import url('landscape.css') screen and (orientation: landscape);
```

1.1.4 التنسيقات السطرية Inline Styles

تُستعمل التنسيقات السطرية لتطبيق الأنماط على عنصر محدد داخله مباشرةً، وتستبدل القواعـد المحـددة في الملفات الخارجية أو في الوسم `<style>` في حال حدوث أي تعارض.

> من الأفضل استخدام الملفات الخارجية أو الوسم `<style>` لتطبيق التنسيقات، وذلك للفصل بين البنية الهيكلية للصفحة (ملف HTML) والتنسيقات مما يسهل الرجوع إليها وتعديلها.

مثال:

```
<h1 style="color: green; text-decoration: underline;">
    Hello World!
</h1>
<p style="font-size: 25px; font-family: 'Trebuchet MS';">
    I    CSS
</p>
```

ومن مميزات التنسيقات السطرية أنَّها تضمن تطبيق التنسيقات على العناصر دون أن تتخطاهـا أي قاعـدة أو خاصـية، ولكنَّهـا تسـتغرق وقتًـا طـويلًا في الكتابـة وتصـعب مراجعتهـا وتعـديلها خصوصًـا في الصـفحات والمواقع الكبيرة.

1.1.5 التحكم في تنسيقات CSS باستخدام جافاسكربت

يمكن إضافة أو حذف أو تعديل قواعد CSS باستخدام جافاسكربت عن طريق الخاصية `style`.

مثال:

```
var el = document.getElementById("element");
el.style.opacity: 0.5;
el.style.fontFamily: 'sans-serif';
```

ينبغي ملاحظة استخدام نمط سنام الجمل المنخفض lower camel case لكتابة أسـماء الخاصـيات في جافاسكربت، فمثلًا الخاصية `font-family` في CSS تصبح `fontFamily` في جافاسكربت.

ومن الممكن أيضًا إنشاء عنصر `<style>` وإضافة قواعد CSS بداخلـة أو إنشـاء عنصـر `<link>` يشـير إلى ملف CSS ومن ثم إسناده إلى الصفحة باستخدام جافاسكربت،

1.1.6 استخدام jQuery

إليك المثال التالي للتحكم في خاصية واحدة:

```
$('#element').css('margin', '5px');
```

انظر أيضًا المثال للتحكم في عدد من الخاصيات:

```
$('#element').css({
    margin: "5px",
    padding: "10px",
    color: "black"
});
```

بالنسبة للخواص التي تتكون من أكثر من كلمة مثل الخاصية `font-size`، فيمكن كتابتها بين علامتي تنصيص أو استعمال نمط سنام الجمل لكتابتها.

مثال:

```
$('.example-class').css({
   "background-color": "blue",
    fontSize: "10px"
});
```

للمزيد من المعلومات انظر توثيق jQuery.

1.2 هيكلة وصياغة قواعد CSS

تأخذ بعض خاصيات CSS أكثر من قيمة واحدة، وتسمى مجموعة هذه القيم قائمة الخاصية.

مثال:

```
span {
    text-shadow: yellow 0 0 3px, green 4px 4px 10px;
}

span {
    text-shadow:
        yellow 0 0 3px,
        green 4px 4px 10px;
}
```

تُمكنك CSS أيضًا من تطبيق التنسيقات على عدد من العناصر دون إعادة كتابتها لكل عنصر على حدة، وذلك عن طريق استخدام المحدِّدات المتعددة كما هو موضح في الأمثلة التالية:

```
div, p { color: blue; }
p, .blue, #first, div span {
    color: blue;
}
```

ضع في بالك ما يلي:

- ينبغي الفصل بين المحددات بفاصلة.
- في حال عدم وضع الفاصلة كما في `div span`، فتُطبق التنسيقات على عناصر `span` الموجودة داخل عناصر `div` فقط.

1.3 التعليقات Comments

تُكتب التعليقـات بين العلامـتين `/*` و `*/` ، ويمكن أن تكـون من سـطر واحـد (المثـال الأول) أو عـدد من الأسطر (كما في المثال الثاني).

```
div {
    color: red; /* هذا تعليق من سطر واحد */
}
/*
تعليق
من
أربعة
أسطر
*/

div {
    color: red;
}
```

1.4 المحددات Selectors

تُستعمل المحددات لتحديد أو استهداف عناصر HTML المُراد تطبيق قواعد CSS عليها. وتحتـوي CSS على أكثر من 50 نوع من أنواع المحددات منها محددات العناصر elements، والأصناف classes، والمُعرِّفـات IDs، والعناصر الزائفة pseudo-elements، والأصناف الزائفة pseudo-classes.

يوضح الجدول التالي المحددات الأساسية في CSS:

المحدد	الوصف
`*`	المُحدِّد العام، ويُحدد جميع العناصر.
`div`	محدد الوسوم، ويُحدد جميع العناصر التي لها وسم مُعين.
`.blue`	مُحدد الصنف، ويُحدد جميع العناصر التي لها صِنف class مُعين.
`.blue.red`	يُحدد جميع العناصر التي لها الصنفين blue و red معًا.
`#headline`	مُحدد المُعرِّف، ويُحدد جميع العناصر التي لها مُعرِّف Id مُعين.
`::pseudo-class`	محددات الأصناف الزائفة.
`::pseudo-element`	محددات العناصر الزائفة.
`:lang(en)`	يُحدد العناصر ذات القيمة en للخاصية `:lang` مثلًا العنصر.
`div > p`	محدد الأبناء.

> معيار HTML5 يقول أنَّ قيمة الخاصية `id` يجب أن تكون فريدةً لكل عنصر في المستند، ولا يجوز تكرارها أبدًا.

انظر قائمة محددات CSS في توثيق CSS العربي.

1.4.1 محددات الخاصيات Attribute Selectors

مُحدِّد الخاصيات attribute selector في CSS يُطابِق العناصر إذ وجدت فيها خاصية ما أو كانت تلك الخاصية تملك قيمة معيّنة.

المُحدد	العُنصر المطابق	الوصف	إصدار CSS
`[attr]`	`<div attr>`	يُطابق العناصر ذات الخاصية `attr`.	2
`[attr='val']`	`<div attr='val'>`	يطابق العناصر التي لها القيمة `val` للخاصية `attr`.	2
`[attr~='val']`	`<div attr='val val2 val3'>`	يطابق العناصر التي تظهر فيها القيمة `val` كإحدى قيم الخاصية `attr`.	2
`[attr^='val']`	`<div attr='val1 val2'>`	يطابق العناصر التي تبدأ قيمة الخاصية `attr` فيها بالكلمة `val`.	3
`[attr$='val']`	`<div attr='sth aval'>`	يطابق العناصر التي تنتهي قيمة الخاصية `attr` فيها بالكلمة `val`.	3
`[attr*='val']`	`<div attr='somevalhere'>`	يطابق العناصر التي تحتوي قيمة الخاصية `attr` فيها على الكلمة `val`.	3

المُحدد	العُنصر المطابق	الوصف	إصدار CSS
`[attr|='val']`	`<div attr='val-sth etc'>`	يطابق العناصر التي لها القيمة `val` للخاصية `attr`، أو تبدأ بالكلمة `val` متبوعة بالمحرف (-).	2
`[attr='val' i]`	`<div attr='val'>`	يطابق العناصر التي لها القيمة `val` للخاصية `attr`، مع تجاهل حالة الأحرف.	4(2)

انتبه إلى ما يلي:

1. يجب وضع قيم الخواص بين علامات تنصيص مزدوجة ("") أو مفردة ('').
2. لا توجد إصدارة متكاملة من CSS4، لأنها مقسَّمة إلى عدد من الوحدات. لمزيد من المعلومات انظر دعم المتصفحات.

ا. المحدد [attribute]

يُحدد العناصر التي تمتلك خاصية معينة.

مثال:

```
<style>
    div[data-color] {
        color: red;
    }
</style>

<div data-color="red">This will be red</div>
<div data-color="green">This will be red</div>
<div data-background="red">This will NOT be red</div>
```

اطّلع على تجربة حيَّة على JSBin.

ب. المحدد [attribute="value"]

يُحدد العناصر التي لها قيمة مُعينة للخاصية.

مثال:

```
<style>
    div[data-color="red"] {
        color: red;
    }
</style>

<div data-color="red">This will be red</div>
<div data-color="green">This will be red</div>
<div data-background="red">This will NOT be red</div>
```

اطَّلع على تجربة حيَّة على JSBin.

ج. المحدد [attribute*="value"]

يُحدد العناصر التي لها تحتوي قيمة الخاصية فيها على كلمة معينة.

مثال:

```
<style>
    [class*="foo"] {
        color: red;
    }
</style>

<div class="foo-123">This will be red</div>
<div class="foo123">This will be red</div>
<div class="bar123foo">This will be red</div>
<div class="barfooo123">This will be red</div>
<div class="barfo0">This will NOT be red</div>
```

اطَّلع على تجربة حيَّة على JSBin.

د. المحدد [attribute~="value"]

يُحدد العناصر التي تظهر فيها القيمة `value` كإحدى قيم الخاصية.

مثال:

```
<style>
    [class~="color-red"] {
```

```
        color: red;
    }
</style>

<div class="color-red foo-bar the-div">This will be red</div>
<div class="color-blue foo-bar the-div">This will NOT be red</div>
```

اطَّلع على تجربة حيَّة على JSBin.

ه. المحدد [attribute^="value"]

يُحدد العناصر التي تبدأ قيمة الخاصية فيها بكلمة معينة.

مثال:

```
<style>
    [class^="foo-"] {
        color: red;
    }
</style>

<div class="foo-123">This will be red</div>
<div class="foo-234">This will be red</div>
<div class="bar-123">This will NOT be red</div
```

اطّلع على تجربة حيَّة على JSBin.

و. المحدد [attribute$="value"]

يُحدد العناصر التي تنتهي قيمة الخاصية فيها بكلمة معينة.

مثال:

```
<style>
    class$="file"] {
        color: red;
    }
</style>

<div class="foobar-file">This will be red</div>
```

```
<div class="foobar-file">This will be red</div>
<div class="foobar-input">This will NOT be red</div>
```

اطّلع على تجربة حيَّة على JSBin.

ز. المحدد [attribute|="value"]

يطابق العناصر التي لها القيمة `value` للخاصية، أو تبدأ بالكلمة `value` متبوعة بالمحرف (-).

مثال:

```
<style>
    [lang|="EN"] {
        color: red;
    }
</style>

<div lang="EN-us">This will be red</div>
<div lang="EN-gb">This will be red</div>
<div lang="PT-pt">This will NOT be red</div>
```

اطّلع على تجربة حيَّة على JSBin.

ح. المحدد [attribute="value" i]

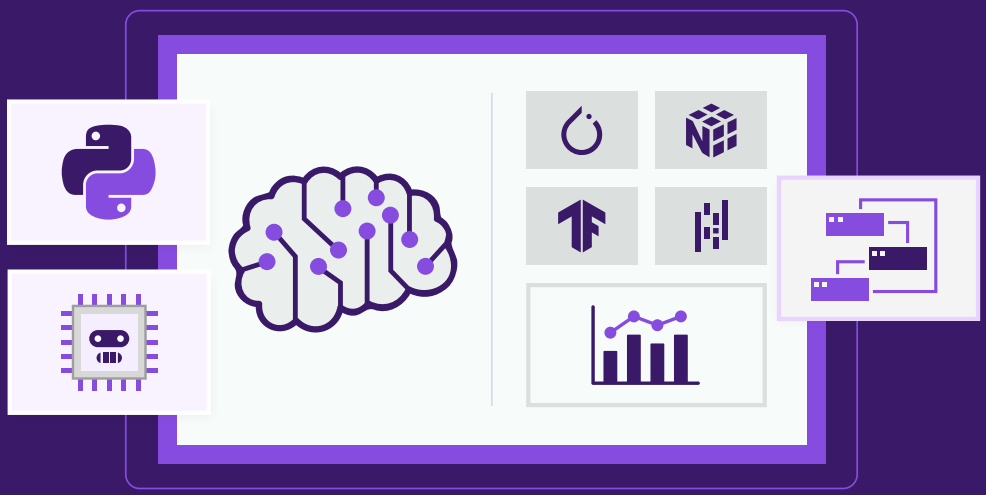
يُحدد العناصر التي لها قيمة مُعينة للخاصية، مع تجاهل حالة الأحرف.

مثال:

```
<style>
    [lang="EN" i] {
        color: red;
    }
</style>
<div lang="EN">This will be red</div>
<div lang="en">This will be red</div>
<div lang="PT">This will NOT be red</div>
```

اطّلع على تجربة حيَّة على JSBin.

دورة الذكاء الاصطناعي



تعلم الذكاء الاصطناعي وتعلم الآلة والتعلم العميق
وتحليل البيانات، وأضفها إلى تطبيقاتك

التحق بالدورة الآن

1.4.2 المُجمّعات Combinators

المحدد	الوصف
`div span`	مُحدِّد العناصر السليلة descendant combinator في CSS الذي يُمثَّل عادةً بفراغٍ واحد ويفصل بين مُحدِّدين، ويطُابِق العنصر الثاني إذا كان ابنًا للعنصر الأول.
`div > span`	مُحدِّد العناصر الأبناء child combinator في CSS رمزه > ويفصل بين مُحدِّدين، ويطُابِق العنصر الثاني إذا كان ابنًا للعنصر الأول.
`a ~ span`	مُحدِّد العناصر الأخوة العامة general sibling combinator في CSS رمزه ~ ويفصل بين مُحدِّدين، ويطابق العنصر الثاني إذا أتى بعد العنصر الأول (وليس بالضرورة أن يأتي بعده مباشرةً)، وكان كلا العنصرين ابناءً لعنصر أب مشترك.
`a + span`	مُحدِّد العناصر الأخوة المتجاورة adjacent sibling combinator في CSS رمزه + ويفصل بين مُحدِّدين، ويطُابِق العنصر الثاني إذا أتى مباشرةً بعد العنصر الأول وكان كلا العنصرين أبناءً لعنصر أب مشترك.

> مُحددات العناصر المتجاورة تُحدد العناصر التالية لعنصر مُعين، ولكن يُمكن استعمال الخاصية flex orde لتحديد العناصر السابقة.

ا. مُحدد العناصر السليلة Descendant Combinator

```
<style>
    div p {
            color:red;
    }
</style>

<div>
    <p>My text is red</p>
    <section>
        <p>My text is red</p>
    </section>
</div>

<p>My text is not red</p>
```

اطَّلع على تجربة حيَّة على JSBin.

ب. مُحدد العناصر الأبناء Child Combinator

```
<style>
    div > p {
        color:red;
    }
</style>

<div>
    <p>My text is red</p>
    <section>
           <p>My text is not red</p>
    </section>
</div>
```

اطّلع على تجربة حيَّة على JSBin.

ج. مُحدد العناصر الإخوة المتجاورة Adjacent Sibling Combinator

```
<style>
    p + p {
        color:red;
    }
</style>
<p>My text is not red</p>
<p>My text is red</p>
<p>My text is red</p>
<hr>
<p>My text is not red</p>
```

اطّلع على تجربة حيَّة على JSBin.

د. مُحدد العناصر الإخوة العامة General Sibling Combinator

```
<style>
    p ~ p {
        color:red;
    }
</style>
```

```
<p>My text is not red</p>
<p>My text is red</p>
<hr>
<h1>And now a title</h1>
<p>My text is red</p>
```

اطّلع على تجربة حيَّة على JSBin.

1.4.3 الأصناف الزائفة pseudo-classes

الأصناف الزائفة هي كلمات مفتاحية تسمح بتحديد العناصر بناءً على معلومات غير متوفرة في البنية الشجرية للصفحة، مثال لذلك تحديد العناصر بناءً على حالة أو ديناميكية العنصر، أو بناءً على الموقع (الأصناف الزائفة للبنية الهيكلية والاستهداف)، أو استعمال الأصناف الزائفة لنفي تحديد عنصر معين، أو لتحديد العناصر بناءً على اللغة المستخدمة فيها.

الصيغة العامة:

```
selector:pseudo-class {
    property: VALUE;
}
```

إليك الجدول التالي الذي يشرح قائمة الأصناف الزائفة.

الاسم	الوصف
`:active`	يُحدد العناصر التي جرى تفعيلها من المستخدم active elements، ويُفَعَّل العنصر عندما يضغط المستخدم عليه بزر الفأرة الرئيسي.
`:any`	يُمكن من إنشاء مجموعة من المحددات بحيث تُحدِّد العناصر التي تمتلك هذه المجموعة.
`:target`	يُحدد عنصرًا فريدًا يُطابِق مُعرِّفه `id` قسمًا من رابط الصفحة.
`:checked`	يُحدد أزرار الانتقاء radio button أو صناديق التأشير checkbox أو الخيارات `<option>` في عنصر `<select>` التي قام المستخدم بتحديدها أو انتقائها.
`:default`	يُمثِّل أي عنصر مُختار افتراضيًا بين مجموعة من العناصر المتعلقة به.
`:disabled`	يُمثِّل أي عنصر مُعطَّل، ولا يمكن اختياره أو النقر عليه أو الكتابة فيه.
`:empty`	يُمثِّل أي عنصر ليس له أبناء.
`:enabled`	يُمثِّل أي عنصر مُفعّل، أي يمكن اختياره أو النقر عليه أو الكتابة فيه.
`:first`	يُستعمَل مع القاعدة `@page`، ويُمثِّل أوّل صفحة من المستند عند طباعته.
`:first-child`	يُمثِّل أوّل عنصر في مجموعة من العناصر الأخوة.

الاسم	الوصف
`:first-of-type`	يُمثِّل أوّل عنصر من نوعه في مجموعة من العناصر الأخوة.
`:focus`	يُمثِّل عنصرًا (مثل عناصر النماذج) استقبل التركيز focus، ويُفعَّل التركيز إذا ضغط المستخدم أو لمس أحد العناصر أو وصل إليه عبر مفتاح tab على لوحة المفاتيح.
`:focus-within`	يُمثِّل عنصرًا (مثل عناصر النماذج) استقبل التركيز foucs أو احتوى على عنصر قد استقبل التركيز،
`:full-screen`	مثِّل العناصر التي تُعرَض عندما يكون المتصفح في وضع ملء الشاشة.
`:hover`	يُطابِق العناصر التي يتفاعل المستخدم معها عبر الفأرة لكن ليس بالضرورة أن يفعِّلها، ويُفعَّل هذا الصنف الزائف عادةً عندما يمرر المستخدم مؤشر الفأرة فوق العنصر.
`:indeterminate`	يُمثِّل أيّة نماذج تكون حالتها غير معروفة.
`:in-range`	يُمثِّل أيّة عناصر `<input>` التي تكون قيمتها ضمن المجال المسموح المُحدَّد عبر الخاصيتين `min` و `max`.
`:invalid`	يُمثِّل أيّة عناصر `<input>` أو `<form>` التي فشل التحقق من صحة محتوياتها.
`:lang`	يطابق العناصر اعتمادًا على لغتها.
`:last-child`	يُمثِّل آخر عنصر في مجموعة من العناصر الأخوة.
`:last-of-type`	يُمثِّل آخر عنصر من نوعه في مجموعة من العناصر الأخوة.
`:left`	يُستعمَل مع القاعدة `@page`، ويُمثِّل الصفحات اليسارية left hand pages من المستند عند طباعته.
`:link`	يُمثِّل عنصرًا لم تتم زيارته من قبل، ويُطابِق جميع عناصر والتي لها الخاصية `href` ولم تتم زيارتها بعد.
`:not()`	مثِّل العناصر التي لا تُطابِق مُحدِّدًا أو أكثر، يسمى هذا الصنف أيضًا بصنف النفي الزائف.
`:nth-child`	طابِق عنصرًا أو أكثر بناءً على موقعه ضمن مجموعة من العناصر الأخوة.
`:nth-of-type`	يُطابِق عنصرًا أو أكثر بناءً على موقعه ضمن مجموعة من العناصر الأخوة siblings، وذلك بدءًا من نهاية المجموعة.
`:only-child`	يُطابِق عنصرًا لا يملك أيّة عناصر أخوة، وهذا المُحدِّد يماثل `:first-child:last-child` أو `:nth-child(1):nth-last-child(1)` لكن درجة التحديد له أقل.
`:optional`	يُمثِّل أيّة عناصر `<input>` أو `<select>` أو `<textarea>` التي لم تُضبَط الخاصية `required` عليها.
`:out-of-range`	يُمثِّل أيّة عناصر `<input>` التي لا تكون قيمتها ضمن المجال المسموح المُحدَّد عبر الخاصيتين `min` و `max`.

الاسم	الوصف
`:placeholder-shown`	يُمثِّل أي عنصر `<input>` أو `<textarea>` الذي يُظهِر نصًا بديلًا placeholder text.
`:read-only`	يُمثِّل أي عنصر فيه محتوى نصي لا يمكن للمستخدم تعديله.
`:read-write`	يُمثِّل أي عنصر فيه محتوى نصي يمكن للمستخدم تعديله.
`:right`	يُستعمَل مع القاعدة `@page`، ويُمثِّل الصفحات اليمينية right hand pages من المستند عند طباعته.
`:root`	يُطابِق العنصر الجذر في شجرة المستند، أي أنَّه يُطابِق العنصر `<html>` في مستندات HTML، وهو مكافئ للمُحدِّد `html` إلا أنَّ درجة التحديد له أعلى.
`:scope`	يُحدد العناصر التي تُمثل نقطة مرجعية للمحددات.
`:visited`	يُمثِّل عنصرًا تمت زيارته من قبل، ولأسباب تابعة للخصوصية فإنَّ عدد الخاصيات التي يمكن تغيير قيمتها باستخدام هذا المُحدِّد قليلةٌ جدًا.

إن أردت الاستزادة أو أردت شرحًا موسعًا لأي صنف زائف بالأعلى، فارجع إلى قسم الأصناف الزائفة من توثيق CSS العربي في موسوعة حسوب.

ا. صنف الابن الزائف Child Pseudo Class

يُحدد المُحدد الزائف `:nth-child(an+1b)` العناصر التي تسبقها 1 - `an+b` عنصر أخ في البنية الشجرية للصفحة، ويمكن أن تأخذ `n` أي عدد موجب أو الصفر.

الجدول أدناه يوضح العناصر التي يُحددها المُحدد الزائف في بنية هيكلة تتكون من عنصر أب و عشرة عناصر أبناء:

المُحدد الزائف	1	2	3	4	5	6	7	8	9	10
`:first-child`	×									
`:nth-child(3)`			×							
`:nth-child(n+3)`			×	×	×	×	×	×	×	×
`:nth-child(3n)`			×			×			×	
`:nth-child(3n+1)`	×			×			×			×
`:nth-child(-n+3)`	×	×	×							
`:nth-child(odd)`	×		×		×		×		×	
`:nth-child(even)`		×		×		×		×		×
`:last-child`										×
`:nth-last-child(3)`								×		

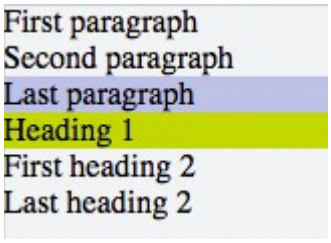
ب. المحدد last-of-type:

يُستخدم المُحدد `:last-of-type` لتحديد آخر عنصر من نوعه في مجموعة من العناصر الأخوة. في المثال أدناه تُحدد آخر فقرة وآخر عنوان رئيسي `h1` في الحاوية.

مثال:

```
<style>
    p:last-of-type {
        background: #C5CAE9;
    }
    h1:last-of-type {
        background: #CDDC39;
    }
</style>

<div class="container">
    <p>First paragraph</p>
    <p>Second paragraph</p>
    <p>Last paragraph</p>
    <h1>Heading 1</h1>
    <h2>First heading 2</h2>
    <h2>Last heading 2</h2>
</div>
```

النتيجة:



اطّلع على تجربة حيّة على JsFiddle.

ج. المحدد in-range:

المُحدد `:in-range` يُحدد عناصر `<input>` التي تكون قيمتها ضمن المدى المسموح المُحدَّد عبر الخاصيتين `min` و `max` مما يُتيح عرض تنبيهات تُفيد بأنَّ القيمة المُدخلة ضمن أو خارج المدى المسموح به.

مثال:

```
<style>
    input:in-range {
        border: 1px solid blue;
    }
</style>

<input type="number" min="10" max="20" value="15">
<p>The border for this value will be blue</p>
```

د. المحدد :not

مثال:

```
<style>
    input:not([disabled]):not(.example){
        background-color: #ccc;
    }
</style>

<form>
    Phone: <input type="tel" class="example">
    E-mail: <input type="email" disabled="disabled">
    Password: <input type="password">
</form>
```

المثال أعلاه يُحدد جميع عناصر `<input>` التي يُمكن للمستخدم التفاعل معهـا (غـير مُعطلـة)، ولا تمتلـك الصنف `.example`.

لاحظ، يقبل المُحدد `:not()` قائمة من المُحددات مفصولة بفاصلة كما هو موضح بالشيفرة التالية:

```
input:not([disabled], .example) {
    background-color: #ccc;
}
```

اطّلع على تجربة حيَّة على JSBin.

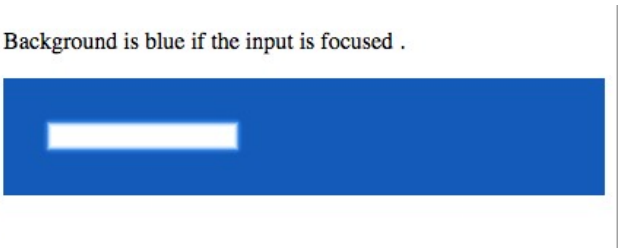
ه. المحدد focus-within:

مثال:

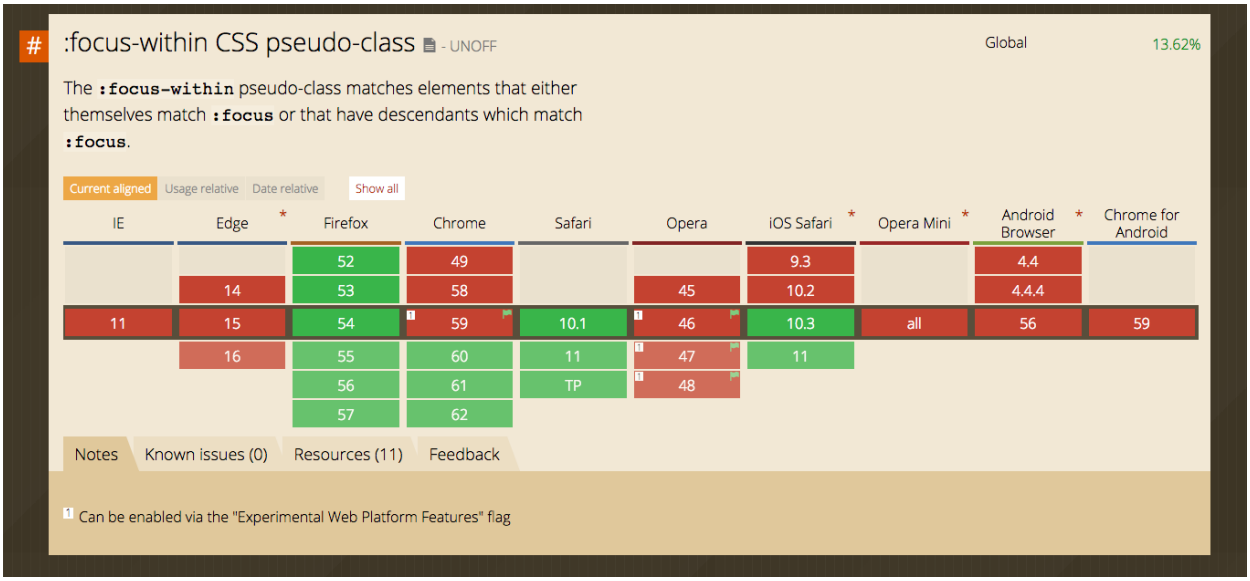
```
<style>
    div {
        height: 80px;
    }
    input {
        margin: 30px;
    }
    div:focus-within {
        background-color: #1565C0;
    }
</style>

<h3>Background os blue if the input is focused</h3>
<div>
    <input type="text" />
</div>
```

النتيجة:

```
div {
  height: 80px;
}
input{
  margin:30px;
}
div:focus-within {
  background-color: #1565C0;
}
```



توضح الصورة التالية دعم المتصفحات لهذا المحدد:



و. إنشاء قيم منطقية باستخدام المحددات

يمكن إنشاء قيم منطقية (أي قيم الصواب والخطأ) عن طريق استخدام المحددات وصناديق الإختيار، والخطوات التالية توضح كيفية ذلك.

الخطوة الأولى: أنشئ أي عدد من صناديق الاختيار checkboxes بمُعرِّفات خاصة، واجعلها مخفية باستخدام الكلمة المحجوزة `hidden`.

```
<input type="checkbox" id="sidebarShown" hidden />
<input type="checkbox" id="darkThemeUsed" hidden />

<!-- here begins actual content -->
<div id="container">
    <div id="sidebar">
        <!-- Menu, Search, ... -->
    </div>
    <!-- Some more content ... -->
</div>

<div id="footer">
    <!-- ... -->
</div>
```

الخطوة الثانية: التحكم في القيمة المنطقية، يُمكنك التحكم في القيمة المنطقية عن طريق إنشاء عنصر `label` وربطه مع صناديق التأشير عن طريق الخاصية `for`.

```
<label for="sodebarShown">Show/Hide the sidebar!</label>
```

الخطوة الثالثة: الوصول إلى القيمة المنطقيـة باسـتخدام CSS، يُمكن الحصـول على مُحـددات تُعطي القيم المنطقية `true` أو `false` باستخدام الشيفرة التالية:

```
/* true */
<checkbox>:checked ~ [sibling of checkbox & parent of target <target>

/* false */
<checkbox>:not(:checked) ~ [sibling of checkbox & parent of target]
<target>
```

لاحـظ يجب اسـتبدال `<checkbox>`، و `[… sibling]` و `<target>` بالمُحـددات المناسـبة، كمـا هـو موضح بالمثال أدناه:

```
#sidebarShown:checked ~ #container #sidebar {
    margin-left: 300px;
}

#darkThemeUsed:checked ~ #container,
#darkThemeUsed:checked ~ #footer {
    background: #333;
}
```

اطَّلع على تجربة حيَّه على JSFiddle.

j. المحدد only-child:

يُستخدم لتحديد جميع العناصر التي لا تمتلك إخوة.

مثال:

```
<style>
    p:only-child {
        color: blue;
    }
</style>

<div>
```

```
    <p>This paragraph is the only child of the div, it will have the
color blue</p>
</div>


<div>
    <p>This paragraph is one of the two children of the div</p>
    <p>This paragraph is one of the two children of its parent</p>
</div>
```

اطَّلع على تجربة حيَّة على JSBin.

ح. تنسيق عناصر input من النوع range

```
<input type="range"></input>
```

التأثير	المحددات
Thumb	input[type=range]::-webkit-slider-thumb input[type=range]::-moz-range-thumb input[type=range]::-ms-thumb
Track	input[type=range]::-webkit-slider-runnable-track input[type=range]::-moz-range-track input[type=range]::-ms-track
OnFocus	input[type=range]:focus
Lower part of the track	input[type=range]::-moz-range-progress input[type=range]::-ms-fill-lower (غير ممكن حاليًا في المتصفحات المعتمدة على WebKit لذا جافاسكربت مطلوبة)

ط. تحديد العناصر باستخدام المُعرِّف الخاص بها ID

يمكن تحديد العناصر باستخدام المُعرِّف الخاص بها ID مع تفادي عُمق التحديد العالي للمُعرِّف، إليك مثال:

- ملف HTML

```
    <div id="element" >
        ....
    </div>
```

- ملف CSS

```
    #element { ... } /* محددات المعرفات لها عمق تحديد عالي */

    [id="element"] { ... }
/* يعطي نفس نتيجة محدد المعرف ولكن له عمق تحديد أقل بسبب استعمال محددات الخواص */
```

1.4.4 محددات الأصناف Class Name Selectors

تُحدد محددات الأصناف العناصر التي تمتلك صنف معين، على سبيل المثال سيُحدد المحـدد `.warning` الحاوية أدناه

```
<div class="warning">
    <p>THis would be some warning copy.</p>
</div>
```

يُمكن دمج عدد من الأصناف للحصول على عمق تحديد عالي كما هو موضح في المثال أدناه

- ملف CSS

```
    .important {
        color: orange;
    }

    .warning {
        color: blue;
    }

    .warning.important {
        color: red;
    }
```

- ملف HTML

```
    <div class="warning">
        <p>This would be some warning copy.</p>
    </div>

    <div class="important warning">
```

```
      <p class="important">This is some really important warning
copy.</p>
  </div>
```

لاحظ عدم وجود مسافة space في المُحدد `.warning.important`، وهـذا يعـني تحديـد العنصـر الـذي يمتلك الصنفين معًا، أما في حـال وجـود مسـافة بينهمـا فـذلك يعـني تحديـد العنصـر الابن الـذي لـه الصـنف `.important` داخل عنصر أب له الصنف `.warning`.

1.4.5 محددات المُعرِّفات ID Selectors

تُستخدم محددات المُعرفات لتحديد العناصـر الـتي لهـا مُعـرِّف ID مُعين، ويجب أن يكـون المُعـرِّف فريـدًا بمعني أنه يظهر مرة واحدة في الصفحة.

مثال:

```
<style>
    #exampleId {
        width: 20px;
    }
</style>

<div id="exampleId">
    <p>Example</p>
</div>
```

1.4.6 محددات العناصر الزائفة

يوضح الجدول التالي قائمة بمحددات العناصر الزائفة:

العنصر	الوصف
`::after`	إضافة محتوى بعد محتوى العنصر الأصلي.
`::before`	إضافة محتوى قبل محتوى العنصر الأصلي.
`::first-letter`	تحدد الحرف الأول من كل عنصر.
`::first-line`	تحدد السطر الأول من كل عنصر.
`::selection`	تُحدد الجزء الذي حدده المستخدم من العنصر عن طريق مؤشر الفأرة.
`::backdrop`	تُستخدم لإنشاء خلفية تُخفي الوثيقة الأساسية للعنصر.
`::placeholder`	تُمكن من تنسيق النص المالئ (placeholder text) لعناصر الإدخال.

العنصر	الوصف
`::marker`	تُستخدم لإضافة تنسيق القائمة (list-style) للعنصر.
`::spelling-error`	تُستخدم للإشارة لنص به خطأ إملائي.
`::grammer-error`	تستخدم للإشارة لنص به خطأ نحوي.

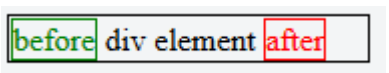
إن أردت الاطلاع على أي تفاصيل إضافية حول أي محدد في الجدول، فارجع إلى توثيق العناصر الزائفــة في CSS في موسوعة حسوب.

تُضاف العناصر الزائفة إلى محددات CSS وتُمكن من تغيير أنماط أجزاء معينة من العنصر.

سِمة content هي سِمة مطلوبة لإنشاء العناصر الزائفة، ويمكن أن تأخذ قيمة فارغة `content: ""`:

```
div::after {
    content: 'after';
    color: red;
    border: 1px solid red;
}

div {
    color: black;
    border: 1px solid black;
    padding: 1px;
}

div::before {
    content: 'before';
    color: green;
    border: 1px solid green;
}
```

النتيجة:



ا. استخدام العناصر الزائفة لتنسيق القوائم

غالبًا ما تُستخدم العناصر الزائفة لتغيير شكل إشارات القوائم وبالأخص القوائم الغير مرتبة `ul`.

الخطوة الأولى هي إزالة النقاط التي تسبق النص في القائمة.

```
ul {
    list-style-type: none;
}
```

ومن ثمة يمكنك إضافة عنصر بشكل معين بدلًا عنها، الشيفرة أدناه تُضيف مربع ذو تدرج لوني.

```
li::before {
    content: "";
    display: inline-block;
    margin-right: 10px;
    height: 10px;
    width: 10px;
    background: linear-gradient(red, blue);
}
```

يكون ملف HTML مثلًا:

```
<ul>
    <li>Test I</li>
    <li>Test II</li>
</ul>
```

النتيجة:

1.4.7 حساب عمق التحديد

لكل مُحدِّد من محددات CSS عُمق محدد، ويزيد عُمق التحديد بزيادة عدد المحددات في السلسـلة. ويمكن تقسيم المحددات من حيث عُمق التحديد إلى ثلاث مجموعات كما هو موضح بالجدول التالي:

المجموعة	تتكون من	مثال
A	محددات المُعرِّفات ID#.	`#foo`
B	محددات الأصناف class، ومحددات السِمات، ومحددات الأصناف الزائفة pseudo-classes.	`.bar [title] [colspan=2] :hover :nth-child(2)`
c	محددات الأنواع والعناصر الزائفة.	`div, li, ::before, ::first-letter`

مُحددات المجموعـة A لهـا عُمـق التحديـد الأكـبر، تليهـا المجموعـة B، ثُم المجموعـة c. المحـدد العـام (*) والمُجمِّعات (مثل > و ~) ليس لها عُمق محدد.

إليك مثال عن عُمق التحديد لبعض محددات CSS:

```
#foo #baz {} /* a=2, b=0, c=0 */
#foo.bar {} /* a=1, b=1, c=0 */
#foo {} /* a=1, b=0, c=0 */
.bar:hover {} /* a=0, b=2, c=0 */
div.bar {} /* a=0, b=1, c=1 */
:hover {} /* a=0, b=1, c=0 */
[title] {} /* a=0, b=1, c=0 */
.bar {} /* a=0, b=1, c=0 */
div ul + li {} /* a=0, b=0, c=3 */
p::after {} /* a=0, b=0, c=2 */
*::before {} /* a=0, b=0, c=1 */
::before {} /* a=0, b=0, c=1 */
div {} /* a=0, b=0, c=1 */
* {} /* a=0, b=0, c=0 */
```

إليك مثال عن كيف تتعامل المتصفحات مع عُمق التحديد:

```
#foo {
    color: blue;
}
.bar {
    color: red;
    background: black;
}
```

بما أن عُمق التحديد للمُحدِّد ID أكبر من عمق التحديد لمحدد الصنف تُطبق القاعـدة `color: blue` من المُحدد `#foo`، والقاعدة `background: black` من المحدد `.bar` .

مثال:

```
.bar {
    color: red;
    background: black;
}
```

```
.baz {
    background: white;
}
```

المحددان في هذا المثال لهما نفس عُمق التحديد، وفي هذه الحالة يستخدم المتصفح طبيعة CSS التتابعية، أي أنه يُطبق الأنماط المُعرَّفة داخل المحدد `.bar` ثُم ينتقل لتطبيق الأنماط المُعرَّفة داخل المحدد `.baz` وتكون النتيجة هي تطبيق القاعدتين `color:red` و `background: white`.

ا. كيفية التحكم في عُمق التحديد

في المثال أعلاه، للتأكد من أن القاعدة `color:red` والمعرفة في المحدد `.bar` هي التي ستُطبق، يمكننا إضافة محددات أخرى لزيادة عُمق هذا المحدد، والمثال التالي يوضح ذلك:

```
span.bar {} /* a=0, b=1, c=1 */
.baz {} /* a=0, b=1, c=0 */
```

المحدد `span.bar` يتكون من مُحددين ويكون عمق التحديد له هو 2، أما المحدد `.baz` فعمق التحديد له 1، لذلك يأخذ العنصر قيمة الخاصية color المعرفة داخل المحدد `span.bar`.

ويُمكن أيضًا تكرار المحدد `.bar` للحصول على نفس النتيجة.

```
.bar .bar {} /* a=0, b=2, c=0 */
.baz {} /* a=0, b=1, c=0 */
```

ب. مُعرِّف !important والأنماط السطرية inline styles

للمُعرِّف !important والأنماط السطرية (التي تُعرف باستعمال سِمة `style` لعنصر HTML) عُمق تحديد أكبر من كل محددات CSS.

> يُفضل زيادة عمق التحديد بإضافة محددات إضافية بدلًا عن استخدام `!important` ولا نستعملها إلا للضرورة.

يُمكنك حساب عمق التحديد باستخدام أداة قياس عُمق التحديد. مثال:

```
#myDiv {
    font-weight: bold !important;
}
#outerdiv #myDiv {
    font-weight: normal;
}
```

يكون ثُخن الخط النهائي في هذا المثال هـو `bold`، وذلـك بسـبب اسـتخدام المُعـرف `!important` ممـا يمنع تطبيق القاعدة `font-weight: normal` على الرغم من أن المُحدد المستعمل لتعريفها له عُمق تحديد أكبر من عُمق تحديد المُحدد `#myDiv`.

1.4.8 توريث التنسيق Cascading

وراثة الخاصيات هي من إحدى أهم المميزات في CSS، حيث تمكنـك من تحديـد الأنمـاط المشـتركة لكـل العناصر في الصفحة في مكان واحد بدلًا عن تحديدها لكـل عنصـر منفـرد ممـا يسـهل الوصـول إليهـا وتعـديلها في المستقبل.

من الخاصيات التي تُوَّرث تلقائيًا الخاصية `font` و `color` و `text-align` و `line-height`.

مثال:

- ملف CSS

```
#myContainer {
    color: red;
    padding: 50px
}
```

- ملف HTML

```
<div id="myContainer">
    <h3>Some header</h3>
    <p>Some paragraph</p>
</div>
```

في هذا المثال، يرث العنصران `h3` و `p` اللون الأحمر تلقائيًـا من العنصـر الأب لهمـا، أمـا بالنسـبة للخاصـية `padding` فهي لا تُوَرِّث قيمتها تلقائيًا.

ا. الوراثة الإجبارية

بعض الخاصيات مثل `padding`، و `margin` لا تُوَرِّث قيمتها للعناصر الأبنـاء تلقائيًـا، ولكن يمكن إجبارهـا على ذلك باستخدام الكلمة المحجوزة `inherit`.

مثال:

- ملف CSS

```
#myContainer {
    color: red;
    padding: 5px;
}
#myContainer p {
    padding: inherit;
}
```

- ملف HTML

```
<div id="myContainer">
    <h3>Some header</h3>
    <p>Some paragraph</p>
</div>
```

في هذا المثال يرث العنصران `h3` و `p` كِلا القاعدتين `color: red` و `padding: 5px` من العنصر الأب.

ب. كيفية معالجة تضارب قواعد CSS

يستخدم توريث التنسيق مع عُمق التحديد لتحديد القيمة النهائية التي تأخذها خاصية CSS، ويحددان الآلية التي تُستخدم لمعالجة التضارب في قواعد CSS.

يتعلق موضوع تضارب قواعد CSS بكيفية تحميل ملفات CSS في المتصفح، حيث يقرأ المتصـفح ملفـات CSS من مصادر مختلفة ويقوم بتحميلها حسب الترتيب التالي:

1. التنسيقات الافتراضية للمتصفح
2. التنسيقات التي يُعرفها المستخدم على المتصفح.
3. تنسيقات CSS والأنماط المُعرفة داخل العنصر `<style>`.
4. التنسيقات السطرية inline styles.

عندما تُستخدم نفس القاعدة بقيمتين مختلفين في نفس العنصر تُطبَّق القاعدة ذات عُمق التحديد الأكـبر، وفي حال تساوت أعماق التحديد، تُطبَّق القاعدة الأخيرة في الملف.

مثال:

```
<style>
.mystyle { color: blue; } /* specificity: 0,  1, 0 */
div { color: red; } /* specificity: 0, 0, 1 */
</style>

<div class="mystyle">Hello World</div>
```

بما أنَّ عمق التحديد لمحددات الأصناف class أكبر من عمق تحديد محـددات العناصـر، يكـون لـون الخـط الناتج أزرق. مثال:

- ملف CSS خارجي

```
.class {
    background: #fff;
}
```

- ملف CSS داخلي (مُحدد بالعنصر <style>)

```
<style>
    .class {
        background: #000;
    }
</style>
```

بما أن عُمق التحديد متساوي، يُطبَّق المتصفح الأنماط الموجودة في آخر ملف حُمِّل إليـه، أي أنـه سـيُطبِّق القاعدة background: #000.

إليك مثال عن استخدام قواعد توريث التنسيق مع قواعد عمق التحديد:

```
<style>
    body > .mystyle { background-color: blue; } /* specificity: 0, 0,
1, 1 */
    .otherstyle > div { background-color: red; } /* specificity: 0, 0,
1, 1 */
</style>

<body class="otherstyle">
<div class="mystyle">Hello World</div>
</body>
```

نُلاحظ تساوي عمق التحديد لكل من المحددين ولذلك يتَّبع المتصفح قواعد توريث التنسيق أي انه سيُطَبِّق الأنماط حسب ترتيب ظهورها في الملف، أي أنه سيُطبَّق في هذه الحالة اللون الأزرق للخاصية ومن ثم ينتقل للقاعدة الثانية والتي تُغيير لون الخلفية للون الأحمر، أي أن اللون النهائي للخلفية سكون أحمر.

مثال:

```
<style>
div {
    font-size: 7px;
    border: 3px dotted pink;
    background-color: yellow;
    color: purple;
}
body.mystyle > div.myotherstyle {
    font-size: 11px;
    background-color: green;
}
#elmnt1 {
    font-size: 24px;
    border-color: red;
}
.mystyle .myotherstyle {
    font-size: 16px;
    background-color: black;
    color: red;
}
</style>

<body class="mystyle">
    <div id="elmnt1" class="myotherstyle">
        Hello, world!
    </div>
</body>
```

تأخذ التنسيقات القيم التالية:

- حجم الخط: بما أن المحدد `#elemnt1` له أعلى عمق للعنصر `div`، يكون حجم الخط النهائي هو 24 بكسل.

- الإطـار: بمـا أن المحـدد `#elemnt1` لـه أعلى عمـق للعنصـر `div`، تُطبَّق القاعـدة `border: 3px dotted red`.
- لـــــون الخلفيـــــة: حُـــــدد لـــــون الخلفيـــــة في ثلاث محـــــددات هي `div` والمحـــــدد `body.mystle > div.myotherstyle`، والمحدد `.mystyle` وبمـا أن عمـق التحديـد للمحـدد الثاني هو الأكبر، يصبح لون خلفية العنصر أخضر.
- لـــــون المحتـــــوى color: حُـــــدد لـــــون المحتـــــوى في ثلاث محـــــددات هي `div` والمحـــــدد `body.mystle > div.myotherstyle` والمحـدد `.mystyle` وبمـا أن عمـق التحديـد للمحـدد الأخير هو الأكبر، يصبح لون محتوى العنصر أحمر.

1.5 الوحدات

يوضح الجدول التالي قائمة بالوحدات المتاحة في CSS:

الوحدة	الوصف
`%`	تحدد الطول كنسبة مئوية من طول العنصر الأب أو العنصر الحالي.
`em`	تحدد الطول كنسبة من حجم خط العنصر الأب (font-size) (مثلًا 2em تعني ضعف حجم الخط).
`rem`	تحدد الطول كنسبة من حجم خط العنصر الجذري الذي هو العنصر html.
`vw`	تحدد الطول كنسبة من عرض الشاشة.
`vh`	تحدد الطول كنسبة من ارتفاع الشاشة.
`vmin`	تقيس الأطوال نسبةً ل 1% من البُعد الأصغر لشاشة العرض
`vmax`	تقيس الأطوال نسبةً ل 1% من البُعد الأكبر لشاشة العرض
`cm`	تحدد الطول بالسنتمترات.
`mm`	تحدد الطول بالمليمترات.
`in`	تحدد الطول بالبوصة (2.54cm= 96px= 1in)
`px`	تحدد الطول بالبكسلات.
`pt`	تحدد الطول بالنقاط (1/72in = 1pt)
`pc`	1pc = 12pt
`s`	تحدد الزمن بالثواني.
`ms`	تحدد الزمن بالملي ثانية.
`ex`	تحدد الأطوال نسبة لارتفاع الخط.
`ch`	تُحدد الأطوال نسبةً لعرض المحرف (0).
`fr`	وحدة كسرية (تستخدم مع التخطيط الشبكي).

1.5.1 إنشاء عناصر قابلة للتوسع باستخدام ems و rems

يُمكن استخدام `rem` والتي تُحدد الأطوال استنادًا إلى حجم الخـط، مـع وحـدة `em` لإنشـاء عناصـر ذات حجم ديناميكي بتغيير حجم الخط. إليك مثال:

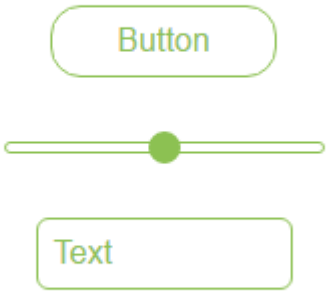
- ملف HTML

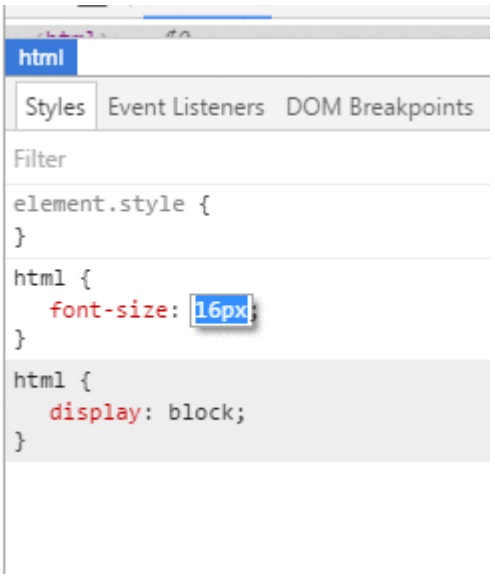
```
<input type="button" value="Button">
<input type="range">
<input type="text" value="text">
```

- ملف CSS

```
html {
    font-size: 16px;
}

input[type="button"] {
    font-size: 1rem;
    padding 0.5em 2em;
}

input[type="range"] {
    font-size: 1rem;
    width: 10em;
}

input[type="text"] {
    font-size: 1rem;
    padding: 0.5em;
}
```

النتيجة (شاهد الصورة متحركة بالضغط عليها):





1.5.2 ضبط حجم الخط باستخدام rem

الفرق بين em و rem هو:

- em: تقيس الأطوال كنسبة من حجم خط العنصر الأب.
- rem: تقيس الأطوال كنسبة من حجم خط عنصر

بافتراض أنَّ القيمة الابتدائية لحجم الخط هي 16 بكسل، يُمكن التعبير عن أحجام الخطوط باستخدام الوحدة `rem` كالتالي:

```
10px = 0.625rem
12px = 0.75rem
14px = 0.875rem
16px = 1rem
18px = 1.125rem
20px = 1.25rem
24px = 1.5rem
30px = 1.875rem
32px = 2rem
```

مثال:

```
html {
    font-size: 16px;
}
```

```
h1 {
    font-size: 2rem; /* 32px */
}

p {
    font-size: 1rem; /* 16px */
}

li {
    font-size: 1.5rem; /* 24px */
}
```

1.5.3 وحدات vmin و vmax

شرحهما:

- vmin: تقيس الأطوال نسبةً ل 1% من البُعد الأصغر لشاشة العرض، فإذا كان طول شاشة العرض أقل من عرضها ستقاس الأطوال نسبة إليه والعكس كذلك.
- vmax: تقيس الأطوال نسبةً ل 1% من البُعد الأكبر لشاشة العرض، فإذا كان طول شاشة العـرض أكـبر من عرضها ستقاس الأطوال نسبة إليه والعكس كذلك.

وبمعنى آخر:

- 1vmin تساوي القيمة الأصغر بين القيمتين 1vh و 1vw.
- 1vmax تساوي القيمة الأكبر بين القيمتين 1vh و 1vw.

1.5.4 الوحدات vh و vw

شرحهما:

- vh: ترمز لارتفاع شاشة العرض، وتقيس الأطول نسبةً لارتفاع شاشة العرض.
- vw: ترمز لعرض شاشة العرض، وتقيس الأطول نسبةً لعرض شاشة العرض.

مثال:

```
div {
    width: 20vw;
    height: 20vh;
```

```
}
```

1.5.5 استخدام النسب المئوية

المعادلة العامة:

```
(parent container width) * (percentage(%)) = output
```

إليك مثال، إذا كان عرض العنصر الأب 100 بكسل، وأضفنا القاعـدة `width: 50%` للعنصـر الابن، يكـون عرض العنصر الابن نصف عرض الأب، أي 50 بكسل. إليك مثال:

- ملف HTML

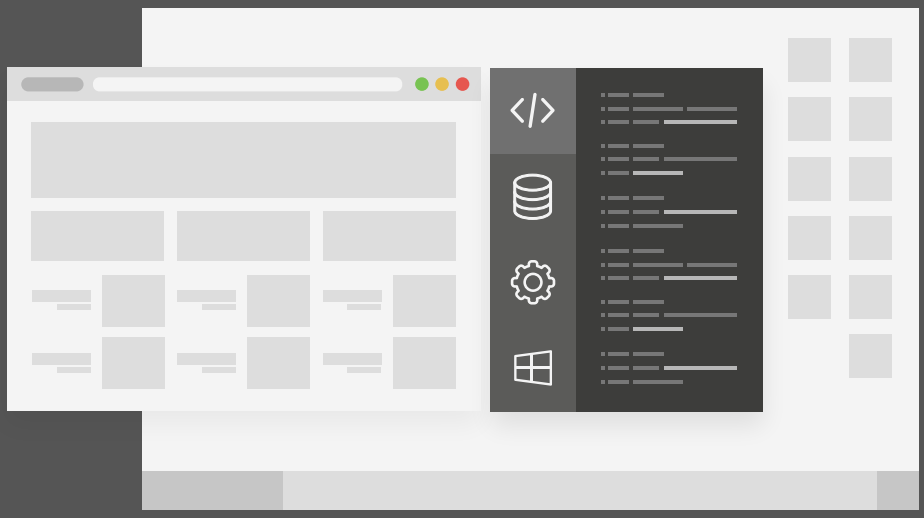
```
<div class="parent">
    PARENT
    <div class="child">
        CHILD
    </div>
</div>
```

- ملف CSS

```
*{
    color: #CCC;
}
.parent{
    background-color: blue;
    width: 100px;
}
.child{
    background-color: green;
    width: 50%;
}
```

النتيجة:

دورة علوم الحاسوب



دورة تدريبية متكاملة تضعك على بوابة الاحتراف
في تعلم أساسيات البرمجة وعلوم الحاسوب

التحق بالدورة الآن

2. تخطيط الصفحة وضبط محاذاة العناصر

2.1 الخاصية display

تُحدد الخاصية `display` ما هو نوع صندوق العرض الذي سيستخدم لعرض العنصر. تؤخذ قيمة الخاصية `display` في HTML من مواصفة HTML أو من الأنماط الافتراضية للمتصفح أو الأنماط التي يُعرِّفها المستخدم، أما القيمة الافتراضية لخاصية `display` لمعظم عناصر HTML هي `block` أو `inline` (بعض العناصر لها قيم افتراضية أخرى). ويمكن أن تأخذ إحدى القيم التالي:

القيمة	الوصف
`none`	تُخفي العنصر وتمنعه من أن يشغل مساحة في الصفحة.
`block`	تعرض العنصر كعنصر كُتَلي block-element، يأخذ كل المساحة الأفقية (العرض) المتاحة، وينتقل للسطر التالي بعد نهاية العنصر.
`inline`	تعرض العنصر كعنصر سطري inline-element، لا يمتلك عرض محدد، ولا ينتقل للسطر التالي بعد نهاية العنصر.
`inline-block`	مزيج بين عناصر الكتلة والعناصر السطرية، لا ينتقل للسطر التالي بعد نهاية العنصر ولكن يمكن أن يمتلك عرض محدد.
`inline-flex`	تعرض حاوية flex كحاوية سطرية.
`inline-table`	تعرض العنصر كجدول سطري.
`grid`	لها سلوك مشابه لسلوك عناصر الكتلة، وتُعرض العناصر بداخلها استنادًا إلى النموذج الشبكي grid model.
`flex`	لها سلوك مشابه لسلوك عناصر الكتلة، وتُعرض العناصر استنادًا إلى نموذج الصندوق المرن flexbox model.
`inherit`	ترث قيمة الخاصية من العنصر الأب.

القيمة	الوصف
`initial`	تُرجع القيمة الابتدائية للخاصية.
`table`	تجعل العنصر يسلك سلوك مماثل للعنصر `<table>`.
`table-cell`	تجعل العنصر يسلك سلوك مماثل للعنصر `<td>`.
`table-column`	تجعل العنصر يسلك سلوك مماثل للعنصر `<col>`.
`table-row`	تجعل العنصر يسلك سلوك مماثل للعنصر `<tr>`.
`list-item`	تجعل العنصر يسلك سلوك مماثل للعنصر `<li>`.

2.1.1 العناصر السطرية inline elements

تأخذ العناصر السطرية عرض محتواها، ولا يُمكن إضافة عنصر غير سطري non-inline element داخل عنصر سطري. مثال:

```
<span>This is some <b>bold</b> text!</span>
```

النتيجة:

This is some **bolded** text!

2.1.2 العناصر الكتلية block elements

تأخذ العناصر الكتلية أكبر عرض متاح لها، وتبدأ بسطر جديد وتنتقل لسطر جديد بعد نهاية العنصر، ويُمكن إضافة أي نوع من العناصر داخلها. مثال:

```
<div>Hello world!</div><div>This is an example!</div>
```

النتيجة:

Hello world!
This is an example!

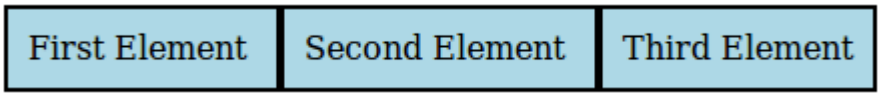
2.1.3 القيمة inline-block

القيمة `inline-block` هي مزيج بين خاصيات العناصر السطرية وعناصر الكتلة، فهي تعرض العناصر في سطر (صف) واحد تمامًا مثل `inline` ولكنها تسمح باستعمال الخواص `padding` و `margin` و`height` (والتي ليس لها أثر على العناصر السطرية) معها.

المثال التالي يوضح الفرق بين القيم `inline` و `block` و `inline-block`:

```
<!-- Inline -->
<style>
li {
    display : inline;
    background : lightblue;
    padding:10px;
    border-width:2px;
    border-color:black;
    border-style:solid;
}
</style>
<ul>
    <li>First Element </li>
    <li>Second Element </li>
    <li>Third Element </li>
</ul>
```

النتيجة:



مثال عن قيمة `block`:

```
<!-- block -->
<style>
li {
    display : block;
    background : lightblue;
    padding:10px;
    border-width:2px;
    border-color:black;
    border-style:solid;
}
</style>
<ul>
    <li>First Element </li>
    <li>Second Element </li>
```

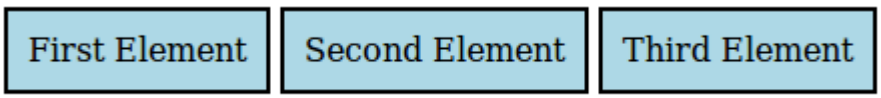
```
    <li>Third Element </li>
</ul>
```

النتيجة:

مثال عن قيمة `inline-block`:

```
<!-- inline-block -->
<style>
li {
    display : inline-block;
    background : lightblue;
    padding:10px;
    border-width:2px;
    border-color:black;
    border-style:solid;
}
</style>
<ul>
    <li>First Element </li>
    <li>Second Element </li>
    <li>Third Element </li>
</ul>
```

النتيجة:



2.1.4 إخفاء العناصر

تُستخدَم القاعدة `display: none` لإخفاء العنصر ومنعه من أن يشغل مساحة في الصفحة، على عكس القاعدة `visibility: hidden` والتي تُخفي العنصر ولكن تبقى المساحة التي يشغلها من الصفحة ثابتة مما يؤثر على موضع عرض العناصر التالية له.

مثال:

- ملف HTML

```
<div id="myDiv">some content</div>
```

- ملف CSS

```
#myDiv {
    display: none;
}
```

2.1.5 إنشاء تخطيط جدول باستخدام العنصر div

انظر المثال:

```
<style>
    table {
        width: 100%;
    }
</style>

<table>
    <tr>
        <td>
            I'm a table
        </td>
    </tr>
</table>
```

يُمكن الحصول على نفس ناتج الشيفرة أعلاه باستخدام العنصر `div` كما هو موضح بالشيفرة التالية:

```
<style>
    .table-div {
```

```
        display: table;
    }
    .table-row-div {
        display: table-row;
    }
    .table-cell-div {
        display: table-cell;
    }
</style>

<div class="table-div">
    <div class="table-row-div">
        <div class="table-cell-div">
            I behave like a table
        </div>
    </div>
</div>
```

2.2 تخطيط Flexbox

يستخدم نموذج أو تخطيط Flexbox لتوزيع العناصر في مساحة العرض المتاحة توزيعًا مرنًا بحيث تتمدد أو تنكمش مع تَغَيُّر حجم الشاشة.

2.2.1 توسيط العناصر أفقيًا ورأسيًا

إليك مثال حول توسيط عنصر واحد:

- ملف HTML:

```
<div class="aligner">
    <div class="aligner-item">
        some content
    </div>
</div>
```

- ملف CSS

```
.aligner {
```

```
        display: flex;
        align-items: center;
        justify-content: center;
    }

    .aligner-item {
        max-width: 50%;
    }
```

اطَّلع على تجربة حيَّة للمثال أعلاه على Codepen.

إليك شرح الخاصيتان في المثال:

- `align-items`: تأخذ القيمة `center` وتُوسِّط العناصر أفقيًا إذا كان اتجاه الحاويـة رأسـي، ورأسـيًا إذا كان اتجاه الحاوية أفقي.
- `justify-content`:تأخذ القيمة `center` و تُوسِّط العناصر أفقيًا إذا كان اتجاه الحاوية أفقي، ورأسيًا إذا كان اتجاه الحاوية رأسي.

سنستعمل شيفرة HTML التالية في كل الأمثلة اللاحقة:

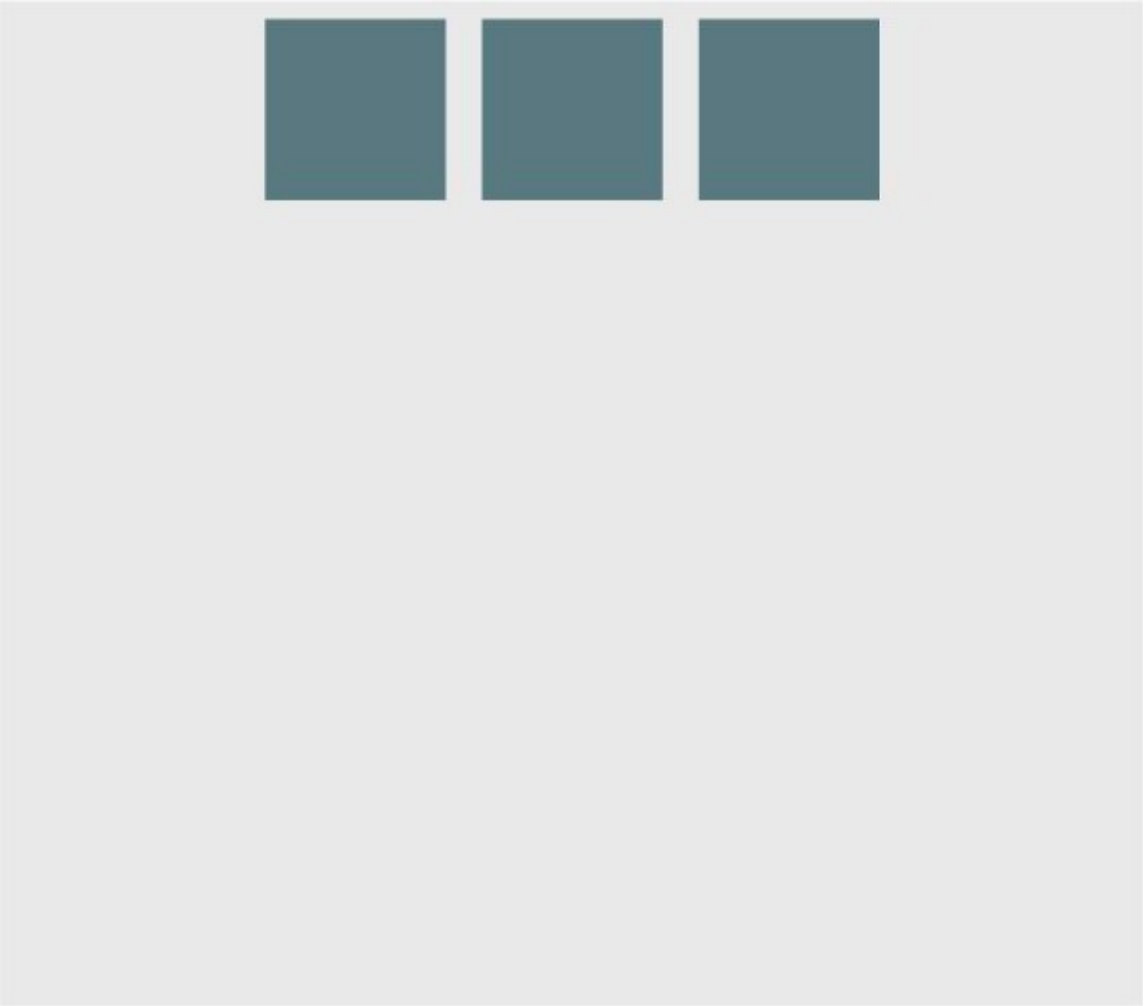
```
<div id="container">
    <div></div>
    <div></div>
    <div></div>
</div>
```

١. التوسيط أفقيًا في حاوية أفقية

مثال:

```
div#container {
    display: flex;
    flex-direction: row;
    justify-content: center;
}
```

النتيجة:



اطَّلع على تجربة حية للمثال على JSFiddle.

ب. التوسيط رأسيًا في حاوية رأسية

مثال:

```
div#container {
    display: flex;
    flex-direction: column;
    justify-content: center;
}
```

النتيجة:

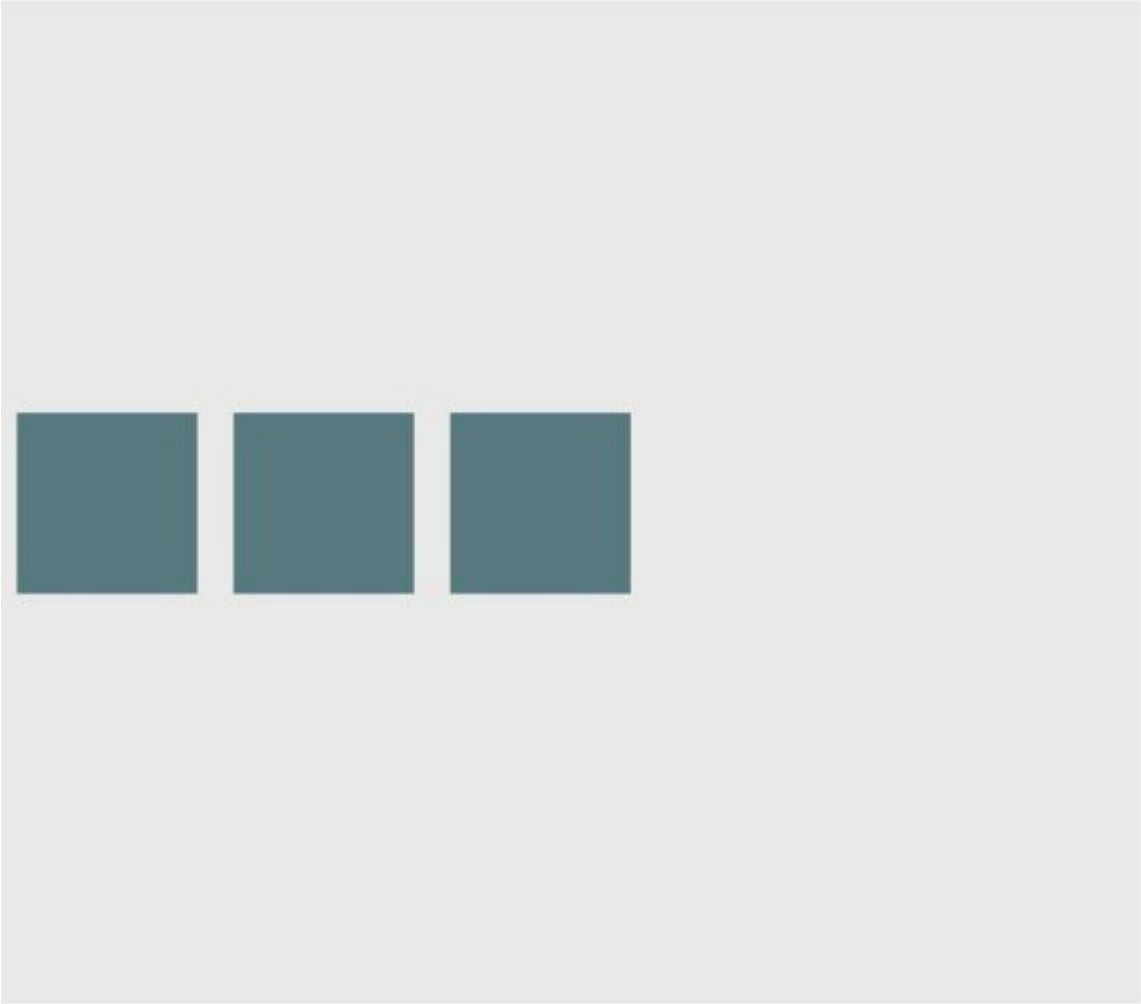
اطَّلع على تجربة حية للمثال على JSFiddle.

ج. التوسيط رأسيًا في حاوية أفقية

مثال:

```
div#container {
    display: flex;
    flex-direction: row;
    align-items: center;
}
```

النتيجة:



اطّلع على تجربة حية للمثال على JSFiddle.

د. التوسيط أفقيًا في حاوية رأسية

مثال:

```
div#container {
    display: flex;
    flex-direction: column;
    align-items: center;
}
```

النتيجة:

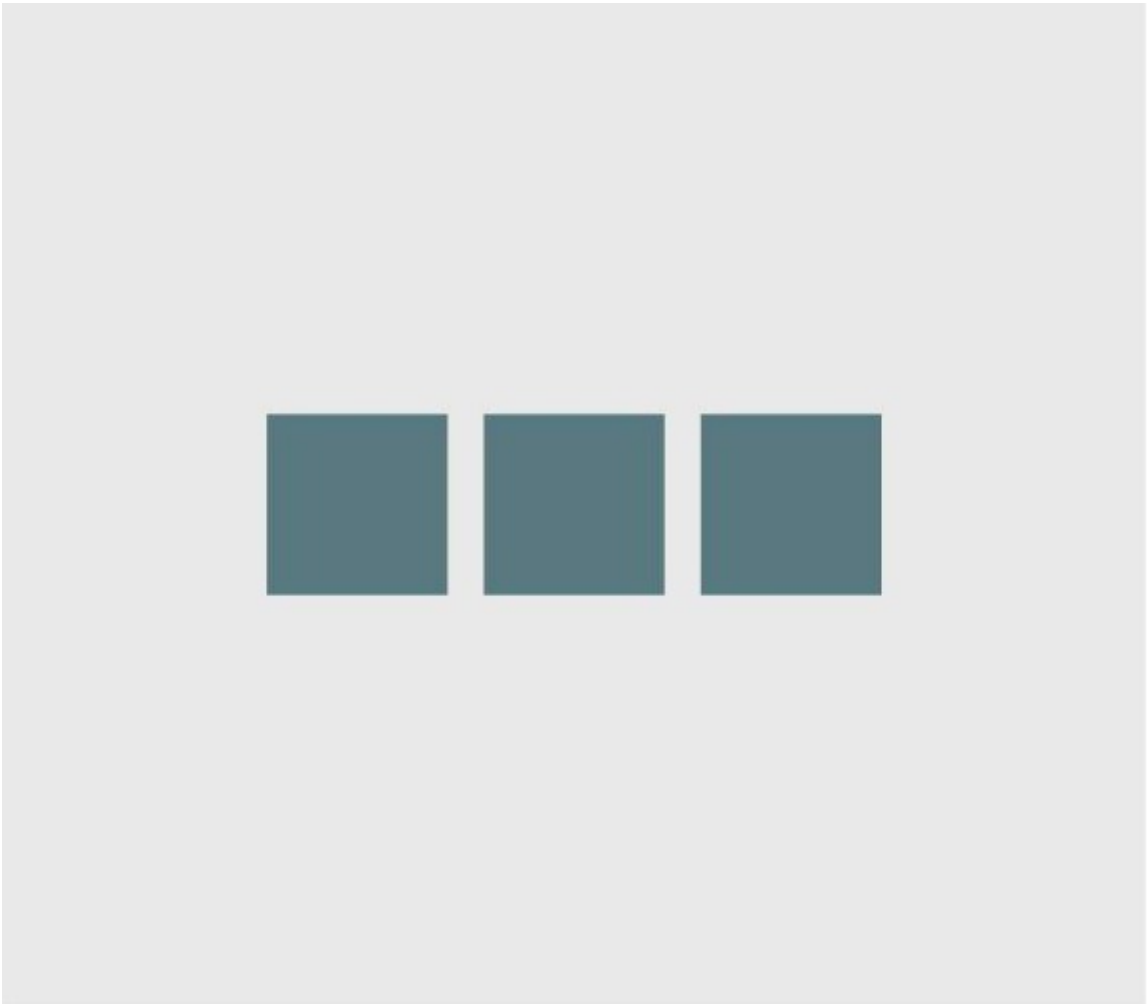
اطلع على تجربة حية للمثال على JSFiddle.

ه. التوسيط أفقيًا ورأسيًا في حاوية أفقية

مثال:

```
div#container {
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
}
```

النتيجة:



اطلع على تجربة حية للمثال على JSFiddle.

و. التوسيط أفقيًا ورأسيًا في حاوية رأسية

مثال:

```
div#container {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
```

النتيجة:

اطلع على تجربة حية للمثال على JSFiddle.

2.2.2 إنشاء تذييل ثابت لصفحة

انظر المثال التالي:

- ملف HTML

```
  <div class="header">
      <h2>Header</h2>
  </div>

  <div class="content">
      <h1>Content</h1>
      <p>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Integer nec odio. Praesent libero.
```

```
  Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh
elementum imperdiet. Duis sagittis
  ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta.
Mauris massa. Vestibulum lacinia
  arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per
conubia nostra, per inceptos
  himenaeos. Curabitur sodales ligula in libero.
      </p>
  </div>

  <div class="footer">
      <h4>Footer</h4>
  </div>
```

- ملف CSS

```
  html, body {
      height: 100%;
  }
  body {
      display: flex;
      flex-direction: column;
  }
  .content {
      flex: 1 0 auto;
  }
  .header, .footer {
      background-color: grey;
      color: white;
      flex: none;
  }
```

اطَّلع على تجربة حيَّة للمثال أعلاه على JSFiddle.

2.2.3 توزيع العناصر بشكل مثالي داخل الحاوية

انظر المثال التالي:

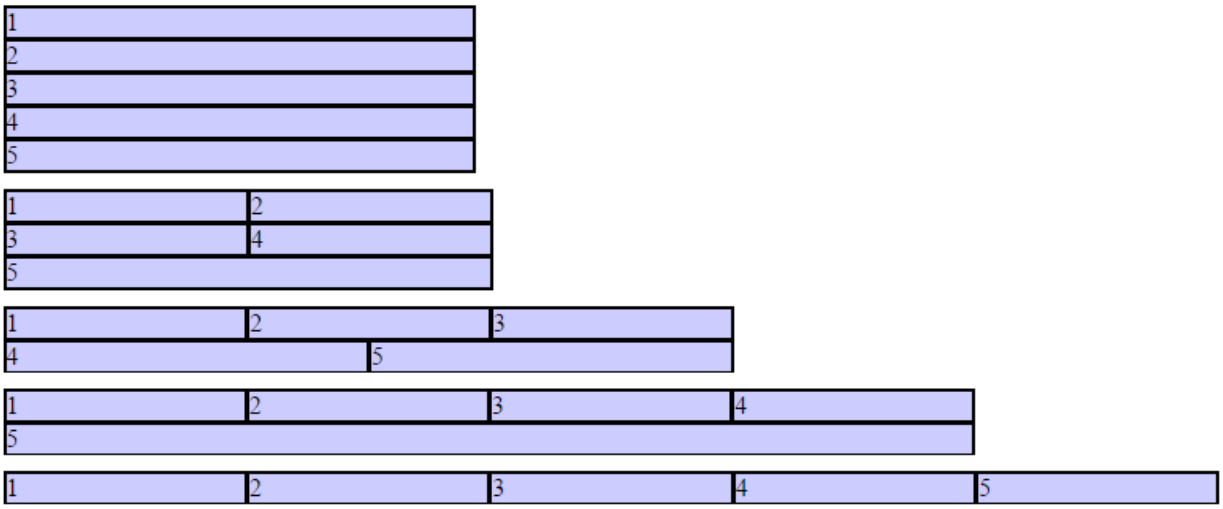
- ملف HTML

```
<div class="flex-container">
    <div class="flex-item">1</div>
    <div class="flex-item">2</div>
    <div class="flex-item">3</div>
    <div class="flex-item">4</div>
    <div class="flex-item">5</div>
</div>
```

- ملف CSS

```
.flex-container {
    background-color: #000;
    height: 100%;
    display:flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: flex-start;
    align-content: stretch;
    align-items: stretch;
}
.flex-item {
    background-color: #ccf;
    margin: 0.1em;
    flex-grow: 1;
    flex-shrink: 0;
    flex-basis: 200px;
}
```

النتيجة:



اطلع على تجربة حية لهذا المثال على JSFiddle.

2.2.4 إنشاء تخطيط باستخدام حاوية flex

أغلب التخطيطات اليوم تتكون من رأس صفحة header بارتفاع ثابت وثلاثة أعمدة وذيل footer ثابت.

- ملف HTML

```
<div class="container">
    <header class="header">Header</header>
    <div class="content-body">
        <main class="content">Content</main>
        <nav class="sidenav">Nav</nav>
        <aside class="ads">Ads</aside>
    </div>
    <footer class="footer">Footer</footer>
</div>
```

- ملف CSS

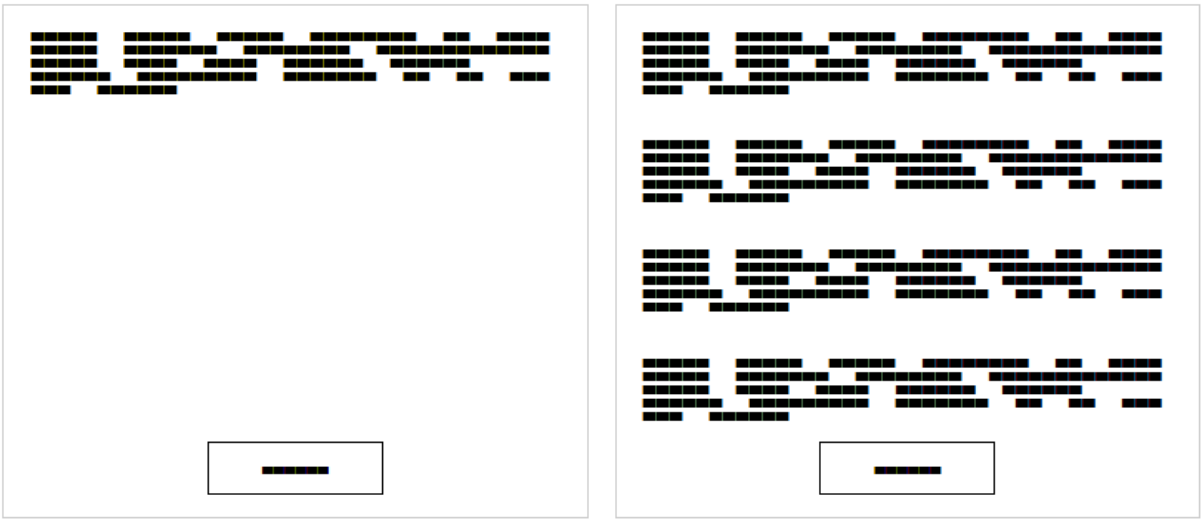
```
body {
    margin: 0;
    padding: 0;
}
.container {
    display: flex;
    flex-direction: column;
```

```
        height: 100vh;
    }
    .header {
        flex: 0 0 50px;
    }
    .content-body {
        flex: 1 1 auto;
        display: flex;
        flex-direction: row;
    }
    .content-body .content {
        flex: 1 1 auto;
        overflow: auto;
    }
    .content-body .sidenav {
        order: -1;
        flex: 0 0 100px;
        overflow: auto;
    }
    .content-body .ads {
        flex: 0 0 100px;
        overflow: auto;
    }
    .footer {
        flex: 0 0 50px;
    }
```

اطلع على تجربة حية لهذا المثال على JSFiddle.

2.2.5 محاذاة الأزرار داخل البطاقات

انظر المثال التالي الموضح في الصورة:



التخطيط الموضح في الصورة أعلاه هو من التخطيطات الشائعة جدًا، ويوضح المثال التالي كيفيـة التوصـل إليه باستخدام حاوية flex.

- ملف HTML

```
  <div class="cards">
      <div class="card">
          <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur
reprehenderit culpa esse enim
          mollit labore dolore voluptate ullamco et ut sed qui
minim.</p>
          <p><button>Action</button></p>
      </div>

      <div class="card">
          <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur
reprehenderit culpa esse enim
          mollit labore dolore voluptate ullamco et ut sed qui
minim.</p>
          <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur
reprehenderit culpa esse enim
          mollit labore dolore voluptate ullamco et ut sed qui
minim.</p>
          <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur
reprehenderit culpa esse enim
          mollit labore dolore voluptate ullamco et ut sed qui
minim.</p>
```

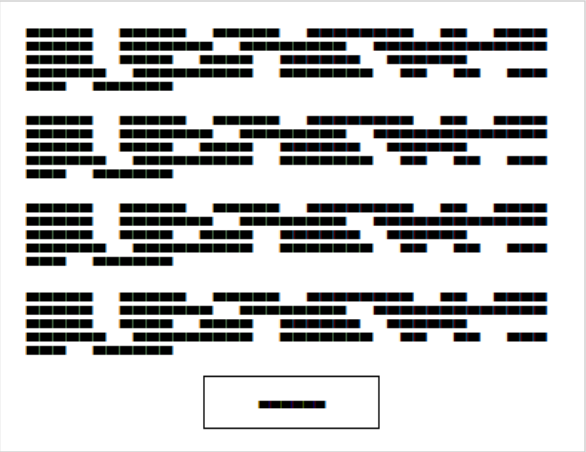
```
            <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur
reprehenderit culpa esse enim
            mollit labore dolore voluptate ullamco et ut sed qui
minim.</p>
            <p><button>Action</button></p>
        </div>
    </div>
```

أولًا نُطبق القاعدة `display: flex` على الحاوية للحصول على عمودين متساويين في الارتفاع.

- ملف CSS

```
    .cards {
        display: flex;
    }
    .card {
        border: 1px solid #ccc;
        margin: 10px 10px;
        padding: 0 20px;
    }
    button {
        height: 40px;
        background: #fff;
        padding: 0 40px;
        border: 1px solid #000;
    }p
    :last-child {
        text-align: center;
    }
```

النتيجة:



لدفع الزر إلى أسفل البطاقة يجب إضافة القاعدة `display: flex` للبطاقة، وتحديد اتجاه الحاوية ليكون عموديًا باستخدام القاعدة `flex-direction: column`، وأخيرًا نُطبق القاعدة `margin-top: auto` على آخر عنصر داخل البطاقة لدفعه إلى أسفلها.

- ملف CSS النهائي

```
.cards {
    display: flex;
}
.card {
    border: 1px solid #ccc;
    margin: 10px 10px;
    padding: 0 20px;
    display: flex;
    flex-direction: column;
}
button {
    height: 40px;
    background: #fff;
    padding: 0 40px;
    border: 1px solid #000;
}p
:last-child {
    text-align: center;
    margin-top: auto;
}
```

2.2.6 إنشاء حاويات متداخلة بارتفاعات متساوية

انظر المثال التالي:

- ملف HTML

```
<div class="container">
    <div style="background-color: red">
        Some <br />
        data <br />
        to make <br />
        a height <br />
    </div>
    <div style="background-color: blue">
        Fewer <br />
        lines <br />
    </div>
</div>
```

- ملف CSS

```
.container {
    display: flex;
    align-items: stretch
}
```

اطلع على تجربة حية على JSFiddle.

2.3 التخطيط الشبكي Grid

يُستخدم التخطيط الشبكي لتقسيم الصفحة إلى صفوف وأعمدة.

القيم المتاحة للخاصية هي:

```
display: grid / inline-grid
```

مثال:

- ملف HTML

```
<section class="container">
```

```
        <div class="item1">Item1</div>
        <div class="item2">Item2</div>
        <div class="item3">Item3</div>
        <div class="item4">Item4</div>
    </section>
```

- ملف CSS

```
    .container {
        display: grid;
    }
```

يـؤدي تطـبيق الشـيفرة أعلاه إلى تـرادف العناصـر فـوق بعضـها البعض، لـذلك نحتـاج لتعريـف خاصـيات إضافية هي:

1. `grid-columns`: وتستخدم لتحديد عدد الأعمدة وعرض كل عمود.
2. `grid-rows`: وتستخدم لتحديد عدد الصفوف وارتفاع كل صف.

```
    .container {
        display: grid;
        /* إنشاء ثلاثة أعمدة كل منها بعرض 50 بكسل */
        grid-columns: 50px 50px 50px;
        /* إنشاء صفين كل منهما بإرتفاع 50 بكسل */
        grid-rows: 50px 50px;
    }
```

ومن الممكن أيضًا تحديد تـرتيب العناصـر في التخطيـط الشـبكي عن طريـق الخاصـيتين `grid-column` و `grid-row` واللتان تحددان العمود والصف اللذان سيُعرض فيهما العنصر.

```
    .container .item1 {
        grid-column: 1;
        grid-row: 1;
    }
    .container .item2 {
        grid-column: 2;
        grid-row: 1;
    }
    .container .item3 {
```

```
    grid-column: 1;
    grid-row: 2;
}
.container .item4 {
    grid-column: 2;
    grid-row; 2;
}
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

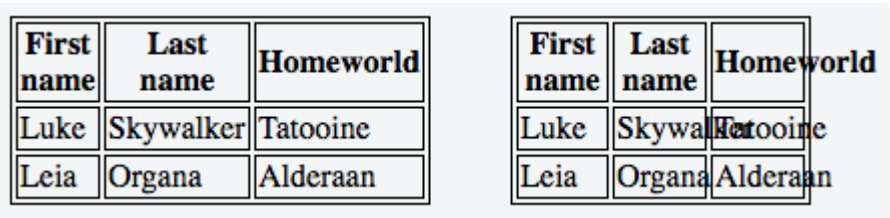
انظر دعم المتصفحات للقاعدة display: gird.

2.4 التخطيط الجدولي table

2.4.1 الخاصية table-layout

تُستخدم الخاصية `table-layout` لتحديد كيفية إنشاء الجدول ووضع العناصر بداخله، وتأخذ إحدى القيمتين:

1. `auto`: وتسمح للجدول بأخذ العرض اللازم لمنع طفحان المحتوى خارج الجدول.
2. `fixed`: وتُعطي الجدول العرض المحدد له بالخاصية `width` فقط، ولا تسمح له بالتمدد مما يؤدي لطفحان المحتوى خارج الجدول في حال عدم وجود مساحة كافية له.



First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

2.4.2 الخاصية empty-cells

تحدد الخاصية `empty-cells` ما إذا كانت الخلايا الفارغة ستُعرض أم لا، ولا يكون لها أثرٌ إلا إذا كانت قيمة الخاصية `border-collapse` هي `separate`، وتأخذ إحدى القيمتين `show` والتي تعني إظهار الخلايا الفارغة أو `hide` والتي تعني إخفائها.

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

2.4.3 الخاصية border-collapse

تحدد الخاصية `border-collapse` ما إذا كان إطار الجدول منفصلًا عن إطار الخلايا (أي لكلٍ واحدٍ منها إطارٌ خاصٌ به) أم أنه مدمج معها (أي تتشارك الخلايا المتجاورة الإطار مع بعضها بعضًا)، وتأخذ إحدى القيمتين `separate` والتي تفصل بين الإطارات أو `collapse` والتي تدمج إطارات الخلايا المتجاورة.

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

2.4.4 الخاصية border-spacing

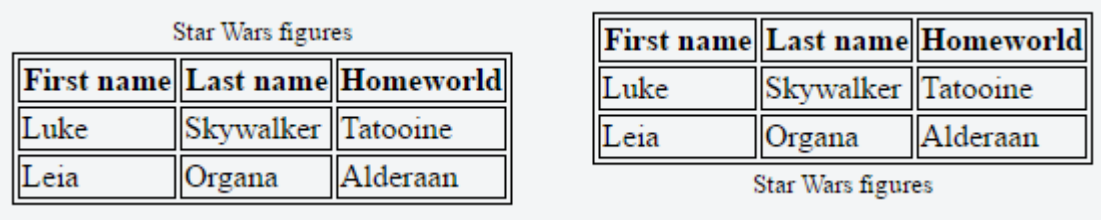
الخاصية `border-spacing` في CSS تُحدِّد المسافة بين إطارات خلايا الجداول المتجاورة. لن يكون لهذه الخاصية أثرٌ إلا إذا كانت قيمة الخاصية `border-collapse` هي `separate`. ويمكن تحديد المسافات الأفقية والرأسية بين الإطارات في آن واحد عن طريق تحديد قيمة واحدة للخاصية مثل `border-spacing: 2px`، أو تحديد قيمة مختلفة لكل واحدة على حدة مثل `border-spacing: 2px 4px`، حيث تمثل القيم الأولى المسافة بين الإطارات الأفقية وتحدد القيمة الثانية المسافة بين الإطارات الرأسية.

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

2.4.5 الخاصية caption-side

تحدد الخاصية `caption-side` موضع لافتة الجدول المُعرَّفة عبر العنصر `<caption>`، وتأخذ إحدى القيمتين `top` والتي تُضيف اللافتة أعلى الجدول أو `bottom` والتي تُضيفها أسفل الجدول.



Star Wars figures

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

Star Wars figures

ارجع إلى توثيق تخطيط الجداول في موسوعة حسوب وراجع هذا المثال على Codepen.

2.5 التخطيط الكتلي السطري Inline-block layout

2.5.1 إنشاء شريط تَنَقُّل علوي navigation bar

من الشائع إنشاء شريط تَنَقُّل علوي يحتوي على ازار للتنقل بين صفحات الموقع المختلفة، وفيه تكون المسافات بين الأزرار متساوية، ويكون للزر الأول فيه هامش أيمن فقط (وأيسر إذا كانت الصفحة باللغة العربية)، ولآخر زر هامش أيسر فقط (وأيمن إذا كانت الصفحة باللغة العربية).

مثال:

- ملف HTML

```
<nav>
    <ul>
        <li>abc</li>
        <li>abcdefghijkl</li>
        <li>abcdef</li>
    </ul>
</nav>
```

- ملف CSS

```
nav {
    width: 100%;
    line-height: 1.4em;
}
ul {
    list-style: none;
    display: block;
    width: 100%;
    margin: 0;
```

```
        padding: 0;
        text-align: justify;
        margin-bottom: -1.4em;
    }
    ul:after {
        content: "";
        display: inline-block;
        width: 100%;
    }
    li {
        display: inline-block;
    }
```

ملاحظات:

1. استُعملت العناصر nav، وul وli لمعناها الدلالي فقط (عناصر قائمة التنقل)، ويمكن استعمال أي عناصر أخرى.
2. استُخدمت قيمة سالبة مساوية لقيمة ارتفاع السطر للخاصية margin-bottom للتخلص من المساحة الفارغة التي يتسبب فيها العنصر الزائف :after.
3. إذا تقلَّص عرض الصفحة لحد لا يسمح بأن تكون جميع العناصر في سطر واحد، يتنقل جزء من العناصر للسطر التالي.

2.6 التوسيط Centering

2.6.1 توسيط العناصر باستخدام حاوية Flexbox

انظر المثال التالي:

- ملف HTML:

```
    <div class="container">
      <img src="http://lorempixel.com/400/200" />
    </div>
```

- ملف CSS:

```
    html,
```

```
body,
.container {
  height: 100%;
}
.container {
  display: flex;
  justify-content: center; /* التوسيط أفقيًا */
}
img {
  align-self: center; /* التوسيط رأسيًا */
}
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

- ملف HTML:

```
<img src="http://lorempixel.com/400/400" />
```

- ملف CSS:

```
html,
body {
  height: 100%;
}
body {
  display: flex;
  justify-content: center; /* التوسيط أفقيًا */
  align-items: center; /* التوسيط رأسيًا */
}
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

بخصوص دعم المتصفحات:

- كل المتصفحات الرئيسية تدعم حاوية flex، عدا الإصدارات التي تسبق الإصدار العاشر من متصفح IE.
- بعض إصدارات المتصفحات، مثل Safari 8 و IES10، تتطلب استخدام البوادئ الخاصة بها vendor prefixes.
- لتوليد البوادئ بشكل تلقائي يمكن استعمال أداة Autoprefixer.

- يمكن تعويض نقص دعم المتصفحات القديمة لحاوية flex بإضافة شيفرات خاصة.
- لمزيد من التفاصيل حول دعم المتصفحات لحاوية flex، انظر هذه الإجابة.

زُر التوثيق الرسمي لحاوية flex لمزيد من المعلومات حول التوسيط الأفقي والرأسي، ومزيد من المعلومات حول حاوية flex ومعاني الأنماط المختلفة.

2.6.2 توسيط العناصر باستخدام التحويلات CSS Transform

تعتمد تحويلات CSS على أحجام العناصر. إذا كنت لا تعرف طول أو عرض العنصر، اتَّبع الخطوات التالية لتوسيطه أفقيًا ورأسيًا:

- طبِّق القاعدة `position: absolute` على العنصر المراد توسيطه.
- حدد القيمة `50%` للخاصيتين `top` و `left`.
- طبِّق القاعدة `transform: translate(-50%, -50%)`.

خذ في الحسبان أن استعمال هذه الطريقة قد يؤدي إلى ظهور العنصر المراد توسيطه مشوشًا، وذلك بسبب عرضه على بكسلات ذات قيم غير حقيقية (تحتوي على أرقام كسرية مثل 5.6). انظر هذه الاجابة لمعرفة كيفية تفادي هذه المشكلة.

- ملف HTML:

```
<div class="container">
  <div class="element"></div>
</div>
```

- ملف CSS:

```
.container {
  position: relative;
}
.element {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

عرض المثال على JSFiddle.

بخصوص دعم المتصفحات، تتطلب خاصية `transform` استخدام البوادئ لكي تُدعم في المتصفحات القديمة. هذه البوادئ لازمة للمتصفحات الآتية:

- Chrome<=35
- Safari<=8
- Opera<=22
- Android Browser<=4.4.4
- تحويلات CSS غير مدعومة في متصفح IE8 والإصدارات السابقة له.

بعض التحويلات الشائعة للمثال السابق:

```
  -webkit-transform: translate(-50%, -50%); /* Chrome, Safari, Opera,
Android */
      -ms-transform: translate(-50%, -50%); /* IE 9 */
          transform: translate(-50%, -50%);
```

لمزيد من المعلومات حول الدعم، قم بزيارة caniuse.

معلومات إضافية:

- يُحدد موضع العنصر وفقًا لأول أب غير ثابت، أي قيمة الخاصية `position` له هي `relative` أو `absolute` أو `fixed` استكشف المزيد على fiddle.
- للتوسيط أفقيًا فقط، استعمل `left: 50%` و `transform: translateX(-50%)`. كذلك للتوسيط رأسيًا فقط استعمل `top: 50%` و `transform: translateY(-50%)`.
- استعمال عناصر ذات عرض أو ارتفاع غير ثابت مع طريقة التوسيط هذه قد يجعل العنصر المراد توسيطه يبدو مضغوطًا. غالبًا ما يحدث هذا مع العناصر التي تحتوي على نصوص، ويمكن معالجتها بإضافة القواعد: `margin-right: -50%;` و `margin-bottom: -50%`. لمزيد من المعلومات اطّلع على المثال التالي.

> إذا كانت الصفحة باللغة العربية أو إحدى اللغات الأخرى التي تُكتب من اليمين إلى اليسار rtl تستبدل القاعدة `left: 50%` بالقاعدة `right: 50%`.

2.6.3 التوسيط باستخدام الخاصية margin

يمكن توسيط العناصر باستخدام `margin: 0 auto;` إذا كانت عناصر كتليه ولها عرض محدد.

- ملف HTML

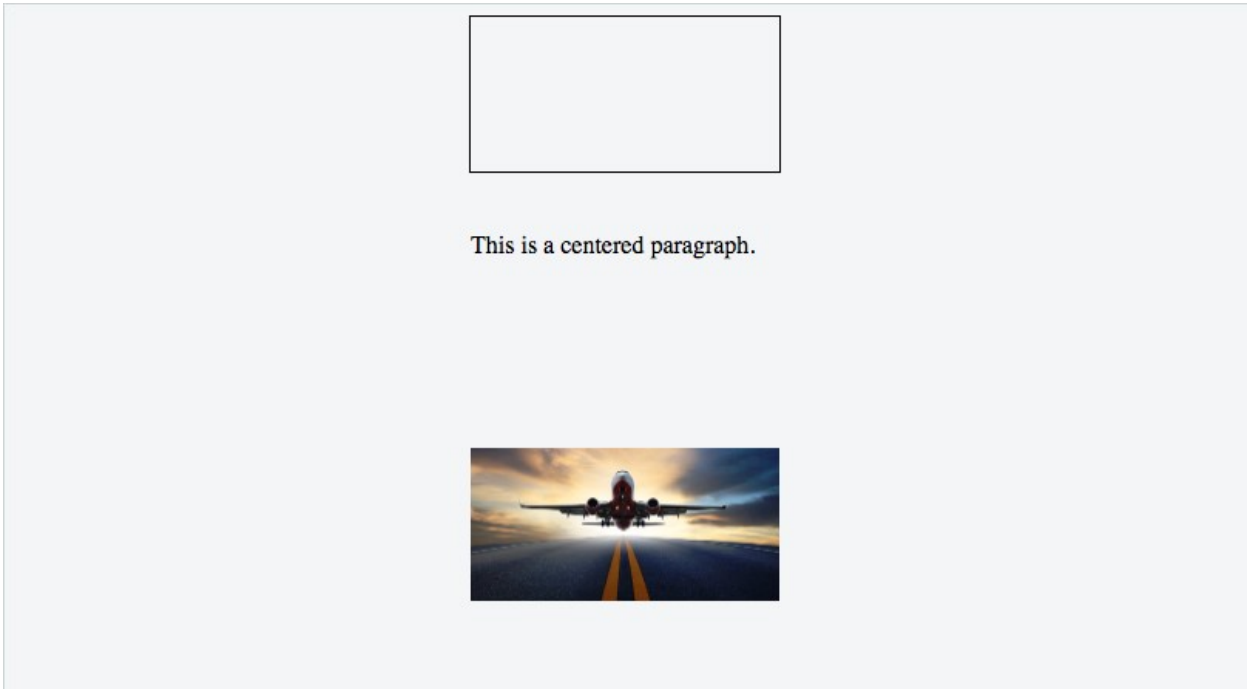
```
  <div class="containerDiv">
    <div id="centeredDiv"></div>
  </div>

  <div class="containerDiv">
    <p id="centeredParagraph">This is a centered paragraph</p>
  </div>

  <div class="containerDiv">
    <img
      id="centeredImage"
      src="https://i.kinja-img.com/gawker-media/image/upload/s--
c7Q9b4Eh--/c_scale,fl_progressive,q_80,w_
  800/qqyvc3bkpyl3mfhr8all.jpg"
    />
  </div>
```

- ملف CSS:

```
  .container {
    width: 100%;
    height: 100px;
    padding-bottom: 40px;
  }

  #centeredDiv {
    margin: 0 auto;
    width: 200px;
    height: 100px;
    border: 1px solid #000;
  }

  #centeredParagraph {
    width: 200px;
    margin: 0 auto;
  }
```

```
#centeredImage {
  display: block;
  width: 200px;
  margin: 0 auto;
}
```

النتيجة:



اطّلع على تجربة حيّة لهذا المثال على JSFiddle.

2.6.4 التوسيط باستخدام محاذاة النص text-align

من أسهل أنواع التوسيط وأكثرها انتشارًا توسيط النصوص داخل عنصر معين باستخدام القاعدة:

```
text-align: center;
```

- ملف HTML:

```
<p>lorem ipsum</p>
```

- ملف CSS:

```
p {
  text-align: center;
}
```

تستعمل `text-align` لمحاذاة المحتوى السطري (inline-content) فقط مثل محاذاة النصوص والصور داخل العنصر الأب لها، ولا يمكن استعمالها لتوسيط العناصر الكتلية.

2.6.5 استخدام الموضع المطلق position: absolute

يمكن توسيط عنصر ذي موضع مطلق داخل العنصر الأب له باستعمال الهوامش التلقائية مع وضع قيمة صفر لكل من الخواص `left`، و`right` و`top` و `bottom`.

- ملف HTML

```
<div class="parent">
  <img class="center" src="http://lorempixel.com/400/200" />
</div>
```

- ملف CSS:

```
.parent {
  position: relative;
  height: 500px;
}

.center {
  position: absolute;
  margin: auto;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
}
```

تحتاج العناصر التي لا تمتلك ارتفاعًا وعرضًا ضمنيين مثل الصور إلى تحديد قيمتهما.

يمكن أيضًا الاطلاع على المثال العملي التالي: Absolute Centering in CSS.

2.6.6 التوسيط باستخدام الدالة calc

دالة `calc()` هي صيغة جديدة تمت إضافتها في CSS3. تمكنك هذه الدالة من أن تحسب رياضيًا الحجم أو الموضع الذي يشغله العنصر باستخدام عدد من القيم مثل: البكسلات (نقاط الشاشة)، النسب المئوية ...الخ.

انتبه، عند استخدام هذه الدالة، يجب مراعاة المسافة بين القيمتين `calc(100% - 80px)` .

- ملف HTML

```
<div class="center"></div>
```

- ملف CSS

```
.center {
  position: absolute;
  height: 50px;
  width: 50px;
  background: red;
  top: calc(50% - 50px / 2); /* الإرتفاع مقسوم على 2 */
  left: calc(50% - 50px / 2); /* العرض مقسوم على 2 */
}
```

2.6.7 توسيط النص رأسيًا باستخدام ارتفاع السطر line-height

يمكن استخدام خاصية ارتفاع السطر `line-height` لتوسيط نص من سطر واحد رأسيًا داخل الحاوية.

```
div {
  height: 200px;
  line-height: 200px;
}
```

قد لا يبدو ذلك أنيقًا، ولكن قد يكون مفيد إذا أردت توسيط النص رأسيًا داخل عنصر `<input />`. خاصية `line-height` تعمل فقط إذا كان النص المراد توسيطه من سطر واحد، ولكن إذا كان أكثر من سطر لن يكـــون الناتج في الوسط.

2.7 محاذاة أي شيء رأسيًا في ثلاثة أسطر

يمكنك باستعمال الثلاثة أسطر أدناه توسيط أي شيء رأسيًا، فقط تأكد أن الحاوية div أو الصورة التي تطبق عليها الشيفرة لها أب parent ذو ارتفاع محدد.

- ملف CSS:

```
div.vertical {
  position: relative;
  top: 50%;
  transform: translateY(-50%);
```

```
}
```

- ملف HTML:

```
<div class="vertical">Vertical aligned text!</div>
```

عرض النتيجة.

2.7.1 التوسيط نسبةً لعنصر آخر

ستتعلم في هذا القسم كيفية توسيط المحتوى استنادًا إلى ارتفاع عنصر مجاور.

- ملف HTML:

```
<div class="content">
  <div class="position-container">
    <div class="thumb">
      <img src="http://lorempixel.com/400/200" />
    </div>
    <div class="details">
      <p class="banner-title">text</p>
      <p class="banner-text">content content content content</p>
      <button class="btn">button</button>
    </div>
  </div>
</div>
```

- ملف CSS:

```
.content * {
  box-sizing: border-box;
}

.content .position-container {
  display: table;
}

.content .details {
  display: table-cell;
```

```
    vertical-align: middle;
    width: 33.33333%;
    padding: 30px;
    font-size: 17px;
    text-align: center;
}

.content .thumb {
    width: 100%;
}

.content .thumb img {
    width: 100%;
}
```

اطّلع على تجربة حيّة لهذا المثال على JSFiddle.

النقاط الرئيسية هي الحاويات الثلاث `.thumb` و `.details` و `.position-container`:

- يجب أن تمتلك الحاوية `.position-container` القاعدة `display: table`.
- يجب أن تمتلك الحاوية `.details` الخاصيات:
  - `width`
  - `display: table-cell`
  - `vertical-align: middle`
- يجب أن تمتلك الحاوية `.thumb` القاعدة `width: 100%` إذا كنت تريدها أن تأخذ كل المساحة المتبقية، وستكون محددة بعرض حاوية `.details`.
- يجب أن تكون الصورة (في حال كان لديك صورة) داخل الحاوية `.thumb` ويجب أن تمتلك القاعدة `width: 100%`، لكنها غير ضرورية إذا كان لديك تناسب صحيح.

2.7.2 طريقة العنصر الشبح: خدعة Michat Czernow

يمكن استعمال هذه الطريقة حتى إذا كانت أبعاد الحاوية مجهولة.

قم بإنشاء عنصر "شبح" بارتفاع 100% داخل الحاوية التي تريد توسيطها، ومن ثم طبق القاعدة `vertical-align: middle` عليه وعلى الحاوية المراد توسيطها.

- ملف CSS:

```
/* يُمكن أن يأخذ العنصر الأب أي ارتفاع وعرض */
.block {
  text-align: center;

  /* انظر الملاحظة رقم 1 */
  white-space: nowrap;
}

/* العنصر الشبح */
.block:before {
  content: "";
  display: inline-block;
  height: 100%;
  vertical-align: middle;

  /* انظر الملاحظة رقم 2 */
  margin-right: -0.25rem;
}

/* العنصر المراد توسيطه، يمكن أن يكون بأي عرض وارتفاع */
.centered {
  display: inline-block;
  vertical-align: middle;
  width: 300px;
  white-space: normal;
}
```

ملاحظات:

1. يكون عرض الحاوية في بعض الأحيان أقل من عرض العناصر الداخلية لها مما يؤدي إلى قص أجزاء منها، ويمكنك إضافة القاعدة `white-space: nowrap` لتفادي هذا الأمر.
2. يتسبب حرف المسافة الخاص بالعنصر الشبح في وجود فراغ بينه وبين الحاوية `.centered` للتخلص من هذا الفراغ يمكن دفع الحاوية `.centered` إلى اليمين (أو إلى اليسار في حالة الصفحات العربية) قليلًا (تعتمد المسافة التي يجب دفع الحاوية بها على نوع الخط المستخدم في الصفحة) أو عن طريق

تحديـد قيمـة صـفر لخاصـية `font-size` في الحاويـة `.parent` ومن ثم إعـادة تعيينـه في الحاويـة `.centered`.

مثال:

```
<div class="block">
  <div class="centered"></div>
</div>
```

2.7.3 التوسيط أفقيًا ورأسيًا دون القلق بشأن ارتفاع وعرض العنصر

الطريقة التالية تمكنك من إضافة المحتوى إلى عنصـر HTML و توسـيطه أفقيًـا ورأسـيًا دون القلـق بشـأن ارتفاعه أو عرضه.

الحاويـة الخارجيـة يجب أن تمتلـك القاعـدة `display: table` أمـا الحاويـة الداخليـة يجب أن تمتلك القواعد التالية:

```
display: table-cell;
vertical-align: middle;
text-align: center;
```

صندوق المحتوى:

- يجب أن يمتلك القاعدة `display: inline-block`.
- يجب إعادة ضبط محـاذاة النص، مثلاً `text-align-left` أو `text-align: right`، إلا إذا كنت ترغب في أن يكون النص في الوسط.

إليك المثال التالي:

- ملف HTML:

```
<div class="outer-container">
  <div class="inner-container">
    <div class="centered-content">
      You can put anything here!
    </div>
  </div>
</div>
```

- ملف CSS

```
body {
  margin: 0;
}

.outer-container {
  position: absolute;
  display: table;
  width: 100%; /* This could be ANY width */
  height: 100%; /* This could be ANY height */
  background: #ccc;
}

.inner-container {
  display: table-cell;
  vertical-align: middle;
  text-align: center;
}

.centered-content {
  display: inline-block;
  text-align: left;
  background: #fff;
  padding: 20px;
  border: 1px solid #000;
}
```

اطّلع على تجربة حيّة لهذا المثال على JSFiddle.

2.7.4 محاذاة صورة رأسيًا داخل حاوية div

انظر المثال التالي:

- ملف HTML:

```
<div class="wrap">
  <img src="http://lorempixel.com/400/200" />
</div>
```

- ملف CSS:

```
.wrap {
  height: 50px;
  width: 100px;
  border: 1px solid blue;
  text-align: center;
}

.wrap:before {
  content: "";
  display: inline-block;
  height: 100%;
  vertical-align: middle;
  width: 1px;
}

img {
  vertical-align: middle;
}
```

2.7.5 التوسيط مع حجم ثابت

إذا كان حجم المحتوى ثابتًا، يمكن تعيين قيمة الموضع المطلق بالقيمة 50% مع هوامش تُقلص نصف عرض وارتفاع المحتوى.

- ملف HTML

```
<div class="center">
  Center vertically and horizontally
</div>
```

- ملف CSS:

```
.center {
  position: absolute;
  background: #ccc;

  left: 50;
  width: 150px;
```

```
  margin-left: -75px; /* width * -0.5 */

  top: 50%;
  height: 200px;
  margin-top: -100px; /* height * -0.5 */
}
```

ا. التوسيط أفقيًا مع عرض ثابت

يمكنك توسيط العنصر أفقيًا حتى إذا لم تكن تعرف ارتفاع محتواه.

- ملف HTML:

```
<div class="container">
  Center only horizontally
</div>
```

- ملف CSS:

```
.center {
  position: absolute;
  background: #ccc;

  left: 50%;
  width: 150px;
  margin-left: -75px; /* width * -0.5 */
}
```

ب. التوسيط رأسيًا مع ارتفاع ثابت

يمكنك توسيط العنصر رأسيًا إذا كنت تعرف فقط ارتفاعه.

- ملف HTML:

```
<div class="center">
  Center only vertically
</div>
```

- ملف CSS:

```
.center {
  position: absolute;
  background: #ccc;

  top: 50%;
  height: 200px;
  margin-top: -100px; /* width * -0.5 */
}
```

2.7.6 المحاذاة الرأسية للعناصر ذات الارتفاع الديناميكي

تطبيق CSS بشكل بديهي لا يعطي النتائج المطلوبة وذلك للأسباب الآتية:

- قاعدة `vertical-align: middle` لا تنطبق على العناصر الكتلية.
- قاعدة `margin-top: auto` و `margin-bottom: auto` تُحسب القيم المستخدمة لها كأصفار.
- قاعدة `margin-top: -50%` يتم حساب قيمة الهامش على أساس النسبة المئوية بالنسبة لعرض الكتلة التي تحويها.

للحصول على دعم أغلب المتصفحات، هناك حل بديل باستخدام العناصر المساعدة:

- ملف HTML

```
<div class="vcenter--container">
  <div class="vcenter--helper">
    <div class="vcenter--content">
      <!-- stuff -->
    </div>
  </div>
</div>
```

- ملف CSS:

```
.vcenter--container {
  display: table;
  height: 100%;
  position: absolute;
  overflow: hidden;
  width: 100%;
```

```
}

.vcenter--helper {
  display: table-cell;
  vertical-align: middle;
}

.vcenter--content {
  margin: 0 auto;
  width: 200px;
}
```

المصدر: Stackoverflow وتجد المثال على JSFiddle.

ملاحظات:

- تعمل هذه الطريقة مع العناصر ذات الارتفاعات الديناميكية.
- تأخذ في الاعتبار تدفق المحتوى .
- مدعومة من قبل معظم المتصفحات.

2.7.7 التوسيط أفقيًا ورأسيًا باستخدام تخطيط الجدول

يمكن ببساطة توسيط العنصر الابن باستخدام القاعدة `display: table`.

- ملف HTML:

```
<div class="wrapper">
  <div class="parent">
    <div class="child"></div>
  </div>
</div>
```

- ملف CSS:

```
.wrapper {
  display: table;
  vertical-align: center;
  width: 200px;
  height: 200px;
```

```
    background-color: #9e9e9e;
}

.parent {
    display: table-cell;
    vertical-align: middle;
    text-align: center;
}

.child {
    display: inline-block;
    vertical-align: middle;
    text-align: center;
    width: 100px;
    height: 100px;
    background-color: teal;
}
```

انتبه، بالرغم من بساطة وسهولة هذه الطريقة إلا أنها لا تُعد ممارسة جيدة و يجب ألَّا تستعملها إلا إذا اضطررت لذلك، لمزيد من التفاصيل راجع هذه الاجابة على StackOverflow.

2.7.8 التوسيط باستخدام العناصر الزائفة

انظر المثال التالي:

- ملف HTML

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

- ملف CSS

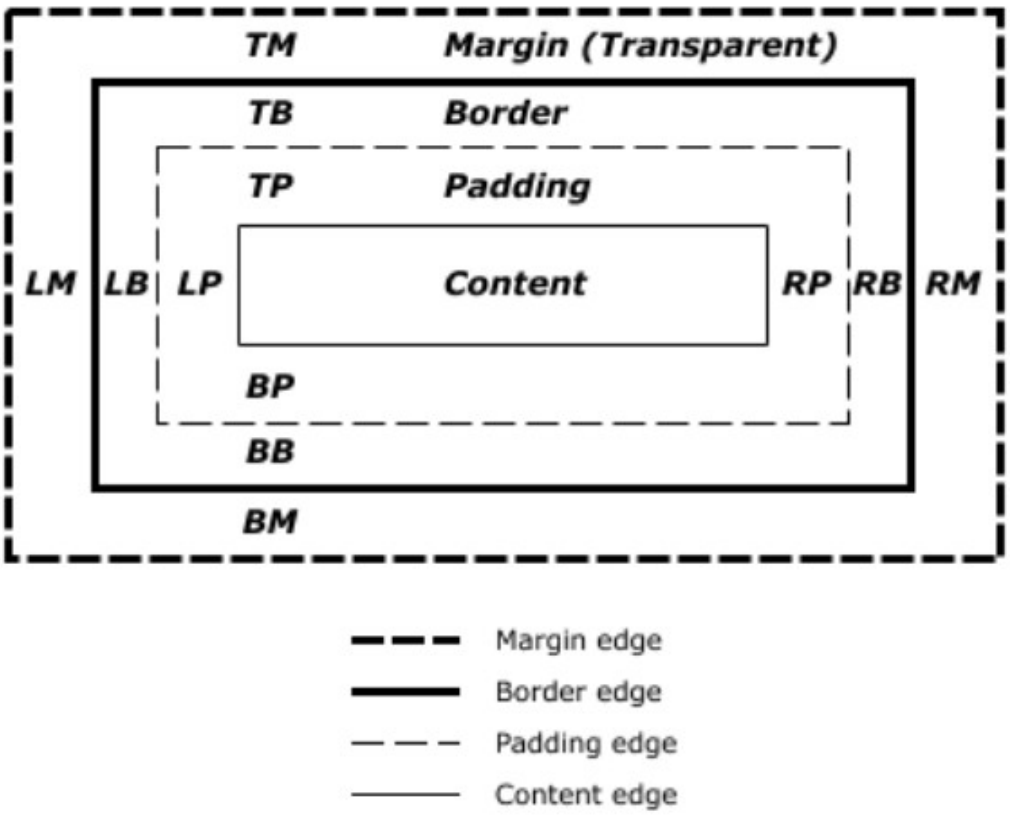
```
.wrapper{
    min-height: 600px;
}
.wrapper:before{
    content: "";
    display: inline-block;
```

```
        height: 100%;
        vertical-align: middle;
    }
    .content {
        display: inline-block;
        height: 80px;
        vertical-align: middle;
    }
```

3. النموذج الصندوقي Box Model

3.1 ما هو النموذج الصندوقي؟

ترسم المتصفحات مستطيلًا حول كل عنصر في صفحة HTML ويصف النموذج الصـندوقي كيـف تُضـاف الحواشي والإطارات والهوامش للمحتوى لرسم هذا المستطيل.



الرسم التخطيطي من CSS2.2 Working Draft.

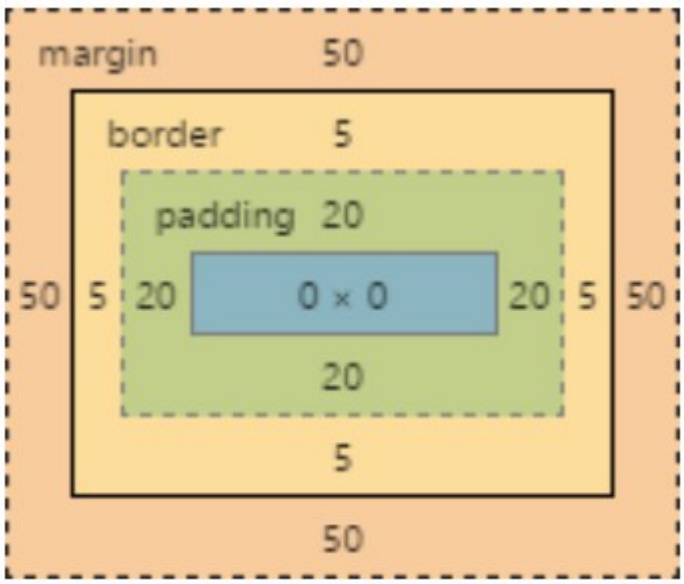
يسمى محيط كل مساحة من المساحات الأربع مُحدِّد، كل مُحدِّد يحدد صندوقًا معينًا.

- يُسمى المستطيل الداخلي صندوق المحتوى content-box، يعتمد عرض وارتفاع هـذا الصـندوق على المحتوى المعروض للعنصر (النصوص، الصور وأي أبناء آخرين للعنصر).
- المستطيل التالي هو صندوق الحواشـي padding-box، ويُعـرَّف بخاصـية الحاشـية padding. إذا لم يتم تحديد عرض الحاشية، تكون حافة الحاشية مطابقة لحافة صندوق المحتوى.
- يلي ذلك صندوق الإطارات، ويُعرَّف بخاصـية الإطـار border. إذا لم يُحـدد عـرض الإطـار، تكـون حافـة صندوق الإطارات مطابقة لحافة صندوق الحواشي.
- المستطيل الخارجي هو صندوق الهوامش، ويُعرَّف بخاصية الهامش margin. إذا لم يتم تحديـد عـرض الهامش، تكون حافة صندوق الهوامش مطابقة لحافة صندوق الإطارات.

مثال:

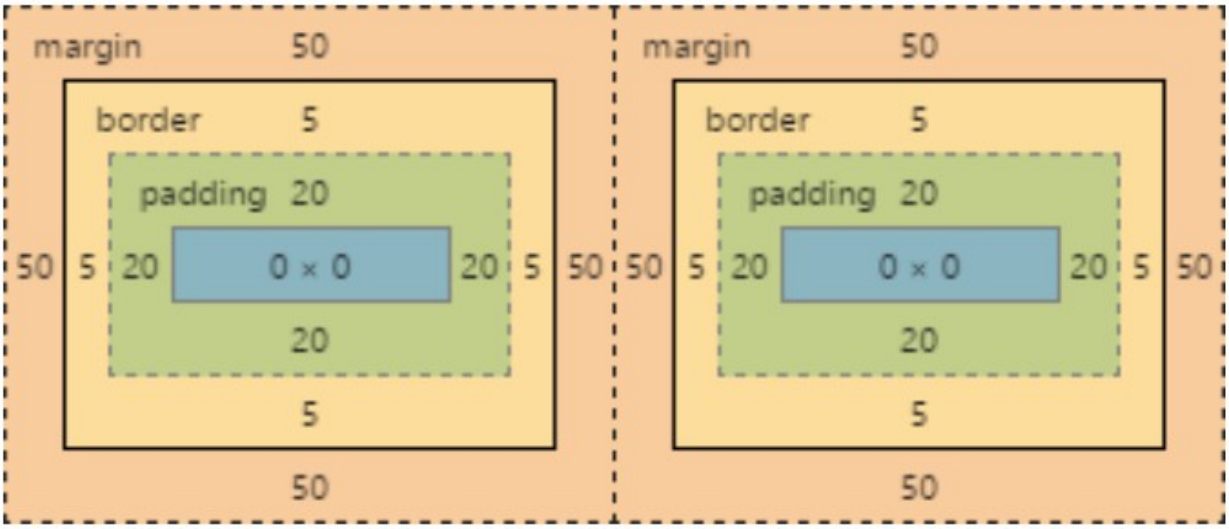
```
div {
    border: 5px solid red;
    margin: 50px;
    padding: 20px;
}
```

هذه الشيفرة تنشئ إطارًا بعـرض 5 بكسـلات لحـواف العنصـر الأربـع لكـل عناصـر div في صـفحة HTML، وكذلك تضيف هوامش بعرض 50 بكسل و حواشي بعرض 20 بكسل في الاتجاهات الأربعة. بتجاهل المحتـوى، يكون النموذج الصندوقي الناتج بهذا الشكل:



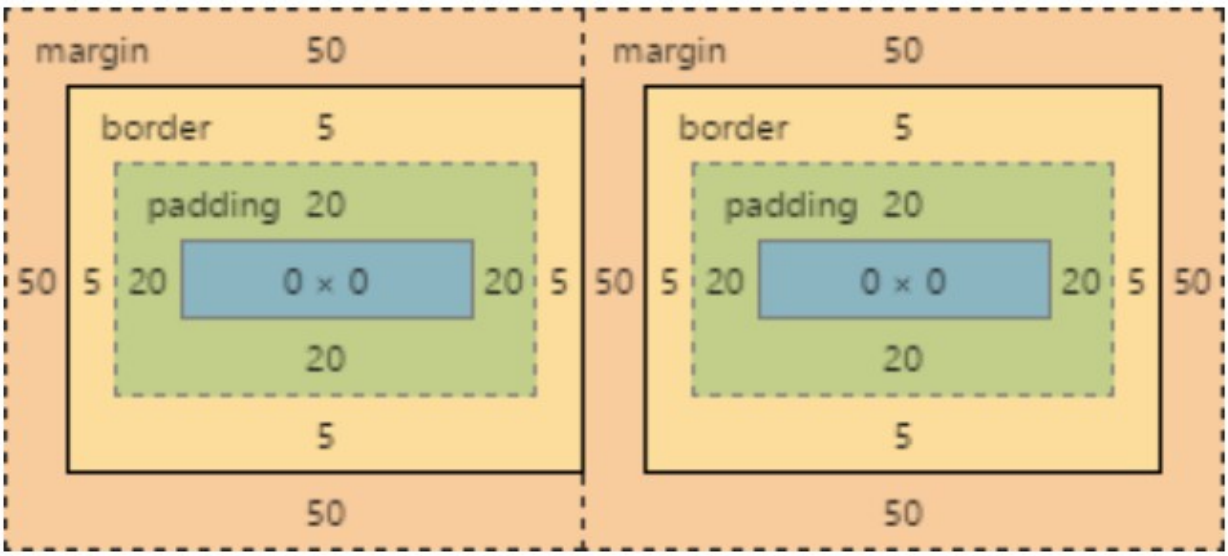
- بما أنه لا يوجد محتوى، سيكون صندوق المحتوى (الصندوق الأزرق في الوسط) بدون عـرض أو ارتفـاع (0 بكسل × 0 بكسل).
- بشكل ابتدائي يكون صندوق الحواشي بنفس مساحة صندوق المحتوى، بالإضافة إلى عـرض الحواشـي 20 بكسل في الحواف الأربع. وتكون مساحة الصندوق (40 بكسل × 40 بكسل).
- صندوق الإطارات له نفس مسـاحة صـندوق الحواشـي، بالإضـافة إلى 5 بكسـلات تمثـل عـرض الإطـار للحواف الأربع. وتكون مساحة الصندوق الكلية (50 بكسل × 50 بكسل).
- أخيرًا، صندوق الهـوامش لـه نفس مسـاحة صـندوق الحواشـي بالإضـافة إلى 50 بكسـل تمثـل عـرض الهوامش في الحواف الأربع. وتكون مساحة صندوق العنصر الكلية (150 بكسل × 150 بكسل).

إذا أضفنا أخًا للعنصر الحالي بنفس الأنماط فسينظر المتصفح للنمـوذج الصـندوقي للعنصـرين ليحـدد أين يوضع العنصر الجديد بالنسبة لمحتوى العنصر السابق.



يفصل بين محتوى العنصرين مسافة 150 بكسل، لكن النماذج الصندوقية لهما متلامسة.

إذا عدلنا العنصر الأول وحذفنا الهامش الأيمن، فسيكون حد صندوق الهوامش الأيمن مطابق لحد صـندوق الإطارات الأيمن. وسيظهر العنصرين بالشكل التالي:



3.2 تعديل النموذج الصندوقي باستخدام الخاصية box-sizing

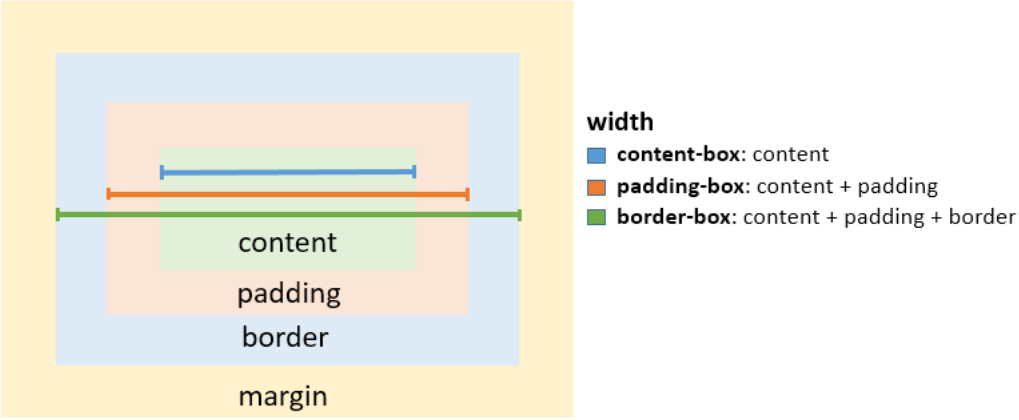
قد يسلك النموذج الصندوقي الأساسي (أي `content-box`) سلوكًا غير متوقع، فعندما تبدأ بإضافة حواشي أو إطارات للعنصر يصبح الارتفاع والعرض المحددين لا يمثلان الارتفاع والعرض الحقيقيين للعنصر الظاهر على الشاشة. ويوضح المثال التالي هذه المشكلة:

```
textarea {
    width: 100%;
    padding: 3px;
    box-sizing: content-box; /* default value */
}
```

بما أن الحواشي ستضاف لعرض العنصر `textarea`، يصبح عرض العنصر الناتج أكبر من 100%.

لحسن الحظ، تمكننا CSS من تعديل النموذج الصندوقي للعنصر باستخدام خاصية `box-sizing` التي يمكن أن تأخذ إحدى القيم التالية لضبط النموذج الصندوقي للعنصر:

المعامل	الوصف
content-box	عرض وارتفاع العنصر يتضمن فقط مساحة المحتوى دون الحاشية والإطار.
padding-box	عرض وارتفاع العنصر يتضمن مساحة المحتوى والحاشية فقط باستثناء الإطار
border-box	عرض وارتفاع العنصر، يتضمن مساحة المحتوى والحاشية والإطار.
initial	تُطبِّق القيمة الابتدائية للنموذج الصندوقي
inherit	تُوَرِّث النموذج الصندوقي للعنصر الأب.



لحـل مشـكلة عنصـر `textarea` في المثـال السـابق، يمكن تغيـير قيمـة الخاصـية `box-sizing` إلى `padding-box` أو `border-box` والأخيرة هي الأكثر انتشارًا.

```
textarea {
    width: 100%;
    padding: 3px;
    box-sizing: border-box;
}
```

لتطبيق نموذج صندوقي معين لكل العناصر في صفحة HTML يمكن استعمال الشيفرة التالية:

```
html {
    box-sizing: border-box;
}

*, *:before, *:after {
    box-sizing: inherit;
}
```

في هـذه الشـيفرة، لا تطبـق الخاصـية `box-sizing: border-box;` مباشـرة على كـل العناصـر (*)، بالتالي يمكنك بسهولة استبدالها في العناصر كلٌّ على حدة.

3.3 الهوامش Margins

يوضح الجدول التالي القيم التي يمكن استعمالها مع خاصية الهامش `margin`:

المعامل	الوصف
0	يحذف جميع الهوامش الخاصة بالعنصر.
auto	يستعمل للتوسيط بإضافة هوامش متساوية على الجانبين.
units	انظر قسم الوحدات لمعرفة الوحدات المتاحة.
inherit	تُوَرِّث قيمة الهامش من العنصر الأب.
initial	تُطبِّق القيمة الابتدائية للهامش.

3.3.1 تداخل الهوامش Margin Collapsing

عند تلامس هامشين رأسيًا فإنهما يتداخلان ويصبحان هامشًا واحدًا بارتفاع يساوي ارتفـاع الهـامش الأكـبر، ولكن عندما يتلامسان أفقيًا لا يحدث تداخل ويكون عرض الهامش الناتج مساويًا لمجموع عرض الهامشين.

مثال لتداخل الهوامش الرأسية:

- ملف CSS:

```
div {
    margin: 10px;
}
```

- ملف HTML:

```
<div>
    some content
</div>
<div>
    some more content
</div>
```

وتكون المسافة الرأسية بين العنصرين 10 بكسلات بسبب تداخل الهوامش.

مثال لتداخل الهوامش الأفقية:

- ملف CSS:

```
span {
    margin: 10px;
}
```

- ملف HTML:

```
    <span>some</span><span>content</span>
```

وتكون المسافة الأفقية بين العنصرين 20 بكسل، أي أنها تساوي مجموع الهوامش الأفقية للعنصرين.

مثال تداخل الهوامش ذات الأبعاد المختلفة:

- ملف CSS:

```
    .top {
        margin: 10px;
    }
    .bottom {
        margin: 15px;
    }
```

- ملف HTML:

```
    <div class="top">
        some content
    </div>
    <div class="bottom">
        some more content
    </div>
```

وتكون المسافة الرأسية بين العنصرين 15 بكسل، وذلك لأن الهوامش المتداخلة تأخذ أبعاد الهامش الأكبر.

- ملف CSS:

```
    .outer-top {
        margin: 10px;
    }
    .inner-top {
        margin: 15px;
    }
    .outer-bottom {
        margin: 20px;
    }
    .inner-bottom {
        margin: 25px;
    }
```

- ملف HTML:

```
<div class="outer-top">
    <div class="inner-top">
        some content
    </div>
</div>
<div class="outer-bottom">
    <div class="inner-bottom">
        some more content
    </div>
</div>
```

بما أنَّ جميع الهوامش متداخلة، تكون المسافة بين النصين 25 بكسل.

ماذا يحدث في حال إضافة إطارات حول العناصر؟

```
div {
    border: 1px solid red;
}
```

عند إضافة إطارات حول العناصر كما في المثال السابق، تصبح المسافة بين النصين 59 بكسل وهي مجموع مساحات الهوامش بالإضافة لأربعة بكسلات تمثل مجموع الإطارات حول الحاويات الأربعة. لا يحدث التداخل في هذه الحالة لأن الهوامش أصبحت غير متلامسة إذ يفصل الإطار بينها.

انظر مثال عن تداخل هوامش العنصر الأب مع هوامش ابنه:

- ملف HTML:

```
<h1>Title</h1>
<div>
    <p>Paragraph</p>
</div>
```

- ملف CSS:

```
h1 {
    margin: 0;
    background-color: #cff;
}
```

```
div {
    margin: 50px 0 0 0;
    background: #cfc;
}
p {
    margin: 25px 0 0 0;
    background: #cf9;
}
```

في المثال أعلاه، تكون المسافة الرأسية بين عنصر العنوان الرئيسي h1 وعنصر الفقرات p هي 50 بكسل (مساحة الهامش الأكبر)، قد تتوقع أن تكون المسافة بينهما 75 بكسل (لأن الحاوية div لديها هامش علوي بارتفاع 50 بكسل وعنصر الفقرات p لديه هامش علوي بارتفاع 25 بكسل)، لكن لا يحدث هذا بسبب تداخل الهوامش.

3.3.2 إضافة هامش باتجاه محدد

يمكن إضافة هامش باتجاه محدد باستعمال إحدى الخواص الأربع التالية:

```
margin-left
margin-right
margin-top
margin-bottom
```

الشيفرة أدناه تُضيف هامشًا أيسر بعرض 30 بكسل للحاوية #myDiv.

- ملف HTML:

```
<div id="myDiv"></div>
```

- ملف CSS:

```
#myDiv {
    margin-left: 30px;
    height: 40px;
    width: 40px;
    background-color: red;
}
```

حيث margin-left اتجاه إضافة الهامش والقيمة 30px هي عرض الهامش.

توضح الشيفرة التالية كيفية استخدام الصياغة المختزلة للخاصية margin:

```
margin: <top> <right> <bottom> <left>;
```

الشيفرة التالية تُضيف هامشًا للحافة العليا بارتفاع صفر بكسل (أي لا يوجد هامش علوي)، وهامشًا للحافة اليُمنى بعرض 10 بكسلات، وهامشًا للحافة السُفلي بارتفاع 50 بكسل، وهامشًا للحافة اليُسرى بعرض 100 بكسل.

- ملف HTML:

```
<div id="myDiv"></div>
```

- ملف CSS:

```
#myDiv {
    margin: 0 10px 50px 100px;
    height: 40px;
    width: 40px;
    background-color: red;
}
```

3.3.3 تبسيط خاصية الهوامش margin

انظر المثال التالي:

```
p {
    margin: 1px;
    margin: 1px 1px;
    margin: 1px 1px 1px;
    margin: 1px 1px 1px 1px;

    /* جميع القواعد أعلاه تؤدي لنفس النتيجة */
}
```

مثال آخر:

```
p {
    margin: 10px 15px;
    margin: 10px 15px 10px 15px;
    margin 10px 15px 10px;
```

```
    /* جميع القواعد أعلاه تؤدي لنفس النتيجة */
}
```

توضح الشيفرة التالية الصيغ العامة لاستعمال خاصية الهامش margin:

```
margin: <all>;
margin: <top and bottom> <left and right>;
margin: <top> <left and right> <bottom>;
margin: <top> <right> <bottom> <left>;
```

3.3.4 توسيط العناصر أفقيًا باستخدام خاصية الهامش margin

تُستخدم القاعدة `margin: 0 auto` لتوسيط العناصر الكُتلية أفقيًا عن طريق إضافة هـوامش متسـاوية يمين ويسار العنصر، ويُشترط أن يكون عرض العنصر المُراد توسيطه أقل من عرض الحاوية.

```
#myDiv {
    width: 80%;
    margin: 0 auto;
}
```

استخدمنا في المثال أعلاه الصياغة المختزلة لكتابة الخاصية `margin` لإضافة هـوامش بعـرض صـفر أعلى وأسفل العنصر (يمكن أن تكون أي قيمة أخرى) ومن ثم استعملنا القيمة `auto` لإضافة هوامش متسـاوية يمين ويسار العنصر وبالتالي توسيطه أفقيًا. عرض العنصر `#myDiv` هـو 80% من عـرض الحاويـة الخارجيـة لـه، ممـا يترك 20% من مساحة الحاوية فارغة. عند تحديد القيمة `auto` لخاصية `margin` يُوَزِّع المتصفح هذه المسـاحة بالتساوي على يمين ويسار العنصر، أي تكون المسافة على كل جانب 10%.

مثال، من الشائع افتراض أن تحديـد قيم نسـبية للهـوامش يكـون نسـبةً إلى العنصـر الأب. هـذا الافـتراض صـحيح بالنسـبة للهـوامش الأفقيـة `margin-left` و `margin-right`. حيث يكـون عـرض الهـامش نسـبيًا لعرض العنصر الأب.

```
.parent {
    width: 500px;
    height: 300px;
}

.child {
    width: 100px;
```

```
    height: 100p;
    margin-left: 10%; /* parentWidth * 10/100 => 50px */
}
```

لكن الأمـر يختلـف عنـد الحـديث عن الهـوامش الرأسـية `margin-bottom` و `margin-top`، فارتفـاع الهامش الرأسي يكون نسبيًا من **عرض** العنصر الأب وليس من ارتفاعه كما يظن البعض.

```
.parent {
    width: 500px;
    height: 300px;
}

.child {
    width: 100px;
    height: 100px;
    margin-left: 10%;
    margin-top: 20%;
}
```

3.3.5 الهوامش السالبة

يمكن استعمال الهوامش السالبة لإنشاء عناصر متداخلة دون الحاجة لاستعمال الموضع المطلق.

- ملف CSS:

```
div {
    display: inline;
}

#over {
    margin-left: -20px;
}
```

- ملف HTML:

```
<div>Base div</div>
<div id="over">Overlapping div</div>
```

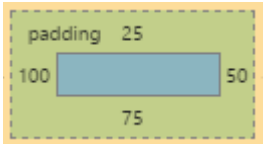
3.4 الحواشي Paddings

مساحة الحاشية هي المساحة بين محتوى العنصر والإطار الخارجي له، وتُحدد باستخدام خاصية padding.

يمكنك استعمال الصياغة المختزلة الموضحة في المثال أدناه من تعريف مساحات الحواشي للعنصر بدلًا عن تحديد مساحة الحاشية لكل حافة من حواف العنصر الأربع بخاصية منفردة (عن طريق استخدام padding-top، أو padding-left أو غيرهما).

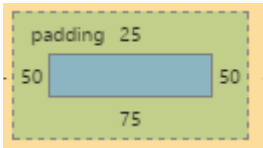
في حالة تحديد أربع قيم للخاصية:

```
<style>
    .myDiv {
        padding: 25px 50px 75px 100px; /* top right bottom left; */
    }
</style>
<div class="myDiv"></div>
```



في حالة تحديد ثلاث قيم للخاصية:

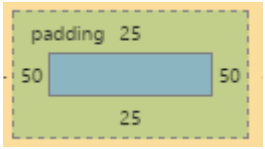
```
<style>
    .myDiv {
        padding: 25px 50px 75px; /* top left/right bottom; */
    }
</style>
<div class="myDiv"></div>
```



في حالة تحديد قيمتين للخاصية:

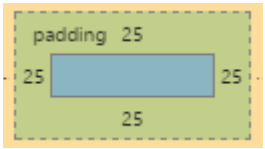
```
<style>
    .myDiv {
        padding: 25px 50px; /* top/bottom left/right; */
```

```
    }
</style>
<div class="myDiv"></div>
```



في حالة تحديد قيمة واحدة للخاصية:

```
<style>
    .myDiv {
        padding: 25px; /* top/right/bottom/left; */
    }
</style>
<div class="myDiv"></div>
```



3.4.1 إضافة حاشية باتجاه محدد

يمكنك إضافة الحاشية باتجاه محدد باستعمال الخواص أدناه:

```
padding-top /* حاشية علوية */
padding-right /* حاشية يُمنى */
padding-bottom /* حاشية يُسرى */
padding-left /* حاشية سُفلية */
```

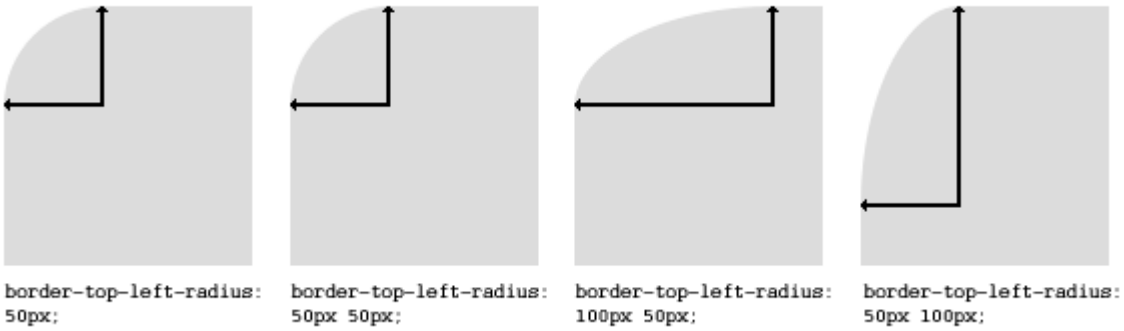
تُضيف الشيفرة أدناه حاشية علوية بعرض 5 بكسلات للعنصر.

```
<style>
    .myClass {
        padding-top: 5px;
    }
</style>

<div class="myClass"></div>
```

3.5 الإطارات Borders

3.5.1 خاصية الأركان المدورة border-radius

تمكنك خاصية border-radius من تغيير شكل النموذج الصندوقي فيمكن بضبطها تغيير زاوية كل ركن من أركان حواف العنصر.



تُعرِّف المجموعة الأولى من القيم الإنحاء الأفقي للإطار بينما تُعرِّف المجموعة الثانية الاختيارية (المسبوقة بعلامة /) الإنحاء الرأسي له. في حال وجود مجموعة واحدة فقط فإنها تُستخدم لتعريف الانحناءات الأفقية والرأسية.

```
border-radius: 10px 5% / 20px 25em 30px 35em;
```

تُحدِّد القيمة 10px الإنحاء الأفقي للإطار من أعلى اليسار وأسفل اليمين، بينما تحدِّد القيمة 5% الإنحاء الأفقي من أعلى اليمين وأسفل اليسار. تُحدِّد الأربع قيم المسبوقة بعلامة / الإنحاء الرأسي للإطار من أعلى اليسار، وأعلى اليمين، وأسفل اليمين وأسفل اليسار على الترتيب.

يُوضح المثال التالي كيفية استخدام الصياغة المختزلة لتدوير الإطارات:

- ملف HTML:

```
<div class='box'></div>
```

- ملف CSS:

```
.box{
    width:250px;
    height:250px;
    background-color:black;
    border-radius:10px;
}
```

تُحدِّد القيمة 10px انحناء الإطار في الاتجاهات الثمانية المذكورة في الفقرة السابقة.

من الاستخدامات الشائعة لخاصية `border-radius` استخدامها لتحويل صندوق العنصـر لشـكل دائـري عن طريق إعطائها قيمة مساوية لنصف ارتفاع الصندوق.

```
.circle{
    width:200px;
    height:200px;
    border-radius:100px;
}
```

من الشائع استخدام نسب مئوية لتحديد قيمة خاصية `border-raduis` لتجنُّب حسابها يدويًا.

```
.circle{
    width:150px;
    height:150px;
    border-radius:50%;
}
```

يصبح شكل صندوق العنصر بيضاويًا في حال اختلاف قيمتي الارتفاع والعرض.

إضافة البوادئ الخاصة بالمتصفحات القديمة إلى الخاصية `border-radius`:

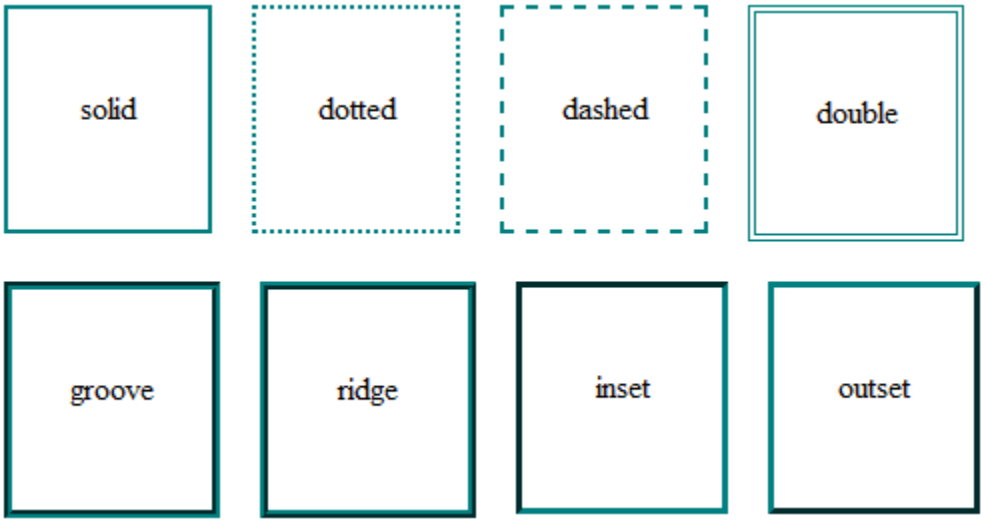
```
{
  -webkit-border-top-right-radius: 4px;
  -webkit-border-bottom-right-radius: 4px;
  -webkit-border-bottom-left-radius: 0;
  -webkit-border-top-left-radius: 0;
  -moz-border-radius-topright: 4px;
  -moz-border-radius-bottomright: 4px;
  -moz-border-radius-bottomleft: 0;
  -moz-border-radius-topleft: 0;
  border-top-right-radius: 4px;
  border-bottom-right-radius: 4px;
  border-bottom-left-radius: 0;
  border-top-left-radius: 0;
}
```

3.5.2 خاصية أنماط الإطارات border-style

تُحدد خاصية border-style شكل خط الإطار الخاص بالعنصر لجميع حواف (الأيمن والأيسر والعلوي والسفلي) ويمكن أن تأخذ من قيمة واحدة إلى أربع قيم، تُحدد كلٌّ منها شكل خط الإطار لأحد الحواف.

مثال:

```
border-style: dotted;
border-style: dotted solid double dashed;
```



و بشكل عام يُحدَّد شكل خط الإطار لحواف العنصر حسب الترتيب التالي:

```
border-style: <ALL>; /* القيمة الواحدة تحدد شكل خط الإطار لجميع الأوجه*/
border-style: <top> <right> <bottom> <left>;
```

يمكن أن تأخذ خاصية border-style القيم none أو hidden ولهما نفس التأثير، إلَّا أن hidden تستخدم لحل مشكلة تداخل إطارات عنصر الجداول

مع إطارات خلايا الجدول حيث تُخفي اطارات الخلايا. بينما none تسبب في تداخل الإطارات المتلامسة وظهورها بسُمك يساوي مجموع سُمكي إطار الجدول وإطار الخلية.

3.5.3 إنشاء إطارات متعددة

مثال استخدام outline:

```
.div1{
    border: 3px solid black;
    outline: 6px solid blue;
```

```
    width: 100px;
    height: 100px;
    margin: 20px;
}
```

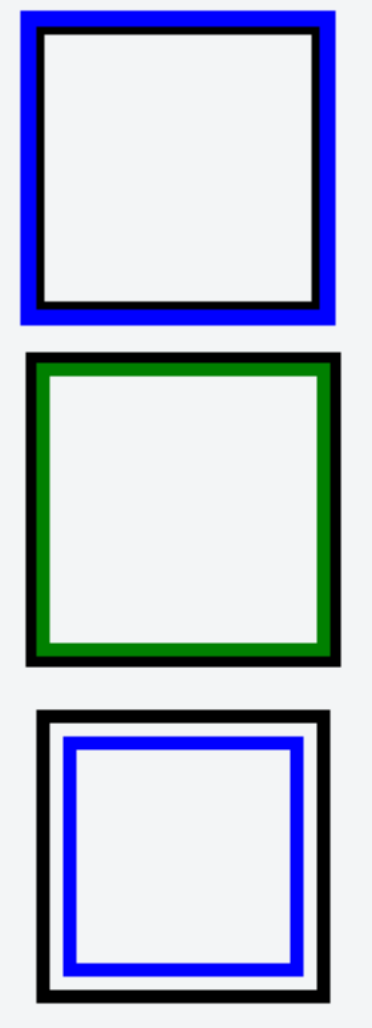
استخدام box-shadow:

```
.div2{
    border: 5px solid green;
    box-shadow: 0px 0px 0px 4px #000;
    width: 100px;
    height: 100px;
    margin: 20px;
}
```

استخدام العناصر الزائفة pseudo elements:

```
.div3 {
   position: relative;
   border: 5px solid #000;
   width: 100px;  height: 100px;
   margin: 20px;
}
.div3:before {
    content: " ";
    position: absolute;
    border: 5px solid blue;
    z-index: -1;  top: 5px;
    left: 5px;
    right: 5px;
    bottom: 5px;
}
```

النتيجة:



اطّلع على تجربة حيّة لهذا المثال على JSFiddle.

3.5.4 الصياغة المختزلة لإنشاء الإطارات

تحتاج في معظم الحالات لتعريف العديد من الخاصيات للإطار مثل `border-width`، و`border-style`، و`border-color` لكل أوجه العنصر، يُمكنك في هذه الحالات استعمال الصياغة المختزلة الموضحة في المثال التالي:

عوضًا عن كتابة:

```
border-width: 1px;
border-style: solid;
border-color: #000;
```

يمكنك كتابة:

```
border:1px solid #000;
```

يمكن استعمال هذه الطريقة لإنشاء إطار في اتجاه محدد، على سبيل المثال تُنشيء الشيفرة التالية إطارًا عُلويًا ذا خط مزدوج وثُخن بكسلين ولون رمادي.

```
border-top: 2px double #aaaaaa;
```

3.5.5 خاصية تداخل الإطارات border-collapse

تعمل خاصية `border-collapse` فقط على الجداول والعناصر المعروضة كجداول (أي تمتلك القاعدة `display: table` أو `display: inline-table`)، وتحدد ما إذا كان إطار الجدول منفصلًا عن إطار الخلايا أم أنه مُدمج معها (أي تتشارك الخلايا المتجاورة الإطار مع بعضها البعض).

```
table {
    border-collapse: separate; /* default */
    border-spacing: 2px; /* تضيف فراغ بين الإطارات المتلامسة */
}
```

لمزيد من المعلومات انظر توثيق خاصية border-collapse.

3.5.6 خاصية border-image

تسمح هذه الخاصية برسم صورة على إطار العنصر، مما يُسَهِّل عملية إنشاء إطارات ذات مظهر مخصص. وتستخدم هذه الخاصية بدلًا من أشكال الإطارات المُعرَّفة عبر الخاصية `border-style`.

تتكون خاصية `border-image` من ثلاث خاصيات هي:

- `border-image-source`: تحدد مسار الصورة المراد استخدامها كإطار.
- `border-image-slice`: تُقسِّم الصورة المحددة عبر الخاصية `border-image-source` إلى تسع مناطق، الأركان الأربعة، والحواف، والوسط.
- `border-image-repeat`: تُعرِّف كيفية عرض الجزء الأوسط من الصورة لكي تُطابق أبعاد الإطار. ويمكن استخدام قيمة واحدة لهذه الخاصية لضبط سلوك جميع الحواف، أو يمكن تحديد قيمتين لضبط سلوك الحواف الأفقية والرأسية كلًا على حدة.

مثال باستخدام صورة بحجم 90 بكسل × 90 بكسل:

```
border-image: url("border.png") 30 stretch;
```

تُقسَّم الصورة في المثال أعلاه لتسع مناطق كـل واحـدة بحجم 30 بكسـل × 30 بكسـل. وتسـتخدم حـواف الصورة كأركان للإطار والجوانب الأربعة لها تُمثل أوجه الإطار. وتتمدد الصورة لتُغطي كل العنصـر في حـال زيـادة ارتفاعه عن 30 بكسل.

3.5.7 إنشاء إطارات متعددة الألوان باستخدام خاصية border-image

انظر المثال التالي:

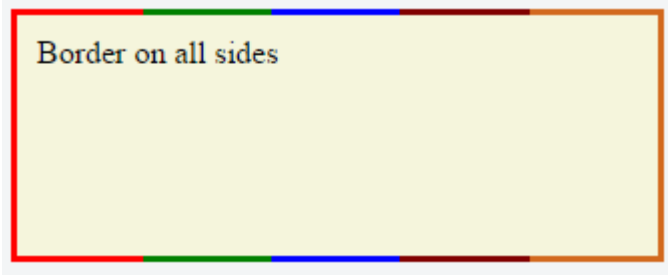
- ملف CSS:

```
.bordered {
    border-image: linear-gradient(to right, red 20%, green 20%, green 40%, blue 40%, blue 60%, maroon 60%, maroon 80%, chocolate 80%); /* gradient with required colors */
    border-image-slice: 1;
}
```

- ملف HTML:

```
<div class='bordered'>Border on all sides</div>
```

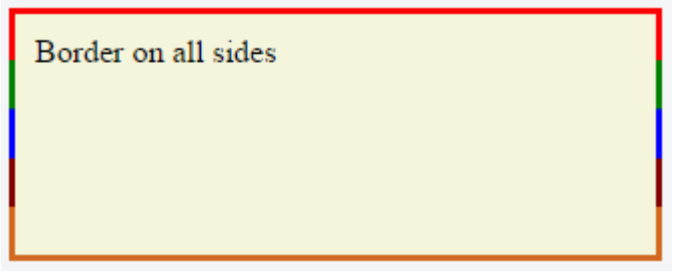
يرسم المثال أعلاه إطارًا مكون من 5 ألوان عُرِّفت باستخدام التدرج الخطي `linear-gradient` (يمكنـك زيارة التوثيق لمزيد من المعلومات عن هذه الخاصية وخاصية border-image-slice).



من المُلاحظ أن للإطار لون واحد فقط في الاتجاهين الأيمن والأيسر من الصندوق، وذلك يرجع للطريقة التي تعمل بها الخاصية `border-image` وهي تطبيق التدرج اللوني على كـل الصـندوق ومن ثم إخفـاء الصـورة من منطقة الحواشي والمحتوى مما يجعلها تبدو كإطار للصندوق.

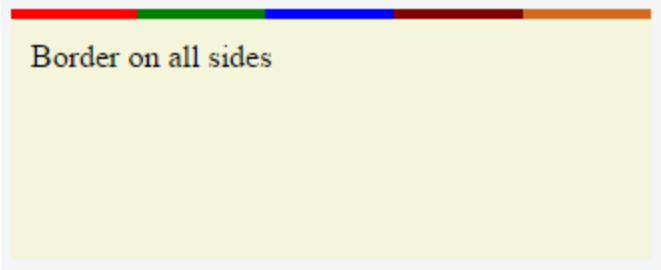
تُحدَّد الاتجاهات التي تأخذ اللون الواحد عبر تعريف التـدرج اللـوني، إذا كـان التـدرج من اليسـار إلى اليمين يأخذ الإطار الأيسر أول لون في التدرج بينما يأخذ الإطار الأيمن آخـر لـون في التـدرج. وإذا كـان التـدرج من أعلى لأسفل يأخذ الإطار العُلوي أول لون في التدرج بينما يأخذ الإطار السفلي آخر لون في التدرج.

مثال يوضح التدرج اللوني من أعلى لأسفل:



إذا كان المطلوب إضافة إطار باتجاه واحد فقـط، يمكن اسـتخدام خاصـية `border-width` وتحديـد قيمـة العرض بصفر بكسل في الاتجاهات التي لا ترغب في إضافة إطار عليها.

```
border-width: 5px 0px 0px 0px;
```



> لا يمكن تدوير إطارات العناصر التي تمتلك خاصية border-image، واستخدام خاصية border-radius معها يؤدي إلى تدوير الخلفية بدلاً من الإطار.

3.5.8 إنشاء إطار للعنصر باتجاه محدد

تسـتخدم الخاصـية `border-[left|right|top|bottom]` لعمـل إطـار للعنصـر باتجـاه محـدد، مثلًا الخاصية `border-left` تُضيف الإطار باتجاه اليسار فقط.

```
#element {
    border-left: 1px solid black;
}
```

3.6 حدود العنصر Outlines

حـدود العنصـر `outline` هي الخطـوط الـتي تُرسـم خـارج إطـار العنصـر، ولا تأخـذ مسـاحة في النمـوذج الصندوقي، وبالتالي لا تؤثر على موضع العنصر أو بقية العناصر في الصفحة. ويوضح الجدول التـالي المعـاملات التي يمكن استعمالها لضبط حدود العنصر:

المعامل	التفاصيل
dotted	خط حد مُنقَّط
dashed	خط حد متقطع
solid	خط حد متصل
double	خط حد مزدوج
groove	خط حد مموج ثلاثي الأبعاد، يتعمد على قيمة لون خط الحد outline-color
ridge	خط حد ثلاثي الأبعاد، يعتمد على قيمة لون خط الحد outline-color
inset	خط حد داخلي ثلاثي الأبعاد، يعتمد على قيمة لون خط الحد outline-color
outset	خط حد خارجي ثلاثي الأبعاد، يعتمد على قيمة لون خط الحد outline-color
none	حذف خط الحد
hidden	إخفاء خط الحد

تأخذ في بعض المتصفحات حدود العنصر شكلاً آخر غير المستطيل، غالبًا ما يحدث هذا عند تطبيق خاصية outline على عنصر span يحتوى على نصوص ذات أحجام خـط مختلفـة. وعلى عكس الإطـارات border، لا يمكن تدوير أركان حدود العنصر.

الخاصيات الأساسية لحدود العنصر هي: outline-color، وoutline-style، وoutline-width.

> على الرغم من التشابه بين حدود العنصر والإطارات في تأثيرهما وطريقة إنشائهما، إلَّا أنهما خاصيتين مختلفتين تمامًا.

يوضح المثال التالي كيفية إنشاء حدود العنصر:

```
outline:1px solid black;
```

3.6.1 خاصية outline-style

تحدد هذه الخاصية شكل حدود العنصـر ويمكن أن تأخـذ أيًـا من القيم الموضـحة في الجـدول الموضـح في بداية هذا الفصل. ويُبيِّن المثال التالي كيفية استخدامها:

- ملف CSS:

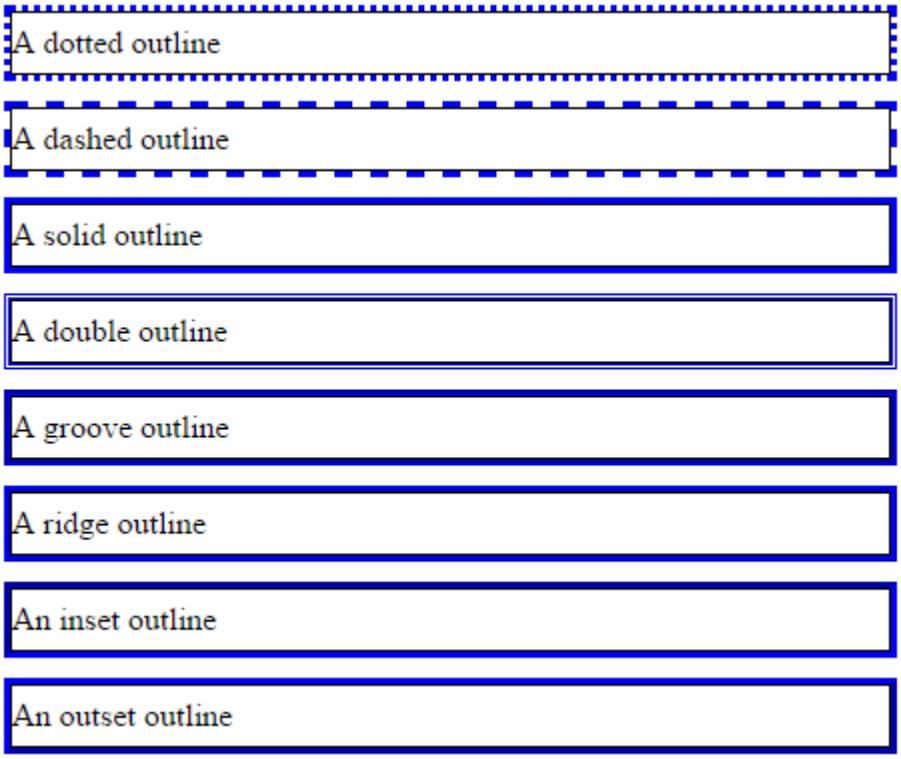
```
p{
border:1px solid black;
outline-color:blue;
line-height:30px;
}
```

```
.p1{
    outline-style:dotted;
}
.p2{
    outline-style:dashed;
}
.p3{
    outline-style:solid;
}
.p4{
    outline-style:double;
}
.p5{
    outline-style:groove;
}
.p6{
    outline-style:ridge;
}
.p7{
    outline-style:inset;
}
.p8{
    outline-style:outset;
}
```

- ملف HTML:

```
<p class="p1">A dotted outline</p>
<p class="p2">A dashed outline</p>
<p class="p3">A solid outline</p>
<p class="p4">A double outline</p>
<p class="p5">A groove outline</p>
<p class="p6">A ridge outline</p>
<p class="p7">A inset outline</p>
<p class="p8">A outset outline</p>
```

النتيجة:



3.7 الخاصية overflow

تأخذ الخاصية `overflow` إحدى القيم التالي:

المعامل	الوصف
`visible`	تُظهر المحتوى الفائض عن حجم العنصر دون اقتصاص.
`hidden`	اقتصاص المحتوى الفائض عن حجم العنصر دون إظهار شريط تمرير scroll bar.
`scroll`	اقتصاص المحتوى الفائض عن حجم العنصر مع إظهار شريط تمرير scroll bar.
`auto`	نفس عمل scroll، لكنها لا تضيف شريط التمرير scroll bar في حالة كان حجم المحتوى أقل من أو يساوي حجم الحاوية.
`inherit`	تُوَرِّث قيمة خاصية overflow الخاصة بالعنصر الأب للابن.

3.7.1 خاصية overflow-wrap

تُخبر الخاصية overflow-wrap المتصفح أنّ بإمكانـه تقسـيم النصـوص والكلمـات الطويلـة على عـدد من الأسطر لمنعها من تجاوز حدود صندوق المحتوى وتأخذ إحدى القيم التالي:

المعامل	الوصف
`normal`	تسمح للكلمة بالخروج من الحاوية إذا كانت أطول منها.
`break`	تقسم الكلمة على عدد من الأسطر إذا كانت أطول من الحاوية.
`inherit`	تُوِّرث قيمة overflow-wrap الخاصة بالعنصر الأب للابن.

إليك المثال التالي:

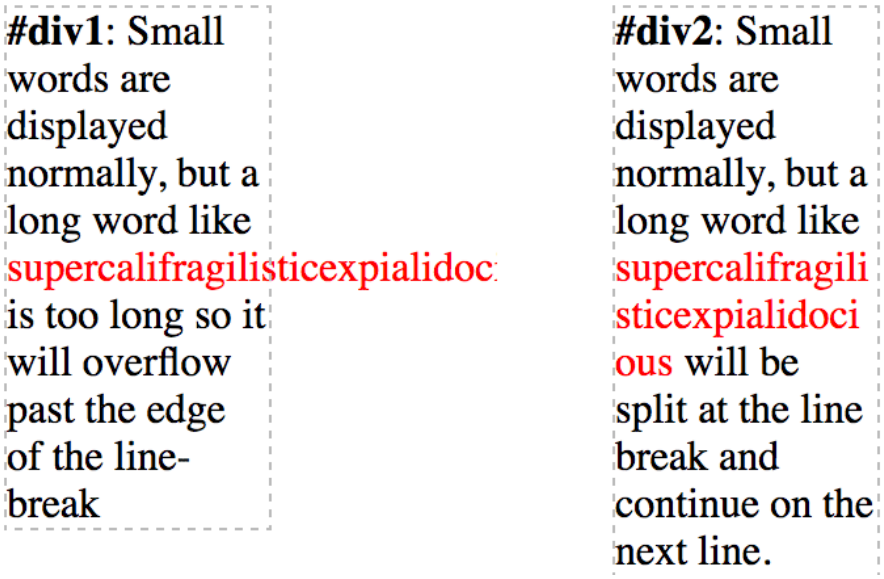
- ملف CSS:

```
div {
    width: 100px;
    outline: 1px dashed #bbb;
}

#div1 {
    overflow-wrap: normal;
}

#div2 {
    overflow-wrap: break-word
}
```

- ملف HTML:

```
<div id="div1">
    <strong>#div1</strong>: Small words are displayed normally but a
word like <span style="red;">supercalifragilisticexpialidocious</span>
is too long so it will overflow past the edge of the line-break
</div>


<div id="div2">
    <strong>#div2</strong>: Small words are displayed normally, but
a long word like <span
style="red">supercalifragilisticexpialidocious</span> will be split at
the line break and continue on the next line.
</div>
```

النتيجة:



3.7.2 خاصية overflow-x و overflow-y

تعمل هاتان الخاصيتان عملًا مشابهًا لخاصـية `overflow` ولهمـا نفس قيمهـا، الاختلاف هـو أن الخاصـية `overflow-x` تعمل في المحور الأفقي فقط بينما تعمل الخاصية `overflow-y` في المحور الرأسي.

- ملف HTML:

```
<div id="div-x">
    If this div is too small to display its contents,
    the content to the left and right will be clipped
</div>
<div id="div-y">
    If this div is too small to display its contents,
    the content to the top and bottom will be clipped
</div>
```

- ملف CSS:

```
div {
    width: 200px;
    height: 200px;
}

#div-x {
    overflow-x: hidden;
}
```

```
#div-y {
    overflow-y: hidden;
}
```

3.7.3 القاعدة overflow: scroll

انظر المثال التالي:

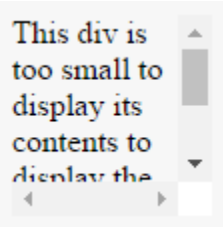
- ملف HTML:

```
<div>
    This div is too small to display its contents to display the
effects of the overflow property.
</div>
```

- ملف CSS:

```
div {
    width: 100px;
    height: 100px;
    overflow: scroll;
}
```

النتيجة:



في المثال أعلاه، مساحة الحاوية أقل من مساحة المحتوى مما يـؤدي إلى اقتصاصـه وإظهـار شـريط تمريـر أفقي وآخر رأسي، ويظهر المحتوى كاملًا عند الطباعة.

3.7.4 القاعدة overflow: visible

انظر المثال التالي:

- ملف HTML:

```
  <div>
      Even if this div is too small to display its content, the
content is not clipped
  </div>
```

- ملف CSS:

```
  div {
      width: 50px;
      height: 50px;
      overflow: visible;
  }
```

النتيجة:

لاحظ عدم اقتصاص المحتوى عند استعمال القاعدة `overflow: visible` وإنما يمتد خارج صندوق المحتوى.

3.7.5 سياق تنسيق العناصر الكتلية الناتج عن خاصية overflow

استعمال خاصية `overflow` بقيمة غير `visible` يتسبب في إنشاء سياق جديد لتنسيق الكُتل مما يُفيد في محاذاة العناصر الكتلية المجاورة للعناصر العائمة floated elements.

مثال:

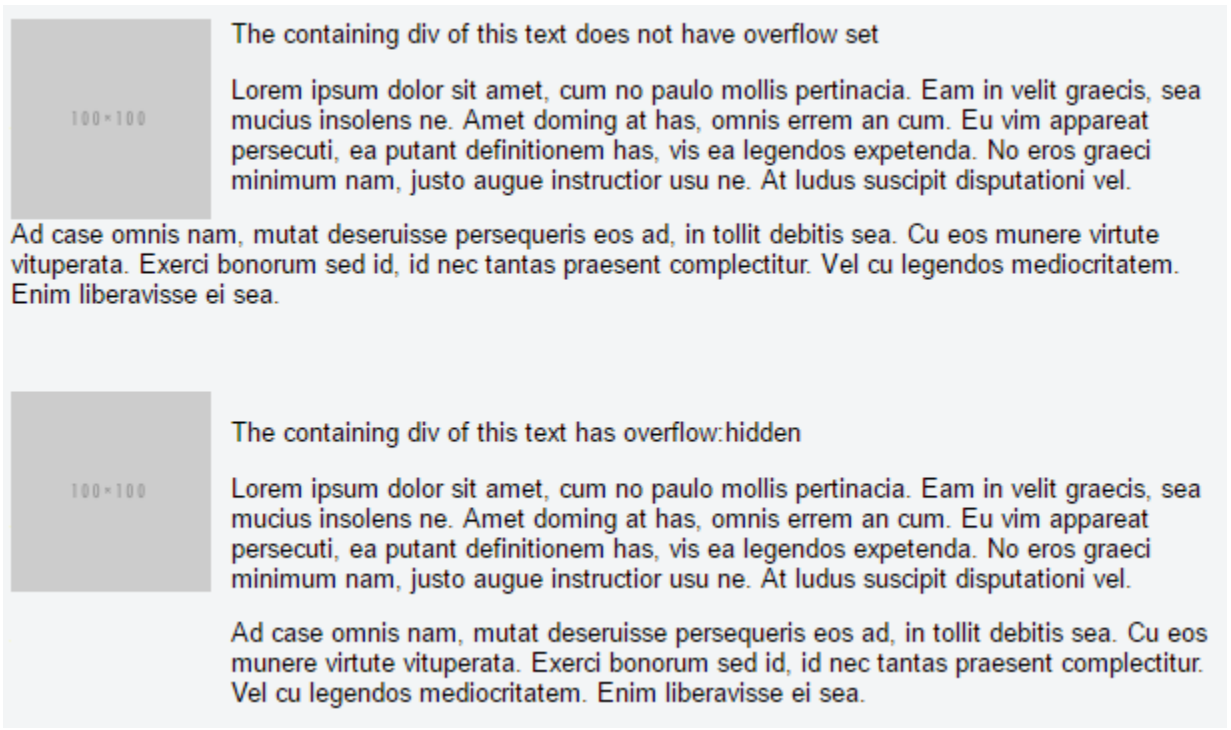
- ملف CSS:

```
img {
    float: left;
    margin-right: 10px;
}

div {
    overflow: hidden; /* تُنشيء سياق جديد لتنسيق العناصر الكتلية */
}
```

- ملف HTML:

```
<img src="http://placehold.it/100*100">
<div>
    <p>lorem ipsum dolor sit amet, cum no paulo mollis pertinácia.</p>
    <p>Ad case omnis nam, mutat deseruisse perseqeris eos ad, in tollit debitis sea.</p>
</div>
```

النتيجة:



دورة تطوير التطبيقات
باستخدام لغة بايثون

مميزات الدورة

- بناء معرض أعمال قوي بمشاريع حقيقية
- وصول مدى الحياة لمحتويات الدورة
- تحديثات مستمرة على الدورة مجانًا
- شهادة معتمدة من أكاديمية حسوب
- إرشادات من المدربين على مدار الساعة
- من الصفر دون الحاجة لخبرة مسبقة

اشترك الآن

4. التحكم في تموضع العناصر

تأخذ الخاصية `position` التي تضبط موضع العنصر في الصفحة القيم الموضحة في الجدول التالي والتي سنشرح كل واحدة منها بالتفصيل مع أمثلة.

المعامل	الوصف
`static`	تعرض العناصر بنفس ترتيبها كتابتها على ملف HTML.
`relative`	تُحدد موضع العنصر نسبةً لموضِعه الأصلي.
`fixed`	تُحدد موضع العنصر نسبةً لنافذة المتصفح.
`absolute`	تُحدد موضع العنصر نسبةً لأوَّل عنصر ذي موضِع نسبي.
`initial`	تُرجِع القيمة الابتدائية للخاصية.
`inherit`	ترث قيمة الخاصية من العنصر الأب.
`sticky`	يُعامَل العنصر كما لو أنَّه ذو موضع نسبي `relative` وبعد تجاوز حد معيّن من التمرير لأسفل فسيُعامَل على أنه ذو موضع ثابت `fixed`.
`unset`	مزيج بين `initial` و `inherit`. انظر توثيق CSS.

4.1 الموضِع المُطلق Absolute Position

عند استخدام الموضع المطلق، يُزال العنصر من البنية التنظيمية للصفحة، ولا يُحجَز له مكانٌ في تخطيطها، وإنما يُحدد موضعه نسبةً إلى أقرب عنصر له موضع نسبي، أو إلى العنصر `<body>`، ويمكن التحكم في موضـعه عبر الخواص `top` و `right` و `bottom` و `left`.

مثال:

```
.abspos {
    position: absolute;
    top: 0px;
    left: 500px;
}
```

4.2 الموضع الثابت Fixed Position

عند استخدام الموضع الثابت يُزال العنصر من البنية التنظيمية للصفحة، ولا يُحجَز له مكـانٌ في تخطيطهـا، وإنما يُحدد موضعه نسبةً إلى إطار العـرض viewport، يمكن التحكم في موضـعه عـبر الخـواص `top` و `right` و `bottom` و `left`، ومن الاستخدامات الشائعة للموضع الثابت استخدامه لتثبيت العنصر في مكانه عند تمرير الصفحة لأسفل.

مثال:

```
#stickyDiv {
    position: fixed;
    top: 10px;
    left: 10px;
}
```

4.3 الموضع النسبي Relative Position

يُحدد موضع العنصر بناءً على البنية التنظيمية للصفحة،ويمكن التحكم في إزاحته عن موضعه الأصـلي عـبر الخواص `top` و `right` و `bottom` و `left` ولا تؤثر هذه الإزاحة على مواضع بقية العناصر.

مثال:

```
.relpos {
    position: relative;
    top: 20px;
    left: 30px;
}
```

4.4 الموضع الافتراضي حيث يجب Static Position

يُحدد موضـع العنصـر بنـاءً على البنيـة التنظيميـة للصـفحة ولا يكـون للخـواص `top` و `right` و `bottom` و `left` و `z-index` أي أثر عليه.

مثال:

```
.element {
    position: static;
}
```

4.5 خاصية z-index

تُحدِّد الخاصية `z-index` ترتيب العناصر ذات الموضع المُحـدد positioned elements على المحـور Z، فعندما تتداخل العناصر تُحدد قيمة هذه الخاصية أيّ العناصر سيظهر فوق بقية العناصر، حيث يظهر العنصـر ذو القيمة الأكبر للخاصية z-index في الأعلى ويليه العنصر ذو القيمة الأقل مباشرة وهكذا.

مثال:

- ملف HTML

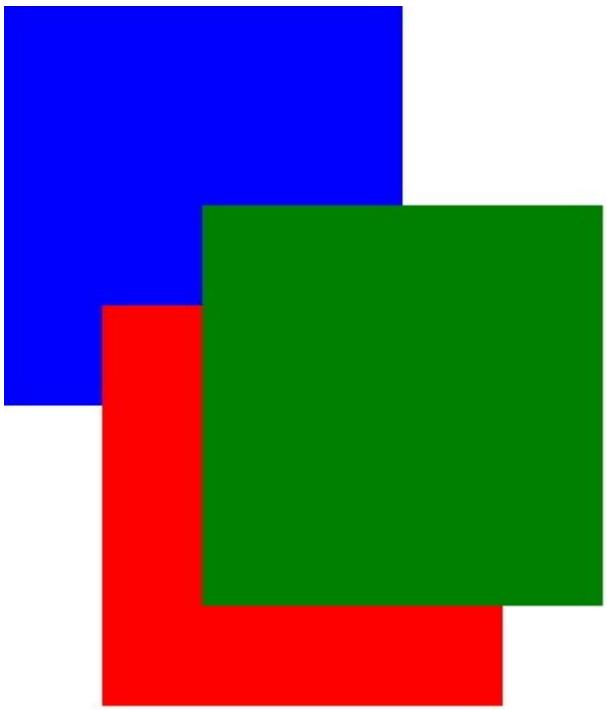
```
<div id="div1"></div>
<div id="div2"></div>
<div id="div3"></div>
```

- ملف CSS

```
div {
    position: absolute;
    height: 200px;
    width: 200px;
}

div#div1 {
    z-index: 1;
    left: 0px;
    top: 0px;
    background-color: blue;
}

div#div2 {
    z-index: 2;
    left: 100px;
    top: 100px;
```

```
        background-color: green;
    }

    div#div3 {
        z-index: 3;
        left: 50px;
        top: 150px;
        background-color: red;
    }
```

النتيجة:



اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

الصيغة العامة لكتابة الخاصية `z-index`:

```
z-index: [number] | auto;
```

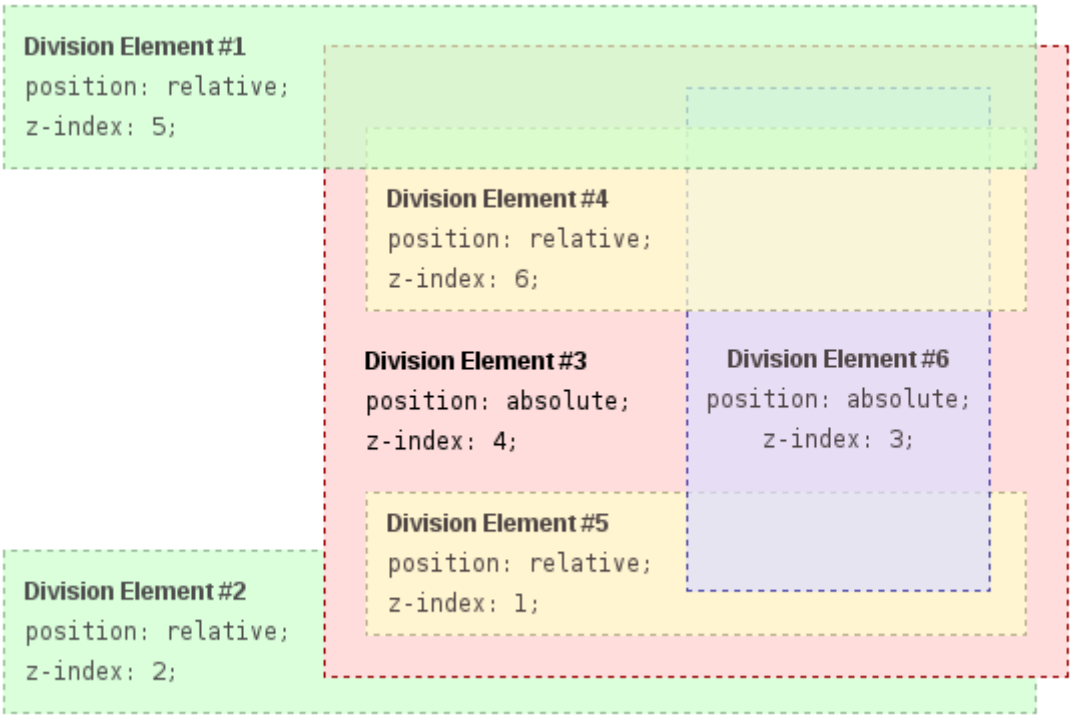
حيث:

- `[number]`: عدد صحيح يمثل ترتيب العنصر على المحور Z.
- `auto`: تُعطي العنصر نفس ترتيب أبيه على المحور Z.

يمكنك الرجوع إلى توثيق الخاصية z-index على موسوعة حسوب لمزيد من التفاصيل.

4.6 سياق التطبيق Stacking Context

في المثال أدناه، يُنشئ كل عنصر سياق تطبيق (من طبقات) خاص به، وبسبب مواضع العناصر و قيمة الخاصية `z-index` لهم، يُنشئ سياق تطبيق بالشكل التالي:



من المهم ملاحظة أن العنصرين `div #4` و `div #5` و `div #6` هم أبناء للعنصر `div #3` لـذلك يُنشـأ سياق التطبيق لهم بالنسبة له، وبعد ذلك ينشأ سياق التطبيق العام للصفحة. مثال:

- ملف HTML

```
<div id="div1">
    <h1>Division Element #1</h1>
    <code>position: relative;<br/>
    z-index: 5;</code>
</div>

<div id="div2">
    <h1>Division Element #2</h1>
    <code>position: relative;<br/>
    z-index: 2;</code>
</div>
```

```
    <div id="div3">
        <div id="div4">GoalKicker.com – CSS Notes for Professionals 210
            <h1>Division Element #4</h1>
            <code>position: relative;<br/>
            z-index: 6;</code>
        </div>
        <h1>Division Element #3</h1>
        <code>position: absolute;<br/>
        z-index: 4;</code>
        <div id="div5">
            <h1>Division Element #5</h1>
            <code>position: relative;<br/>
            z-index: 1;</code>
        </div>
        <div id="div6">
            <h1>Division Element #6</h1>
            <code>position: absolute;<br/>
            z-index: 3;</code>
        </div>
    </div>
```

- ملف CSS

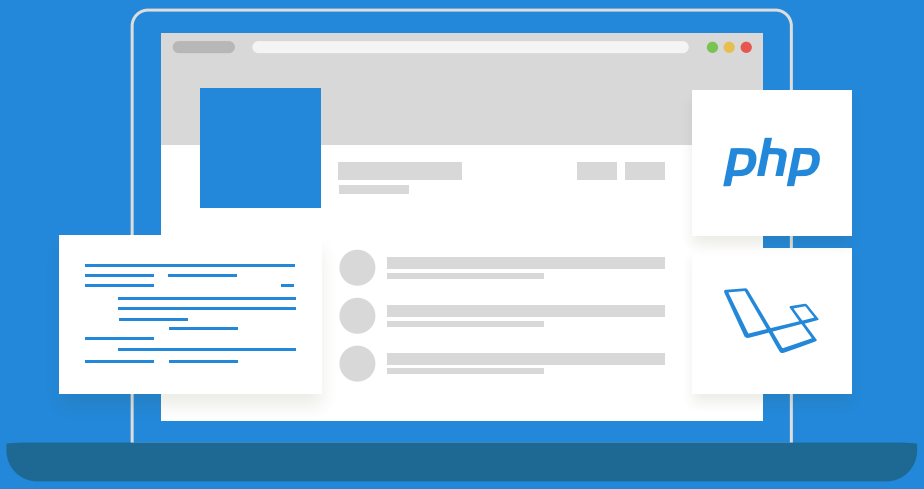
```
    * {
        margin: 0;
    }
    html {
        padding: 20px;
        font: 12px/20px Arial, sans-serif;
    }
    div {
        opacity: 0.7;
        position: relative;
    }
    h1 {
        font: inherit;
        font-weight: bold;
```

```
    }
    #div1,
    #div2 {
        border: 1px dashed #696;
        padding: 10px;
        background-color: #cfc;
    }
    #div1 {
        z-index: 5;
        margin-bottom: 190px;
    }
    #div2 {
        z-index: 2;
    }
    #div3 {
        z-index: 4;
        opacity: 1;
        position: absolute;
        top: 40px;
        left: 180px;
        width: 330px;
        border: 1px dashed #900;
        background-color: #fdd;
        padding: 40px 20px 20px;
    }
    #div4,
    #div5 {
        border: 1px dashed #996;
        background-color: #ffc;
    }
    #div4 {
        z-index: 6;
        margin-bottom: 15px;
        padding: 25px 10px 5px;
    }
    #div5 {
```

```
        z-index: 1;
        margin-top: 15px;
        padding: 5px 10px;
    }
    #div6 {
        z-index: 3;
        position: absolute;
        top: 20px;
        left: 180px;
        width: 150px;
        height: 125px;
        border: 1px dashed #009;
        padding-top: 125px;
        background-color: #ddf;
        text-align: center;
    }
```

المصدر: توثيق stacking context.

دورة تطوير تطبيقات الويب
باستخدام لغة PHP



احترف تطوير النظم الخلفية وتطبيقات الويب
من الألف إلى الياء دون الحاجة لخبرة برمجية مسبقة

التحق بالدورة الآن

5. تنسيق القوائم وإضافة الظلال ورسم الأشكال

5.1 تنسيق القوائم

يوضح الجدول التالي الخاصيات المستعملة في تنسيق القوائم مع وصفها:

المعامل	الوصف
`list-style-type`	تُحدد شكل أو نوع مؤشر القائمة، ويمكن أن تأخذ القيم `disc` أو `circle` أو `square` أو `decimal` أو `lower-roman` أو `upper-roman` أو `none`. انظر توثيقها لمعرفة جميع القيم المتاحة.
`list-style-position`	تُحدِّد مكان مؤشر القائمة.
`list-style-image`	تُحدِّد صورة تستخدم كمؤشر للقائمة.
`initial`	تُرجع القيمة الابتدائية للخاصية.
`inherit`	ترث أنماط القائمة من العنصر الأب.

يمكنك الرجوع إلى قسم القوائم في توثيق CSS الرسمي العربي لمزيد من التفاصيل.

مثال:

```
li {
    list-style-type: square;
}

li {
    list-style-image: url('images/bullet.png')
}
```

```
li {
    list-style-position: inside;
}
```

5.1.1 تموضع عناصر القائمة

لكل من عناصر القائمة أي عناصر <li> والعنصر الحاوي لها أي <ul> أو <ol> هوامش وحواشي خاصة به، وتؤثر هذه الهوامش والحواشي على موضع محتوى عنصر القائمة ، وبما أن المتصفحات المختلفة تضيف هوامش وحواشي بمساحات مختلفة مما يؤدي إلى اختلاف شكل الصفحة حسب المتصفح الذي تُعرض فيه، لذلك لابد من إعادة ضبط هذه المساحات يدويًا. لكل عنصر من عناصر القائمة صندوق يُسمى صندوق المؤشر marker box يوضع بداخلة مؤشر عناصر القائمة، ويمكن أن يكون هذا الصندوق داخل أو خارج صندوق عنصر القائمة <li>.

مثال على وضع صندوق المؤشر داخل صندوق عنصر القائمة:

```
list-style-position: insde;
```

مثال على وضع صندوق المؤشر خارج صندوق عنصر القائمة:

```
list-style-position: outside;
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

5.1.2 حذف المؤشر أو الترقيم من عناصر القائمة

في بعض الأحيان نحذف المؤشر أو الترقيم من عناصر القائمة، ويجب في هذه الحالات التخلص من مساحة صندوق المؤشر كما هو موضح في المثال التالي:

- ملف HTML

```
<ul>
    <li>first item</li>
    <li>second item</li>
</ul>
```

- ملف CSS

```
ul {
    list-style-type: none;
}
```

```
li {
    margin: 0;
    padding: 0;
}
```

5.1.3 تحديد شكل المؤشر أو نوع الترقيم لعناصر القائمة

تحديد شكل المؤشر لعناصر القائمة الغير مرتبة <ul>:

```
list-style: disc; /* A filled circle (default) */
list-style: circle; /* A hollow circle */
list-style: square; /* A filled square */
list-style: '-'; /* any string */
```

تحديد نوع الترقيم لعناصر القائمة المرتبة <li>:

```
list-style: decimal; /* Decimal numbers beginning with 1 (default) */
list-style: decimal-leading-zero;/* Decimal numbers padded by initial
zeros (01, 02, 03, … 10) */
list-style: lower-roman; /* Lowercase roman numerals (i., ii., iii.,
iv., ...) */
list-style: upper-roman; /* Uppercase roman numerals (I., II., III.,
IV., ...) */
list-style-type: lower-greek; /* Lowercase roman letters (α., β., γ.,
δ., ...) */
list-style-type: lower-alpha; /* Lowercase letters (a., b., c.,
d., ...) */
list-style-type: lower-latin; /* Lowercase letters (a., b., c.,
d., ...) */
list-style-type: upper-alpha; /* Uppercase letters (A., B., C.,
D., ...) */
list-style-type: upper-latin; /* Uppercase letters (A., B., C.,
D., ...) */
```

5.1.4 تنسيق العدَّادات Counters

يوضح الجدول التالي ما يتعلق بخاصيات تنسيق العدادات:

المعامل	الوصف
counter-name	اسم العداد.
integer	(اختياري) يُظهر القيمة الابتدائية للعداد، أو القيمة التي سيزيد العداد بمقدارها.

المعامل	الوصف
none	القيمة الابتدائية لجميع خاصيات العداد.
counter-style	تُحدد النمط الذي ستظهر به قيمة العداد.
connector-string	تُحدد العلامة أو النص الذي سيُعرض بين قيم العداد، علامة النقطة (.) في 2.1.1 مثلًا.

ا. استخدام الأرقام الرومانية عبر عدَّادات CSS

انظر المثال التالي:

- ملف CSS

```
body {
    counter-reset: item-counter;
}
.item {
    counter-increment: item-counter;
}
.item:before {
    content: counter(item-counter, upper-roman) ". ";
}
```

- ملف HTML

```
<div class='item'>Item No: 1</div>
<div class='item'>Item No: 2</div>
<div class='item'>Item No: 3</div>
```

في هذا المثال ستُعرض قيم العداد بالأرقام الرومانية (I، II، III).

ب. ترقيم العناصر في صفحة HTML باستخدام العدادات

انظر المثال التالي:

- ملف HTML

```
<div class='item'>
    <div class='item-header'>Item 1 Header</div>
    <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>
</div>
```

```
    <div class='item'>
        <div class='item-header'>Item 2 Header</div>
        <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>
    </div>
    <div class='item'>
        <div class='item-header'>Item 3 Header</div>
        <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>
    </div>
```

- ملف CSS

```
    body {
        counter-reset: item-counter; /* إنشاء العداد */
    }
    .item {
        counter-increment: item-counter; /* تزيد قيمة العداد لكل تكرار للصنف item
*/
    }
    .item-header:before {
        content: counter(item-counter) ". "; /* طباعة قيمة العداد */
    }

    item {
        border: 1px solid;
        height: 100px;
        margin-bottom: 10px;
    }
    .item-header {
        border-bottom: 1px solid;
        height: 40px;
        line-height: 40px;
        padding: 5px;
    }
    .item-content {
        padding: 8px;
    }
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

ج. إنشاء ترقيم متعدد المستويات

انظر المثال التالي:

- ملف HTML

```
<ul>
    <li>Level 1
        <ul>
            <li>Level 1.1
                <ul>
                    <li>Level 1.1.1</li>
                </ul>
            </li>
        </ul>
    </li>

    <li>Level 2
        <ul>
            <li>Level 2.1
                <ul>
                    <li>Level 2.1.1</li>
                    <li>Level 2.1.2</li>
                </ul>
            </li>
        </ul>
    </li>
</ul>
```

في هذا المثال استخدمنا مفهوم العدَّادات المتداخلة لإنشاء ترقيم متعدد المستويات، الفكرة الأساسية للعدادات المتداخلة أنّه إذا كان للعنصر عدَّاد باسم معين وأُجبر على إنشاء عدَّاد آخر بنفس الاسم فإنه سيُنشِئه كابن للعداد الحالي.

في هذا المثال، عنصر `<ul>` الثاني يرث العداد `list-item-number` من العنصر الأب، ومن ثم عليه إنشاء عداد جديد بنفس الاسم لأبنائه، وبالتالي يُنشئ العداد `list-item-number[1]` كفرع من العداد `list-item-number[0]`. ولعرض النتيجة نستخدم الدالة `counters()` بدلًا عن `counter()` لأننا نمتلك أكثر من عدَّاد واحد.

5.2 تطبيق ظلال على العنصر

تسمح الخاصية box-shadow بإنشاء ظلال للعنصر، وإذا كانت الخاصـية border-radius مُحـدَّدة على العنصر مع هذه الخاصية، فستأخذ الظلال شكل الحواف المدورة.

يوضح الجدول التالي القيم التي يمكن تأخذها الخاصية box-shadow:

المعامل	الوصف
inset	تؤدي إلى إنشاء ظلال داخل إطار العنصر.
offset-x	الإزاحة الأفقية للظل عن العنصر.
offset-y	الإزاحة الرأسية للظل عن العنصر.
blur-radius	كلما كبرت القيمة زاد تأثير الضبابية، وبالتالي سيصبح الظل أكبر وأكشف لونًا،
spread-radius	القيم الموجبة لها ستؤدي إلى جعل الظل يتوسع ويكبر، والقيم السالبة ستجعل الظل يتقلص ويصغر، وإذا لم تُحدَّد هذه القيمة فستكون 0 (أي أنَّ قياس الظل سيبقى كما هو).
color	لون الظل.

5.2.1 إنشاء ظل أسفل العنصر باستخدام العناصر الزائفة

انظر المثال التالي:

- ملف HTML

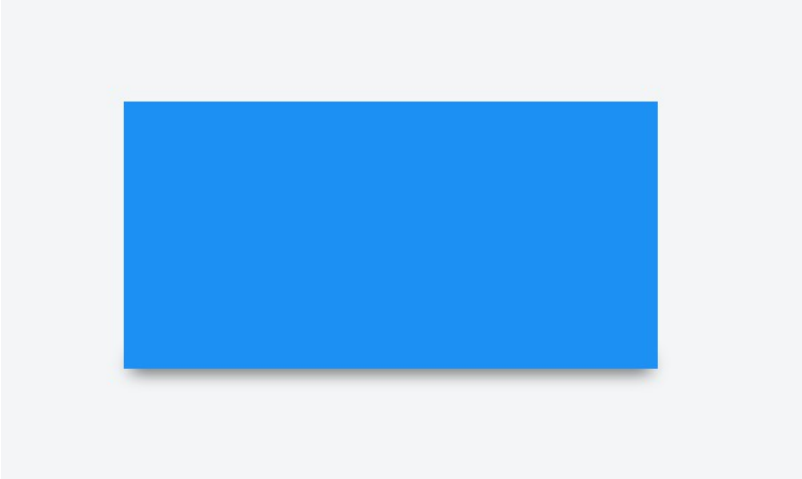
```
<div class="box_shadow"></div>
```

- ملف CSS

```
box_shadow {
    background-color: #1C90F3;
    width: 200px;
    height: 100px;
    margin: 50px;
}
.box_shadow:after {
    content: "";
    width: 190px;
    height: 1px;
    margin-top: 98px;
```

```
        margin-left: 5px;
        display: block;
        position: absolute;
        z-index: -1;
        -webkit-box-shadow: 0px 0px 8px 2px #444444;
        -moz-box-shadow: 0px 0px 8px 2px #444444;
        box-shadow: 0px 0px 8px 2px #444444;
    }
```

النتيجة:



اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

5.2.2 إنشاء ظلال خارجية لأوجه العنصر الأربعة

انظر المثال التالي:

- ملف HTML

```
    <div class="box_shadow"></div>
```

- ملف CSS

```
    .box_shadow {
        -webkit-box-shadow: 0px 0px 10px -1px #444444;
        -moz-box-shadow: 0px 0px 10px -1px #444444;
        box-shadow: 0px 0px 10px -1px #444444;
    }
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

5.2.3 إنشاء ظلال داخلية للعنصر

انظر المثال التالي:

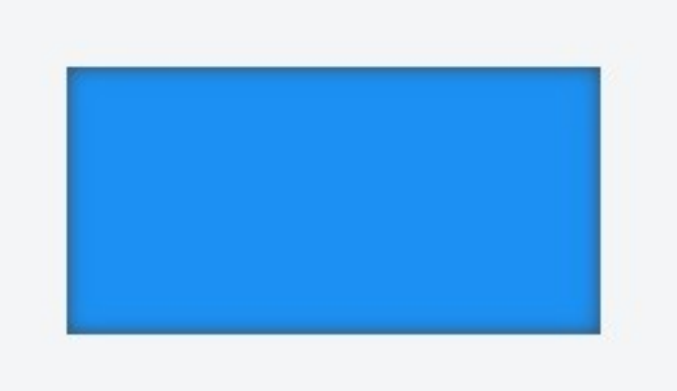
- ملف HTML

```
<div class="box_shadow"></div>
```

- ملف CSS

```
box_shadow {
    background-color: #1C90F3;
    width: 200px;
    height: 100px;
    margin: 50px;
    -webkit-box-shadow: inset 0px 0px 10px 0px #444444;
    -moz-box-shadow: inset 0px 0px 10px 0px #444444;
    box-shadow: inset 0px 0px 10px 0px #444444;
}
```

النتيجة:



اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

5.2.4 إضافة ظلال متعددة للعنصر

انظر المثال التالي:

- ملف HTML

```
<div class="box_shadow"></div>
```

- ملف CSS

```
box_shadow {
    width: 100px;
    height: 100px;
    margin: 100px;
    box-shadow:
    -52px -52px 0px 0px #f65314,
    52px -52px 0px 0px #7cbb00,
    -52px 52px 0px 0px #00a1f1,
    52px 52px 0px 0px #ffbb00;
}
```

النتيجة:

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

5.3 رسم الأشكال باستخدام CSS

5.3.1 رسم شبه المنحرف Trapezoid

لرسم شكل شبه المنحرف، أنشئ عنصر كتلي بارتفاع صفر، وعرض أكبر من الصفر، وإطار شفاف لجميع الأوجه عدا وجه واحد.

- ملف HTML

```
    <div class="trapezoid"></div>
```

- ملف CSS

```
    .trapezoid {
        width: 50px;
        height: 0;
        border-left: 50px solid transparent;
        border-right: 50px solid transparent;
        border-bottom: 100px solid black;
    }
```

5.3.2 رسم المثلثات Triangles

لرسم المثلث أنشئ عنصر كتلي بارتفاع وعرض صفر، ويُرسم الشكل باستخدام الإطارات.

مثال على رأس المثلث إلى أعلى:

```
<div class="triangle-up"></div>

<style>
.triangle-up {
    width: 0;
    height: 0;
    border-left: 25px solid transparent;
    border-right: 25px solid transparent;
    border-bottom: 50px solid rgb(246, 156, 85);
}
</style>
```

النتيجة:

مثال على رأس المثلث إلى أسفل:

```
<div class="triangle-down"></div>

<style>
.triangle-down {
    width: 0;
    height: 0;
    border-left: 25px solid transparent;
    border-right: 25px solid transparent;
    border-top: 50px solid rgb(246, 156, 85);
}
</style>
```

النتيجة:

مثال على رأس المثلث إلى اليمين:

```
<div class="triangle-right"></div>

<style>
.triangle-right {
    width: 0;
    height: 0;
    border-top: 25px solid transparent;
    border-bottom: 25px solid transparent;
    border-left: 50px solid rgb(246, 156, 85);
}
</style>
```

النتيجة:

مثال على رأس المثلث إلى اليسار:

```
<div class="triangle-left"></div>

<style>
.triangle-left {
    width: 0;
    height: 0;
    border-top: 25px solid transparent;
    border-bottom: 25px solid transparent;
    border-right: 50px solid rgb(246, 156, 85);
}
</style>
```

النتيجة:

مثال على المثلث يشير إلى أعلى اليمين:

```
<div class="triangle-up-right"></div>

<style>
.triangle-up-right {
    width: 0;
    height: 0;
    border-top: 50px solid rgb(246, 156, 85);
    border-left: 50px solid transparent;
}
</style>
```

النتيجة:

مثال على المثلث يشير إلى أعلى اليسار:

```
<div class="triangle-up-left"></div>

<style>
.triangle-up-left {
    width: 0;
    height: 0;
    border-top: 50px solid rgb(246, 156, 85);
    border-right: 50px solid transparent;
}
</style>
```

النتيجة:

مثال على المثلث يشير إلى أسفل اليمين:

```
<div class="triangle-down-right"></div>

<style>
.triangle-down-right {
    width: 0;
    height: 0;
    border-bottom: 50px solid rgb(246, 156, 85);
    border-left: 50px solid transparent;
}
</style>
```

النتيجة:

مثال على المثلث يشير إلى أسفل اليسار:

```
<div class="triangle-down-left"></div>

<style>
.triangle-down-left {
    width: 0;
    height: 0;
    border-bottom: 50px solid rgb(246, 156, 85);
    border-right: 50px solid transparent;
}
</style>
```

النتيجة:

5.3.3 رسم الدائرة

لرسم الدائرة أنشئ عنصر كتلة بارتفاع وعرض متساويين، ومن ثم حـدد قيمـة الخاصـية `border-radius` لتكون 50%.

```
<div class="circle"></div>

<style>
.circle {
    width: 50px;
    height: 50px;
    background: rgb(246, 156, 85);
    border-radius: 50%;
}
</style>
```

النتيجة:

5.3.4 رسم الشكل البيضاوي

لرسم الشكل البيضاوي أنشئ عنصر كتلة بارتفاع وعرض مختلفين، ومن ثم حدد قيمة الخاصية `border-radius` لتكون 50%.

```
<div class="oval"></div>

<style>
.circle {
    width: 50px;
    height: 80px;
    background: rgb(246, 156, 85);
    border-radius: 50%;
}
</style>
```

النتيجة:

5.3.5 شكل الانفجار

مثال على رسم شكل انفجار بثماني نقاط:

```
<div class="burst-8"></div>

<style>
.burst-8 {
    background: rgb(246, 156, 85);
    width: 40px;
    height: 40px;
    position: relative;
    text-align: center;
    -ms-transform: rotate(20deg);
    transform: rotate(20eg);
}
```

```
.burst-8::before {
    content: "";
    position: absolute;
    top: 0;
    left: 0;
    height: 40px;
    width: 40px;
    background: rgb(246, 156, 85);
    -ms-transform: rotate(135deg);
    transform: rotate(135deg);
}
</style>
```

النتيجة:

مثال على رسم شكل انفجار باثنتي عشر نقطة:

```
<div class="burst-12"></div>

<style>
burst-12 {
    width: 40px;
    height: 40px;
    position: relative;
    text-align: center;
    background: rgb(246, 156, 85);
}

.burst-12::before, .burst-12::after {
    content: "";
    position: absolute;
    top: 0;
    left: 0;
    height: 40px;
```

```
    width: 40px;
    background: rgb(246, 156, 85);
}

.burst-12::before {
    -ms-transform: rotate(30deg);
    transform: rotate(30deg);
}

.burst-12::after {
    -ms-transform: rotate(60deg);
    transform: rotate(60deg);
}
</style>
```

النتيجة:

5.3.6 رسم المربع

لرسم مربع أنشئ عنصر كتلة بارتفاع وعرض متساويين.

مثال:

```
<div class="square"></div>

<style>
.square {
    width: 100px;
    height: 100px;
    background: rgb(246, 156, 85);
}
</style>
```

النتيجة:

5.3.7 رسم المكعب

تستخدم خاصيات التحويلات ثنائية الأبعاد `skewX()` و `skewY()` على العناصر الزائفة لرسم المكعب.

مثال:

```
<div class="cube"></div>

<style>
cube {
    background: #dc2e2e;
    width: 100px;
    height: 100px;
    position: relative;
    margin: 50px;
}

.cube::before {
    content: '';
    display: inline-block;
    background: #f15757;
    width: 100px;
    height: 20px;
    transform: skewX(-40deg);
    position: absolute;
    top: -20px;
    left: 8px;
}

.cube::after {
    content: '';
```

```
    display: inline-block;
    background: #9e1515;
    width: 16px;
    height: 100px;
    transform: skewY(-50deg);
    position: absolute;
    top: -10px;
    left: 100%;
}
</style>
```

النتيجة:

اطلع على تجربة حيَّة لهذا المثال على JSFiddle.

5.3.8 رسم الهرم

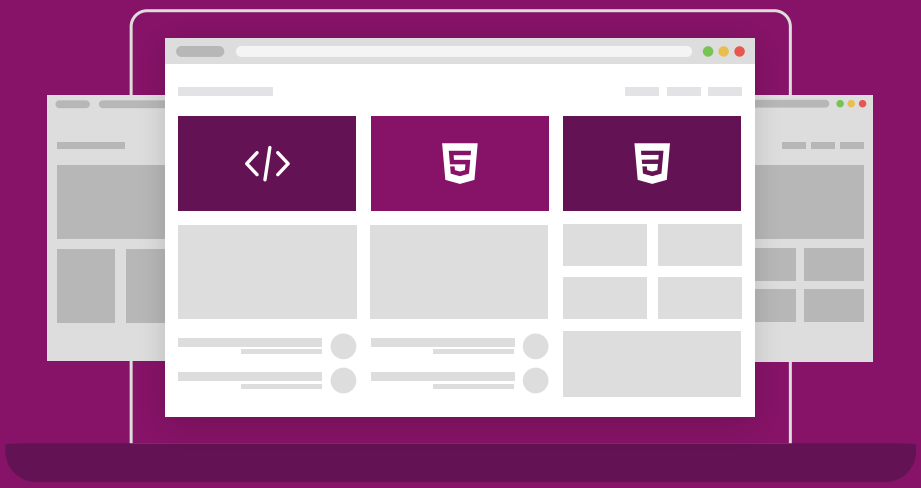
تستخدم الإطارات مع التحويلات ثنائية الأبعاد `skewY()` و `rotate()` على العناصر الزائفة لرسم شكل الهرم.

```
<div class="pyramid"></div>

<style>
.pyramid {
    width: 100px;
    height: 200px;
    position: relative;
    margin: 50px;
}
```

```
.pyramid::before,.pyramid::after {
    content: '';
    display: inline-block;
    width: 0;
    height: 0;
    border: 50px solid;
    position: absolute;
}

.pyramid::before {
    border-color: transparent transparent #ff5656 transparent;
    transform: scaleY(2) skewY(-40deg) rotate(45deg);
}

.pyramid::after {
    border-color: transparent transparent #d64444 transparent;
    transform: scaleY(2) skewY(40deg) rotate(-45deg);
}
</style>
```

النتيجة:

دورة تطوير واجهات المستخدم



مميزات الدورة

- بناء معرض أعمال قوي بمشاريع حقيقية
- وصول مدى الحياة لمحتويات الدورة
- تحديثات مستمرة على الدورة مجانًا
- شهادة معتمدة من أكاديمية حسوب
- إرشادات من المدربين على مدار الساعة
- من الصفر دون الحاجة لخبرة مسبقة

اشترك الآن

6. التنسيقات الأساسية للعناصر

6.1 تنسيقات الخطوط

يمكن استعمال الخاصيات التالية في CSS لتنسيق الخطوط:

الخاصية	الوصف
`font-style`	نمط الخط، مائل italics أو منحرف oblique.
`font-variant`	تأخذ إحدى القيمتين normal أو small-caps.
`font-weight`	ثُخن الخط، وتأخذ القيم normal أو bold أو قيم رقمية بين 100 و 900.
`font-size`	حجم الخط ويمكن أن يكون نسبة مئوية أو بالوحدات px أو em أو غيرها من الوحدات.
`line-height`	ارتفاع السطر ويمكن أن يكون نسبة مئوية أو بالوحدات px أو em أو غيرها من وحدات CSS.
`font-family`	نوع الخط.
`color`	لون الخط.
`font-stretch`	تختار نسخة ضيقة condensed أو عادية normal أو مُوَسَّعة expanded من الخط.
`text-align`	محاذاة النص، ارجع إلى توثيقها لمعرفة جميع القيم التي تأخذها.
`text-decoration`	زخرفة الخط، ارجع إلى توثيقها لمعرفة جميع القيم التي تأخذها..

6.1.1 الصياغة المختزلة لخاصية الخطوط font

يُمكن تعريف جميع خاصيات الخط في سطر واحد باستخدام الصياغة المختزلة الموضحة في المثال التالي:

```
element {
    font: [font-style] [font-variant] [font-weight] [font-size/line-height] [font-family];
}
```

مثال:

```
p {
    font-weight: bold;
    font-size: 20px;
    font-family: Arial, sans-serif;
}
```

يُمكن إعادة كتابة الشيفرة أعلاه في سطر واحد باستخدام الصياغة المختزلة كما هو موضح في المثال التالي:

```
p {
    font: bold 20px Arial, sans-serif;
}
```

انتبه إلى أن الخاصيات `font-style` و`font-variant` و`font-weight` و`line-height` خاصيات اختيارية وعدم تحديدها في الصياغة المختزلة يعني استخدام القيم الابتدائية لها، أما الخاصيتين `font-size` و `font-family` فهما خاصيتين ضروريتين وعدم تحديدهما يؤدي إلى تجاهل الشيفرة وعدم تطبيقها.

القيم الابتدائية لخاصيات الخطوط:

```
font-style: normal;
font-variant: normal;
font-weight: normal;
font-stretch: normal;
font-size: medium;
line-height: normal;
font-family: /* يعتمد على نوع الجهاز المستخدم لعرض الصفحة */
```

6.1.2 الاقتباسات

تستخدم الخاصية `quotes` لتحديد شكل علامات التنصيص للعنصر `<q>`:

```
q {
    quotes: "<" ">";
}
```

6.1.3 حجم الخط

تستخدم الخاصية `font-size` لتحديد حجم الخط ويوضح المثال التالي كيفية استخدامها:

- ملف HTML

```
<div id="element-one">Hello I am some text.</div>
<div id="element-two">Hello I am some smaller text.</div>
```

- ملف CSS

```
#element-one {
    font-size: 30px;
}

#element-two {
    font-size: 10px;
}
```

بتطبيق الشيفرة أعلاه يصبح حجم الخط داخل العنصر `#element-one` هو 30 بكسل، و 10 بكسلات داخل العنصر `#element-two`.

6.1.4 تعريف عدة أنواع من الخطوط

يمكن تحديد عدد من أنواع الخطوط عبر الخاصية `font-family`، ويُعطي المتصفح الأولوية للخط الأول في التعريف وفي حال عدم توفره ينتقل للخط الثاني وهكذا.

```
font-family: 'Segoe UI', Tahoma, sans-serif;
font-family: Consolas, 'Courier New', monospace;
```

> إذا تكون اسم الخط من أكثر من كلمة واحدة فيجب وضعه بين علامتي تنصيص.

6.1.5 الخاصية font-variant

تأخذ القيم الموضحة بالجدول التالي:

القيمة	الوصف
`normal`	تستعمل الخاصيات العادية للخط.
`small-caps`	تُحول الحروف الصغيرة لحروف كبيرة ولكن بنفس حجم الخطوط الصغيرة مع تصغير الحروف الصغيرة قليلًا.

مثال:

- ملف HTML

```
<p class="smallcaps">
    Documentation about CSS Fonts
    <br>
    aNd ExAmpLe
</p>
```

- ملف CSS

```
.smallcaps {
    font-variant: small-caps;
}
```

النتيجة:

DOCUMENTATION ABOUT CSS FONTS
AND EXAMPLE

لاحظ أن الخاصية `font-variant` خاصية مُختزلة، وتضم الخاصيات التالية:

```
font-variant-caps
font-variant-numeric
font-variant-alternates
font-variant-ligatures
font-variant-east-asian
```

6.2 اتجاه الكتابة

انظر المثال التالي:

```
div {
    /* اتجاه الكتابة من اليسار لليمين */
    direction: ltr;
}

.ex {
    /* اتجاه الكتابة من اليمين لليسار */
    direction: rtl;
}

.horizontal-tb {
    /* اتجاه الكتابة أفقي من اليمين إلى اليسار */
    writing-mode: horizontal-tb;
}

.vertical-rtl {
    /* اتجاه الكتابة رأسيًا من أعلى لأسفل وأفقيًا من اليمين إلى اليسار */
    writing-mode: vertical-rl;
}

.vertical-ltr {
    /* اتجاه الكتابة رأسيًا من أعلى لأسفل وأفقيًا من اليسار إلى اليمين */
    writing-mode: vertical-lr;
}
```

تستخدم الخاصية `direction` لتحديد الاتجاه الأفقي للكتابة (من اليمين إلى اليسار أو العكس)، ولها أربع قيم هي:

```
direction: ltr | rtl | initial | inherit;
```

تستخدم الخاصية `writing-mode` لتغيير محاذاة النص، فإما أن يُكتب من أعلى لأسفل أو من اليسار إلى اليمين.

```
writing-mode: horizontal-tb | vertical-rl | vertical-lr;
```

6.3 خاصيات تنسيق النصوص والأحرف

6.3.1 طفحان النص

تُحدِّد الخاصية `text-overflow` ماذا سيحدث عند زيادة النص عن مساحة صندوقه، فيمكن أن يُقص المحتوى، أو أن تُعرض ثلاث نقاط (...)، أو أن تُعرض سلسلة نصية خاصة.

```
.text {
    overflow: hidden;
    text-overflow: ellipsis;
}
```

تستخدم القيمة `ellipsis` لتمثيل النص المقصوص بثلاث نقاط تُعرض داخل صندوق المحتوى مما يؤدي إلى تقليل كمية النص المعروض إذا لم تكن هناك مساحة كافية لعرض النقاط.

6.3.2 إضافة تأثير الظلال على النص

تستخدم الخاصية `text-shadow` لإضافة تأثير الظلال على النص، وتقبل تحديد أكثر من ظل (يُفصَل بين قيمها بفاصلة)، ويُطبَّق الظل على النص وعلى خطوط الزخرفة `text-decoration` التابعة للعنصر، وتوضح الشيفرة التالية الصيغة العامة لاستخدامها:

```
text-shadow: horizontal-offset vertical-offset blur color;
```

مثال عن إضافة ظِل بلون معين:

```
h1 {
    text-shadow: 2px 2px #0000FF;
}
```

مثال عن إضافة ظل مع تأثير ضبابي:

```
h1 {
    text-shadow: 2px 2px 10px #0000FF;
}
```

مثال عن إضافة ظلال متعددة:

```
h1 {
    text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
}
```

6.4 حالة الأحرف

تستخدم الخاصية `text-transform` لتحديد حالة الأحرف التي سيُعرَض فيها العنصر؛ فإما أن تُعرض كل الأحرف بالحالة الكبيرة uppercase أو بالحالة الصغيرة lowercase أو أن يُعرض أول حرف من كل كلمة كبيرًا.

إليك مثال:

- ملف HTML

```
<p class="example1">
    all letters in uppercase
</p>

<p class="example2">
    all letters in Capitalize
</p>

<p class="example3">
    all letters in lowercase
</p>
```

- ملف CSS

```
.example1 {
    text-transform: uppercase;
}

.example2 {
    text-transform: capitalize;
}

.example3 {
    text-transform: lowercase;
}
```

6.4.1 التباعد بين الأحرف

تحدد الخاصية `letter-spacing` التباعد بين الأحرف النصية.

```
h2 {
    /* تزيد المسافة بين الأحرف بمقدار 1 بكسل */
    letter-spacing: 1px;
}

p {
    /* القيمة السالبة تعني تقليل المسافة بين الأحرف  */
    letter-spacing: -1px;
}
```

يمكنك الرجوع إلى توثيق letter-spacing في موسوعة حسوب لمزيد من التفصيل.

6.4.2 المسافة البادئة النص

تُحَدِّد الخاصية `text-indent` مقدار المسافة التي توضع قبل أول سطر في النص داخل عناصر الكتلة.

```
p {
    text-indent: 50px;
}
```

يمكنك الرجوع إلى توثيق text-indent في موسوعة حسوب لمزيد من التفصيل.

6.4.3 زخرفة النص

تحدد الخاصية `text-decoration` شكل خطوط الزخرفة (decorative lines) المستخدمة على النص؛ وهي خاصيةٌ مختصرة تضبط كل من الخاصيات:

```
text-decoration-line
text-decoration-color
text-decoration-style
```

انظر المثال التالي:

```
h1 { text-decoration: none; } /* لا توجد زخرفة */
h2 { text-decoration: overline; } /* إضافة خط فوق النص */
h3 { text-decoration: line-through; } /* إضافة خط يمر عبر النص */
h4 { text-decoration: underline; } /* إضافة خط تحت النص */
/* الشيفرتان أدناه متكافئتان */
.title {
```

```
    text-decoration: underline dotted blue;
}

.title {
    text-decoration-style: dotted;
    text-decoration-line: underline;
    text-decoration-color: blue;
}
```

انتبه إلى أن الخاصيات التالية مدعومة فقط في متصفح Firefox، ولكن يُمكن استخدام الصـياغة المختزلـة في بقية المتصفحات:

```
text-decoration-line
text-decoration-color
text-decoration-style
```

6.4.4 التباعد بين الكلمات

تحدد الخاصية `word-spacing` التباعد بين الكلمات والوسوم، والقيم المُتاحة التي تأخذها:

- رقم صحيح موجب أو سالب يحدد المسافة بإحدى الوحدات (px أو em أو غيرها من وحدات CSS).
- نسبة مئوية.
- الكلمة المحجوزة `normal`.
- الكلمة المحجوزة `inherit`.

مثال:

- ملف HTML

```
  <p>
    <span class="normal">This is an example, showing the effect of
"word-spacing".</span><br>
    <span class="narrow">This is an example, showing the effect of
"word-spacing".</span><br>
    <span class="extensive">This is an example, showing the effect of
"word-spacing".</span><br>
  </p>
```

- ملف CSS

```
.normal { word-spacing: normal; }
.narrow { word-spacing: -3px; }
.extensive { word-spacing: 10px; }
```

اطلع على تجربة حية للمثال أعلاه على JSFiddle.

يمكنك الرجوع إلى توثيق word-spacing في موسوعة حسوب لمزيد من التفصيل.

6.4.5 تقسيم النص إلى أعمدة

يُمكن استخدام الخاصية column-count لفصل النص لعدد من الأعمدة.

مثال:

```
<div id="multi-columns">Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud

exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in

reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
pariatur. Excepteur sint

occaecat cupidatat non proident, sunt in culpa qui officia deserunt
mollit anim id est

laborum</div>


<style>
.multi-columns {
    -moz-column-count: 2;
    -webkit-column-count: 2;
    column-count: 2;
}
</style>
```

النتيجة:

Stack Overflow is a privately held website, the flagship site of the Stack Exchange Network,[4][5][6] created in 2008 by Jeff Atwood and Joel Spolsky.[7][8] It was created to be a more open alternative to earlier Q&A sites such as Experts-Exchange. The name for the website was chosen by voting in April 2008 by readers of Coding Horror, Atwood's popular programming blog.

The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki or Digg.[13] Users of Stack Overflow can earn reputation points and "badges"; for example, a person is awarded 10 reputation points for receiving an "up" vote on an answer given to a question, and can receive badges for their valued contributions,[14] which represents a kind of gamification of the traditional Q&A site or forum. All user-generated content is licensed under a Creative Commons Attribute-ShareAlike license.

تُحدد الخاصية `column-width` أقل عرض ممكن للعمود.

مثال:

```
<div id="multi-columns">

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut

labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris

nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit

esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt

in culpa qui officia deserunt mollit anim id est laborumGoalKicker.com
- CSS Notes for Professionals 194

</div>


<style>

.multi-columns {

    -moz-column-width: 100px;

    -webkit-column-width: 100px;

    column-width: 100px;

}

</style>
```

النتيجة:

Stack Overflow is a privately held website, the flagship site of the Stack Exchange Network,[4][5] [6] created in 2008 by Jeff Atwood and Joel Spolsky.[7][8] It was created to be a more open alternative to earlier Q&A sites such as Experts-Exchange. The name for the website was chosen by voting in April 2008 by readers of Coding Horror, Atwood's popular programming blog.

The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki or Digg.[13] Users of Stack Overflow can earn reputation points and "badges"; for example, a person is awarded 10 reputation points for receiving an "up" vote on an answer given to a question, and can receive badges for their valued contributions,[14] which represents a kind of gamification of the traditional Q&A site or forum. All user-generated content is licensed under a Creative Commons Attribute-ShareAlike license.

انتبه، إذا لم تُعرَّف الخاصية `column-count` سينشئ المتصفح اكبر عدد ممكن من الأعمدة.

مثال:

```
<div class="content">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim
ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl
ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in
hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu
feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui
blandit praesent luptatum zzril delenit augue duis dolore te feugiat nulla
facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil
imperdiet doming id quod mazim placerat facer possim assum.
</div>
```

```
<style>
content {
    -webkit-column-count: 3; /* Chrome, Safari, Opera */
    -moz-column-count: 3; /* Firefox */
    column-count: 3;
}
</style>
```

مثال:

```
<section>
<p>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
nonumy eirmod tempor
invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
At vero eos et accusam et
justo duo dolores et ea rebum.</p>
<p> Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum
dolor sit amet. Lorem
ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore
et dolore magna aliquyam erat, sed diam voluptua. At vero eos et
accusam et justo duo dolores et ea
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
ipsum dolor sit amet.</p>
<p>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
nonumy eirmod tempor
invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
At vero eos et accusam et
justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea
takimata sanctus est Lorem ipsum
dolor sit amet.</p>
</section>

<style>
section {
    columns: 3;
    column-gap: 40px;
    column-rule: 2px solid gray;
}
</style>
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

6.4.6 مؤشر إدخال النص

تحدد الخاصية `caret-color` لون مؤشر الإدخال النصي (وهو العلامة التي تظهر للإشارة إلى موضع الحرف النصي الذي سيدخله المستخدم). الشكل الافتراضي لمؤشر الإدخال هو خط رأسي يومض لتسهيل ملاحظته، ويكون لونه الافتراضي هو الأسود، لكن يمكن تعديله باستخدام هذه الخاصية.

مثال:

```
<style>
    #example {
        caret-color: red;
    }
</style>

<input id="example" />
```

يمكنك الرجوع إلى توثيق caret-color في موسوعة حسوب لمزيد من التفصيل.

6.5 تنسيق ألوان النصوص

تستخدم خاصية `color` لضبط القيمة اللونية الأمامية foreground للمحتوى النصي للعنصر والزخرفة النصية `text-decoration`، وتضبط أيضًا قيمة الكلمة المحجوزة `currentColor`، والتي يمكن أن تستخدم كقيمة غير مباشرة في الخاصيات الأخرى التي لا تأخذ لونها من الخاصية `color` مباشرةً (مثل الخاصية `border-color`).

6.5.1 استخدام currentColor

مثال:

```
div {
    color: red;
    border: 5px solid currentColor;
    box-shadow: 0 0 5px currentColor;
}
```

في المثال أعلاه، استُخدمت الكلمة المحجوزة `currentColor` لإعطاء الإطار نفس اللون المحسوب computed color للعنصر (اللون الأحمر).

```
div {
    color: blue;
    border: 3px solid currentColor;
    color: green;
}
```

بما أن `currentColor` تأخذ القيمة المحسوبة computed value للخاصية، فإنَّها في هذا المثال ستأخذ القيمة `green`، ويكون لون الإطار الناتج أخضر.

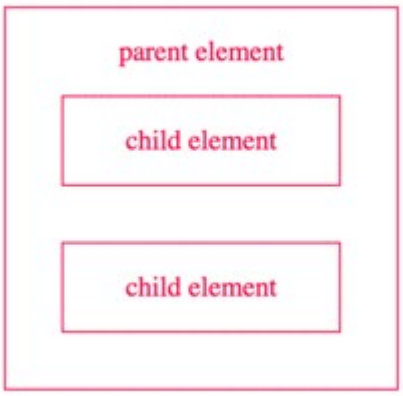
هنالك حالات يرث الابن اللون من العنصر الأب، في المثال أدناه تَرِث الخاصية `currentColor` اللون الأزرق من العنصر الأب.

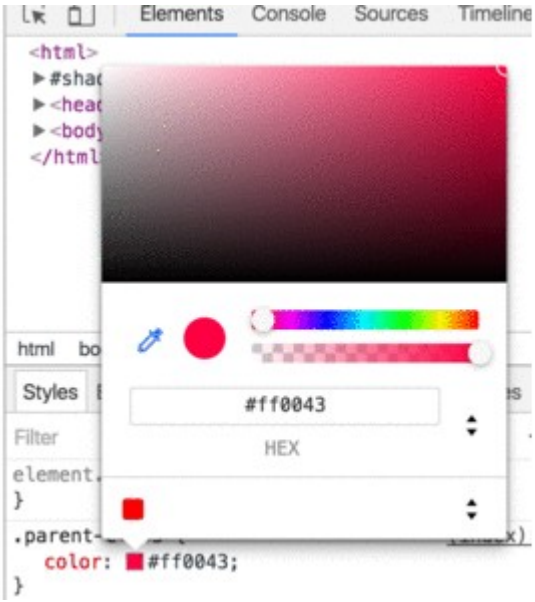
```
.parent-class {
    color: blue;
}

.parent-class .child-class {
    border-color: currentColor;
}
```

يُمكن استخدام `currentColor` لتوريث اللون للخواص التي لا ترثه عادةً مثل الخاصية `background-color`، يوضح المثال التالي كيفية ذلك:

```
.parent-class {
    color: blue;
}
.parent-class .child-class {
    background-color: currentColor;
}
```

النتيجة (شاهد الصورة متحركة بالضغط عليها):





6.5.2 الكلمات المحجوزة للألوان

تستخدم العديد من المتصفحات كلمات محجوزة لتعريف الألوان

```
.some-class {
    color: blue;
}
```

> الكلمات المحجوزة في CSS غير حساسة لحالة الأحرف، فالقيم blue، وBlue، وBLUE كلها تشير للون الست عشري #0000FF.

إن أردت الاطلاع على قائمة كل الكلمات المحجوزة للألوان، فارجع إلى قسم الكلمات المحجـوزة للألـوان من توثيق اللون في CSS في موسوعة حسوب.

6.5.3 القيم الست عشرية للألوان

يُمكن تمثيل الألـوان في CSS عن طريـق ثلاث أزواج من القيم السـت عشـرية، تُمثـل قيم الألـوان الأحمـر، والأخضـــر، والأزرق المكونـــة للـــون، ويأخـــذ كـــل زوج قيمـــة تـــتراوح بين 00 إلى FF (أو بين 0 و 255 في النظام العشري).

عدد الألوان التي يُمكن تمثيلها باستخدام النظام الست عشري هو 256^3 أي 16,777,216 لون.

الصياغة:

```
color: #rrggbb;
color: #rgb
```

حيث:

- rr: قيمة بين 00 و FF، تحدد كمية اللون الأحمر.
- gg: قيمة بين 00 و FF، تحدد كمية اللون الأخضر.
- bb: قيمة بين 00 و FF، تحدد كمية اللون الأزرق.

مثال:

```
.some-class {
    color: #0000FF;
}

.also-blue {
    color: #00F;
}
```

القيم الست عشرية للألوان غير حساسة لحالة الأحرف، فالقيمتان 0000FF و #0000ff تشيران لنفس اللون.

مصادر إضافية:

- النظام الست عشري
- القيم الرقمية للألوان

6.5.4 الدالة rgb

ترميز rgb هو المكافئ العشري للترميز الست عشـري للألـوان، ويُعَبِّر عن الألـوان الأحمـر والأخضـر والأزرق بقيم رقميـة بين 0 و 255 (تمثـل المكافئـات العشـرية للقيم 00 و FF على التـوالي)، أو بنسـب مئويـة بين 0% و 100%.

```
.some-class {
    color: rgb(0, 0, 255); /* blue color */
}

.also-blue {
    color: rgb(0%, 0%, 100%); /* blue color */
}
```

الصيغة العامة:

```
rgb(<red>, <green>, <blue>);
```

6.5.5 الدالة rgba

ترميز rgba هو نفس ترميز rgb مع إضافة قيمة أخيرة تُمثل نسبة تعتيم اللون opacity، وتأخذ قيم بين صفر (شفاف بالكامل) وواحد (معتم بالكامل).

```
.red {
    color: rgba(255, 0, 0, 1); /* أحمر معتم بالكامل */
}

.red-50p {
    color: rgba(255, 0, 0, 0.5); /* 50 أحمر شفاف بنسبة% */
}
```

الصيغة العامة:

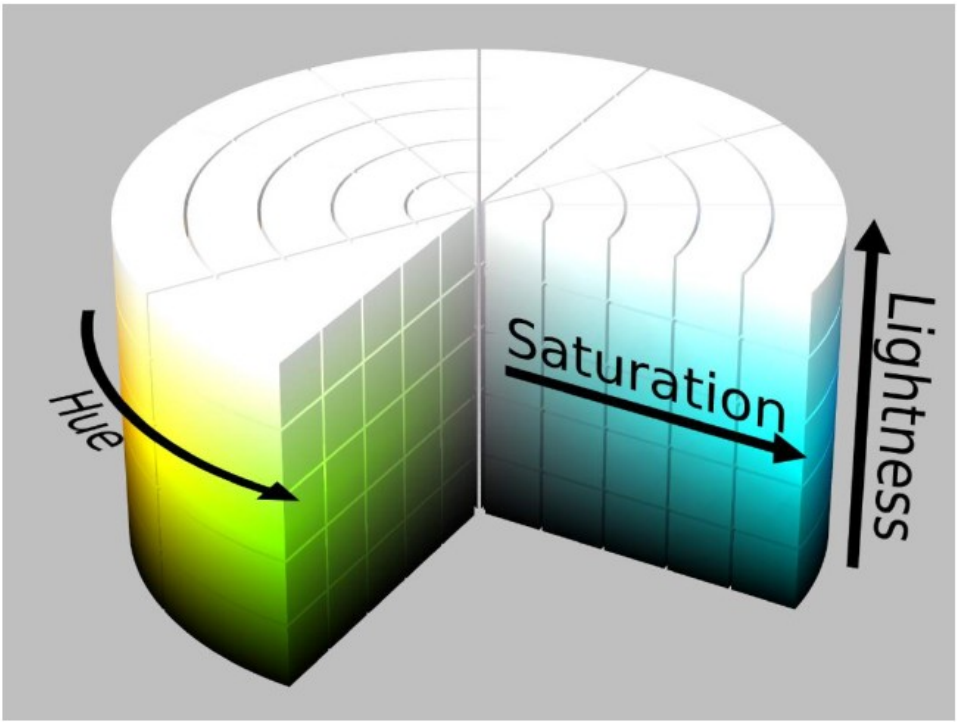
```
rgba(<red>, <green>, <blue>, <opacity>);
```

6.5.6 الدالة hsl

ترمز hsl للكلمات صبغة (hue) وتستعمل لتحديد اللون، إشباع (saturation) وتستعمل لتحديد تركيز اللون، إضاءة (lightness) وتستعمل لتحديد كمية اللون الأبيض في اللون، تحدد قيمة الصبغة بزاوية بين صفر و 360 درجة، بينما تحدد قيم الإشباع والإضاءة بنسب مئوية.

```
p {
    color: hsl(240, 100%, 50%); /* Blue */
}
```

الصورة أدناه توضح الأسطوانة اللونية المستخدمة في ترميز `rgb()`.



الصيغة العامة:

```
color: hsl(<hue>, <saturation>%, <lightness>%);
```

يوضح الجدول التالي وصف القيم:

القيمة	الوصف
`<hue>`	تُحدد زاوية بين صفر و360 درجة، حيث تمثل الزاوية 0 اللون الأحمر، و60 للون الأصفر، و120 للون الأخضر، و180 للون الأزرق السماوي، و240 للأزرق، و300 للبنفسجي، و360 للأحمر أيضًا.
`<saturation>`	تُحدد درجة تشَبُّع اللون بنسبة مئوية، حيث 0% تعني أن اللون غير مشبع (تدرج رمادي)، و100% تعني أن اللون مشبع بالكامل (لون واضح).
`<lightness>`	تُحدد إضاءة اللون بنسبة مئوية، حيث تشير 0% للون الأسود (مظلم بالكامل)، وتشير 100% للون الأبيض.

6.5.7 الدالة hsla

ترميز hsla هو نفس ترميز hsl مع إضافة قيمة أخيرة تُمثل نسبة تعتيم اللون opacity، وتأخذ قيم بين صفر (شفاف بالكامل) وواحد (معتم بالكامل).

```
hsla(240, 100%, 50%, 0) /* شفاف بالكامل */
hsla(240, 100%, 50%, 0.5) /* 50 شفاف بنسبة% */
hsla(240, 100%, 50%, 1) /* معتم بالكامل */
```

الصيغة العامة:

```
hsla(<hue>, <saturation>, <lightness>, <opacity>);
```

6.5.8 التعتيم

تستخدم الخاصية `opacity` لتحديد درجة تعتيم العنصر وتأخذ قيم رقمية بين صفر (شفاف بالكامل أو غير مرئي) وواحد (معتم بالكامل).

مثال:

```
<div style="opacity: 0.8;">
    هذا عنصر شفاف جزئيًا
</div>
```

حيث:

قيمة الخاصية	الوصف
1.0	معتم بالكامل.
0.75	شفاف بنسبة 25%.
0.5	شفاف بنسبة 50%.
0.25	شفاف بنسبة 25%.
0.0	شفاف بنسبة 100%.

6.6 شكل المؤشر cursor style

تُحدد الخاصية `cursor` شكل مؤشر الفأرة الذي سيُعرَض عندما تمر الفأرة فوق العنصر.

الصيغة العامة:

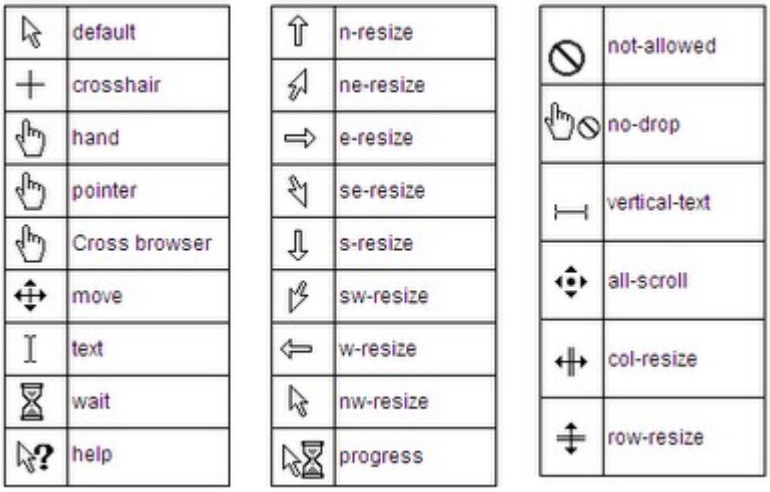
```
cursor: value;
```

يوضح الجدول التالي القيم التي تأخذها الخاصية `cursor`:

القيمة	الوصف
`none`	تُخفي مؤشر الفأرة.
`auto`	سيُحدِّد المتصفح شكل المؤشر الأفضل للعنصر.
`help`	يشير المؤشر إلى توفر معلومات مساعدة.
`wait`	البرنامج يجري مهمة في الخلفية، ولا يمكن أن يتفاعل المستخدم معه.

القيمة	الوصف
`move`	يشير المؤشر إلى أنَّ شيئًا ما تحَرَّك.
`pointer`	يمكن التفاعل مع العنصر بالضغط عليه، كما في الروابط والأزرار.

الأشكال المتاحة:



6.6.1 الخاصية pointer-event

تُحدد الخاصية `pointer-event` كيف سيتعامل العنصر مع مؤشر الفأرة. مثال:

```
.disabled {
    pointer-events: none;
}
```

القيمة `none` توقِف كل أحداث الفأرة مثال الضغط والتمرير وغيرها.

مصادر أجنبية إضافية:

- CSS tricks
- توثيق MDN
- davidwalsh

هل تطمح لبيع منتجاتك الرقمية عبر الإنترنت؟

استثمر مهاراتك التقنية وأطلق منتجًا رقميًا
يحقق لك دخلًا عبر بيعه على متجر بيكاليكا

أطلق منتجك الآن

7. تنسيق الخلفيات Backgrounds

تُمكنك CSS من اختيار ألوان، أو تدرجات لونية، أو صور كخلفيات للعناصر، والتحكم في حجمها، وموضعها، وعدد مرات تكرارها.

7.1 إضافة الألوان كخلفيات للعناصر

تُستعمل الخاصية `background-color` لإضافة خلفية بلون معين للعنصر، وتقبل القيم `transparent` و `inherit` و `initial` أيضًا، ويمكن تطبيقها على العناصر و العناصر الزائفة `::first-letter` و `::first-line`.

- `transparent`: تجعل خلفية العنصر شفافة، وهي القيمة الافتراضية للخاصية.
- `inherit`: تَرِث قيمة الخاصية من العنصر الأب.
- `initial`: تُرجع القيمة الابتدائية للخاصية.

تُعرَّف الألوان في CSS بعدد من الطُرق نذكرها تاليًا.

7.1.1 الكلمات المحجوزة للألوان

```
<style>
    div {
        background-color: red;
    }
</style>
```

```
<div>This will have a red background</div>
```

7.1.2 القيم الست عشرية للألوان

تُستعمل القيم الست عشرية لتمثيل مكونات الألوان (أو ما يُعرف بترميز RGB) بالنظام الست عشري. فمثلًا يكون اللون `#ff0000` هو اللون الأحمر، حيث يُمثل بالحرفين `ff` اللذان يمثلان المقابل الست عشري للرقم 256.

يُمكن تقسم الترميز الست عشري للألوان إلى ثلاثة أقسام، كل منها يتكون من حرفين ويُمثل أحد الألوان الأحمر أو الأخضر أو الأزرق (على الترتيب من اليسار إلى اليمين)، وفي حال كانت الحروف الممثلة لكل لون متشابهه، يمكن اختصار الترميز الست عشري إلى ثلاث أحرف، فعلى سبيل المثال يُمكن اختصار اللون `#ff0000` إلى `#f00`.

انتبه إلى أن الترميز الست عشري غير حساس لحالة الأحرف.

```
body {
    background-color: #de1205; /* red */
}

.main {
    background-color: #00f; /* blue */
}
```

7.1.3 ترميز RGB و RGBa

ترمُز RGB إلى الألوان الأحمر والأخضر والأزرق (على الترتيب من اليسار إلى اليمين)، وتستخدم ثلاث قيم تُمثَّل كل منها القيمة العشرية لأحد الألوان الثلاثة، وتتراوح هذه القيم بين صفر و 255.

يُمكِّن ترميز RGBa من إضافة مُعامل أخير يُسمى معامل ألفا، ويُحدد درجة تعتيم/شفافية العنصر، وتتراوح قيمته بين 0.0 و 1.0.

```
header {
    background-color: rgba(0, 0, 0); /* black */
}

footer {
    background-color: rgba(0, 0, 0, 0.5); /* black with 50% opacity */
}
```

7.1.4 ترميز HSL و HSLa

ترمز hsl للكلمات صبغة hue وتستعمل لتحديد اللون، إشباع saturation وتستعمل لتحديد تركيز اللون، إضاءة lightness وتستعمل لتحديد كمية اللون الأبيض في اللون، وتحدد قيمة الصبغة بزاوية بين صفر و 360 درجة، بينما تحدد قيم الإشباع والإضاءة بنسب مئوية.

يُمكِّن ترميز HSLa من إضافة مُعامل أخير يُسمى معامل ألفا، ويُحدد درجة تعتيم/شفافية العنصر، وتتراوح قيمته بين 0.0 و 1.0.

```
li a {
    background-color: hsl(120, 100%, 50%); /* أخضر */
}

#p1 {
    background-color: hsla(120, 100%, 50%, 0.3); /* أخضر مع شفافية */
}
```

7.2 استخدام التدرجات اللونية كخلفيات للعناصر

أُضيفت التدرجات اللونية كإحدى أنواع الصور في CSS3، ويُمكن استعمالها كقيمة للخاصية `background-image` أو الخاصية المُختزلة `background`. ويوجد نوعين من التدرجات اللونية هما التدرجات الخطية linear و التدرجات الدائرية radial، وكلٌّ منهما يمكن أن يكون متكرر أو غير متكرر.

الدوال التي يمكن استعمالها لتوليد تدرجات:

- `linear-gradient()`
- `repeating-linear-gradient()`
- `radial-gradient()`
- `repeating-radial-gradient()`

7.2.1 التدرج الخطي: الدالة linear-gradient

الصيغة العامة:

```
background: linear-gradient(<direction>?, <color-stop-1>, <color-stop-
2>, ...);
```

يشرح الجدول التالي القيم التي تأخذها:

القيمة	الوصف
<direction>	يُمكن أن يأخذ قيم مثل to top، أو to bottom، أو to right، أو to left، أو تأخذ زاوية تبدأ من الأعلى وتدور في اتجاه عقارب الساعة، وتُقاس بالوحدات deg، أو grad، أو rad، أو turn. وإذا لم تُحدد هذه القيمة يكون التدرج من أعلى لأسفل.
<color-stop-list>	تُحدد قائمة من الألوان، ويُمكن إتباع كل لون بنسبة مئوية أو مسافة تُحدد الموضع الذي سيُعرض فيه اللون (اختاري)، مثل: yellow 10% rgba(0,0,0, 0.5) 40px, #fff 100%

مثال على إنشاء تدرج لوني يتجه من اليمين إلى اليسار ويتدرج بين اللونين الأحمر والأزرق:

```
.linear-gradient {
    background-color: linear-gradient(to left, red, blue); /* you can
also use 270deg */
}
```

إنشاء تدرج لوني يتجه من أسفل اليمين إلى أعلى اليسار:

```
.diagonal-linear-gradient {
    background-color: linear-gradient(to left top, red, yellow 10%);
}
```

يُمكنك تحديد أي عدد من النقاط اللونية في التدرج اللوني والفصل بينها بفاصلة، كما هو موضح في المثال التالي:

```
.linear-gradient-rainbow {
    background-color: linear-gradient(to left, red, orange, yellow,
green ,blue, indigo, violet);
}
```

7.2.2 التدرج الدائري: الدالة radial-gradient

مثال:

```
.radial-gradient-simple {
    background: radial-gradient(red, blue);
}


.radial-gradient {
    background: radial-gradient(circle farthest-corner at top left,
red, blue);
```

```
}
```

حيث:

- `circle`: شكل التدرج اللوني، ويمكن أن تكون `circle` أو `ellipse` وهي القيمة الافتراضية.
- `farthest-corner`: كلمة محجوزة تُحدد حجم الشكل النهائي، ويمكن أن تكون `closest-side` أو `farthest-side` أو `closest-corner` أو `farthest-corner`.
- `top left`: تُحدد موضع مركز التدرج اللوني.

7.2.3 التدرجات المُتكررة Repeating Gradients

تأخذ التدرجات اللونية المتكررة نفس معاملات التدرجات اللونية العادية، ولكنها تُكرر التدرج اللوني على خلفية العنصر.

```
.bullseye {
    background: repeating-radial-gradient(red, red 10%, white 10%,
white 20%);
}


.warning {
    background: repeating-linear-gradient(-45deg, yellow, yellow 10%,
black 10%, black 20%);
}
```

حيث:

- `-45deg`: زاوية تبدأ من الأعلى وتدور في اتجاه عقارب الساعة، وتُقاس بالوحدات `deg` أو `grad` أو `rad` أو `turn`.
- `top left`: اتجاه التدرج اللوني، وتأخذ `top` أو `bottom` على المحور Y والقيمة `left` أو `right` على المحور X.
- `yellow 10%`: قائمة الألوان، ويُمكن إتباع كل لون بنسبة مئوية أو مسافة تُحدد الموضع الذي سيُعرض فيه اللون (اختاري).

7.3 استخدام الصور كخلفيات للعناصر

تُستخدم الخاصية `background-image` لوضع صورة كخلفية للعنصر المُحدد.

```
.myClass {
    background-image: url('/path/to/image.jpg');
}
```

ويُمكن تحديد عدد من الصور والفصل بينها بفاصلة، وتكون النتيجة أن تظهر الصور مترادفة حسب ترتيب كتابتها في الشيفرة.

```
.myClass {
    background-image: url('/path/to/image.jpg'),
                      url('/path/to/image2.jpg');
}
```

حيث:

- `url(...)`: تُحدد مسار الصورة المُراد استخدامها.
- `none`: تحذف صورة الخلفية.
- `initial`: تُرجع القيمة الافتراضية للخاصية.
- `inherit`: تَرِث قيمة الخاصية من العنصر الأب.

بعض الخاصيات التي تُستخدم مع الخاصية `background-image`:

```
background-size: xpx ypx | x% y%;
background-repeat: no-repeat | repeat | repeat-x | repeat-y;
background-position: left offset (px/%) right offset (px/%) | center
center | left top | right bottom;
```

7.4 حجم الخلفية Background Size

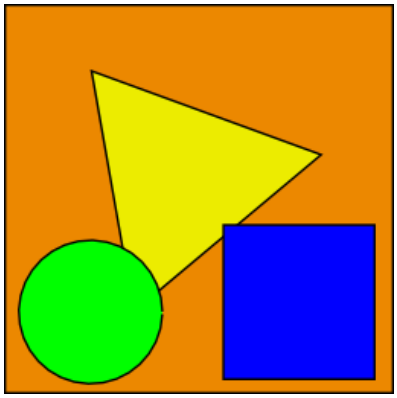
تُستخدم الخاصية background-size للتحكم في تكبير وتصغير صورة الخلفية، وتأخذ قيمة واحدة تُحدد التكبير أو التصغير في الاتجاهين الأفقي والرأسي، أو تأخذ قيمتين تحددان التكبير أو التصغير في كل اتجاه على حدة.

تُحافظ القيمة `auto` للخاصية `background-size` على نسبة أبعاد الصورة (نسبة الارتفاع إلى العرض)، فعلى سبيل المثال إذا كان لدينا صورة بحجم 256px * 256px، فإن جميع القواعد أدناه متكافئة وتؤدي لعرض الصورة بارتفاع وعرض 50 بكسل.

```
background-size: 50px;
```

```
background-size: 50px auto;
background-size: auto 50px;
background-size: 50px 50px;
```

الصورة الابتدائية ذات الأبعاد 256px × 265px:

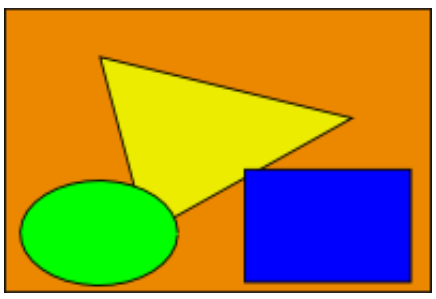


النتيجة صورة بأبعاد 50px × 50px محتواة في خلفية العنصر:

ويُمكن أيضًا استعمال نسب مئوية لتحديد التكبير أو التصغير للصورة.

```
#withbackground {
    background-image: url('to/some/background.png');
    background-size: 100% 66%;
    width: 200px;
    height: 200px;
    padding: 0;
    margin: 0;
}
```

النتيجة:

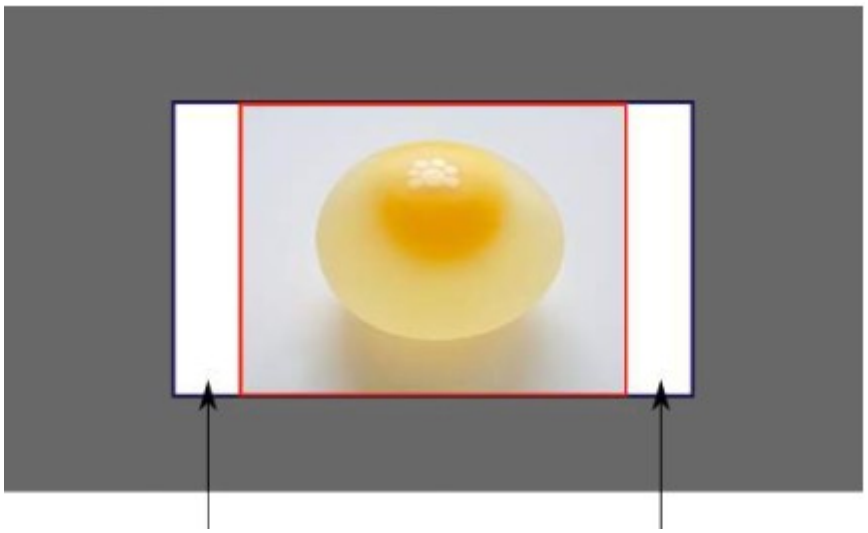


يعتمد سلوك الخاصية `background-size` على الخاصية background-origin.

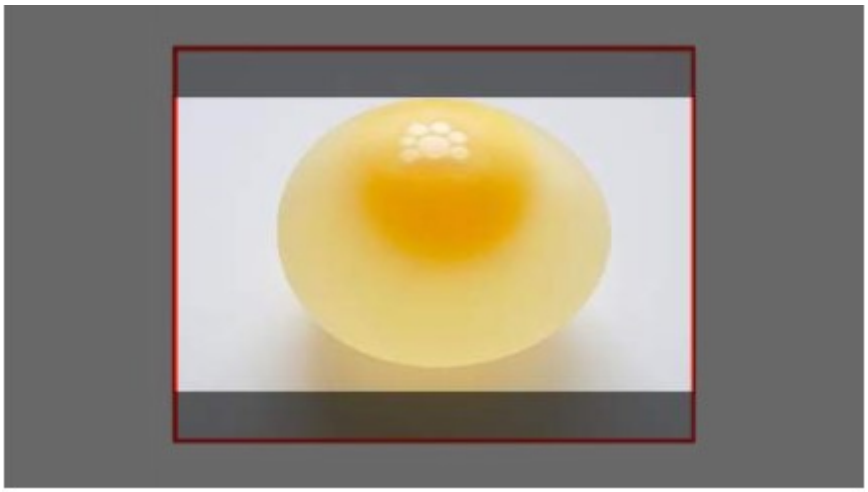
ا. المحافظة على نسبة أبعاد الصورة

للمحافظة على نسبة أبعاد الصورة عند استخدامها لتغطية عنصر أو وضعها داخله يجب استخدام القيم `cover` أو `contain` للخاصية `background-size`.

افترض أن المساحة البيضاء في الصورة أعلاه تُمثل شاشة العرض الخاصة بك، استخدام القيمة `contain` للخاصية `background-size` سيؤدي إلى تكبير أو تصغير الصورة بحيث يُطابق أحد أبعادها أحد بُعدي الصندوق مع المحافظة على نسبة أبعاد الصورة.



أما القيمة `cover` فستؤدي إلى تكبير الصورة بحيث تطابق أبعادها أبعاد الصندوق، مع قص الأجزاء الزائدة عن مساحة الصندوق للمحافظة على نسبة أبعاد الصورة.



مثال:

- ملف CSS

```
div > div {
    background-image: url(http://i.stack.imgur.com/r5CAq.jpg);
    background-repeat: no-repeat;
    background-position: center center;
    background-color: #ccc;
    border: 1px solid;
    width: 20em;
    height: 10em;
}
div.contain {
    background-size: contain;
}
div.cover {
    background-size: cover;
}

/*******************************************
Additional styles for the explanation boxes
*******************************************/

div > div {
    margin: 0 1ex 1ex 0;
    float: left;
```

```
  }
  div + div {
      clear: both;
      border-top: 1px dashed silver;
      padding-top:1ex;
  }
  div > div::after {
      background-color: #000;
      color: #fefefe
      margin: 1ex;
      padding: 1ex;
      opacity: 0.8;
      display: block;
      width: 10ex;
      font-size: 0.7em;
      content: attr(class);
  }
```

- ملف HTML

```
  <div>
      <div class="contain"></div>
      <p>Note the grey background. The image does not cover the whole
region, but it's fully
      <em>contained</em>.
      </p>
  </div>


  <div>
      <div class="cover"></div>
      <p>Note the ducks/geese at the bottom of the image. Most of the
water is cut, as well as a part
      of the sky. You don't see the complete image anymore, but
neither do you see any background color;
      the image <em>covers</em> all of the <code><div></code>.</p>
  </div>
```

النتيجة:

Note the grey background. The image does not cover the whole region, but it's fully *contained*.

Note the ducks/geese at the bottom of the image. Most of the water is cut, as well as a part of the sky. You don't see the complete image anymore, but neither do you see any background color; the image *covers* all of the `<div>`.

7.4.2 نقوش الصور Image Sprites

نقوش الصور هي مجموعة من الصور مُخزَّنة في صورة واحدة، ويمكن استعمال كل صوره منها على حدة.

مثال:

الصورة أعلاه تحتوي على عدد من الصـور، ويمكن اسـتعمال كـل واحـدة منهـا منفـردة كمـا هـو موضـح في الشيفرة التالية:

- ملف HTML

```
    <div class="icon icon1"></div>
    <div class="icon icon2"></div>
    <div class="icon icon3"></div>
```

- ملف CSS

```
.icon {
    background: url("icons-sprite.png");
    display: inline-block;
    height: 20px;
    width: 20px;
}
.icon1 {
    background-position: 0px 0px;
}
.icon2 {
    background-position: -20px 0px;
}
.icon3 {
    background-position: -40px 0px;
}
```

تُفصل الصور عن طريق تحديد مواقعها بالخاصية background-position.

7.5 موضع الخلفية Background Position

تُستخدم الخاصية background-position لتحديد نقطة بداية الخلفية سواء كانت صورة أو تدرج لوني، ويوضح الجدول التالي القيم التي تأخذها:

الوحدة	الوصف
%value% value	تُحدد الإزاحة الأفقية لصورة الخلفية كنسبة من عرض الصورة، والإزاحة الرأسية كنسبة من ارتفاعها.
valuepx valuepx	تُحدد الإزاحات الأفقية والرأسية بالبكسلات، وتُقاس المسافة من أعلى ويسار الصورة.

مثال:

```
.myClass {
    background-image: url('path/to/image.jpg');
    background-position: 50% 50%;
}
```

بالإضافة للخاصية المختزلة background-position والتي تُحدد الإزاحتان الأفقية والرأسية معًا، يُمكن استخدام:

- الخاصية `background-position-x` لتحديد الإزاحة الأفقية
- الخاصية `background-position-y` لتحديد الإزاحة الرأسية

7.5.1 الخاصية background-origin

تُحدد الخاصية `background-origin` موضع صورة الخلفية، وفي حال كانت قيمة الخاصية `background-attachment` هي `fixed` فلن يكون لهذه الخاصية أي تأثير، والقيمة الافتراضية لها هي `padding-box`.

القيمة	الوصف
`padding-box`	يُحدد الموضع نسبةً لصندوق الحواشي.
`border-box`	يُحدد الموضع نسبةً لصندوق الإطارات.
`context-box`	يُحدد الموضع نسبةً لصندوق المحتوى.
`initial`	تُرجع القيمة الابتدائية للخاصية.
`inherit`	ترث قيمة الخاصية من العنصر الأب.

مثال:

- ملف HTML

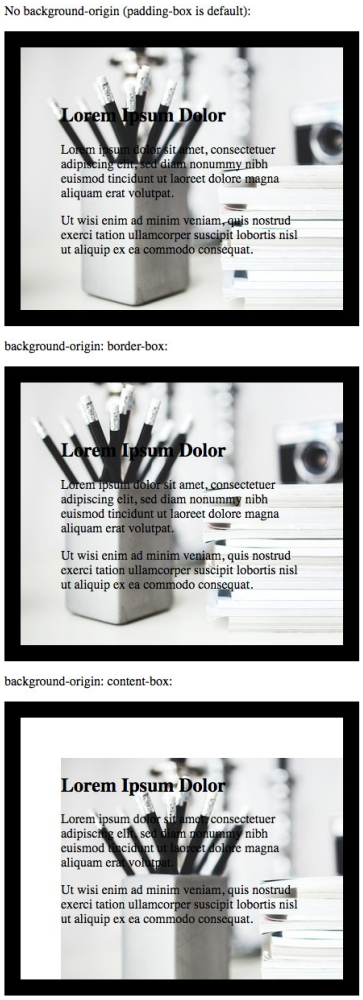
```
  <p>No background-origin (padding-box is default):</p>
  <div class="example example1">
      <h2>Lorem Ipsum Dolor</h2>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
diam nonummy nibh euismod
      tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
      <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut
      aliquip ex ea commodo consequat.</p>
  </div>

  <p>background-origin: border-box:</p>
  <div class="example example2">
      <h2>Lorem Ipsum Dolor</h2>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
diam nonummy nibh euismod
      tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
```

```
      <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut
      aliquip ex ea commodo consequat.</p>
  </div>


  <p>background-origin: content-box:</p>
  <div class="example example3">
      <h2>Lorem Ipsum Dolor</h2>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
diam nonummy nibh euismod
      tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
      <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut
      aliquip ex ea commodo consequat.</p>
  </div>
```

- ملف CSS

```
  .example {
      width: 300px;
      border: 20px solid black;
      padding: 50px;
      background: url(https://static.pexels.com/photos/6440/magazines-
desk-work-workspace-medium.jpg);
      background-repeat: no-repeat;
  }
  .example1 {}
  .example2 { background-origin: border-box; }
  .example3 { background-origin: content-box; }
```

النتيجة:



للمزيد ارجع إلى توثيق background-origin في موسوعة حسوب.

7.5.2 إضافة صور مُتعددة للخلفية

يُمكن إضافة عدد من الصور كخلفيات للعنصر، وتوضع الصور فـوق بعضـها البعض بنفس تـرتيب ظهورهـا في الشيفرة كما هو موضح في المثال أدناه

```
#mydiv {
    background-image: url(img_1.png), /* top image */
                      url(img_2.png), /* middle image */
                      url(img_3.png); /* bottom image */
    background-position: right bottom,
                         left top,
                         right top;
    background-repeat: no-repeat,
                       repeat,
                       no-repeat;
```

```
}
```

ومن الممكن أيضًا استخدام الصياغة المختزلة للخاصية كما هو موضح في المثال التالي:

```
#mydiv {
    background: url(img_1.ong) right bottom no-repeat,
                url(img_2.png) left top repeat,
                url(img_3.png) right top no-repeat;
}
```

يُمكنك أيضًا إضافة تدرج لوني مع الصورة:

```
#mydiv {
    background: url(image.png) right bottom no-repeat,
                linear-gradient(to bottom, #fff 0%, #000 100%);
}
```

اطَّلع على تجربة حية لهذه الأمثلة على JSFiddle.

7.5.3 الخاصية background-attachment

تُحدد الخاصية `background-attachment` ما إذا كانت صورة الخلفية ثابتة fixed أم أنها تتحرك مع بقية الصفحة، ويوضح الجدول التالي القيم التي تأخذها:

القيمة	الوصف
`scroll`	تتحرك صورة الخلفية مع العنصر.
`fixed`	صورة الخلفية ثابتة بالنسبة لشاشة العرض.
`local`	تتحرك صورة الخلفية مع محتوى العنصر.
`initial`	تُرجع القيمة الابتدائية للخاصية.
`inherit`	ترث قيمة الخاصية من العنصر الأب.

مثال:

```
body {
    background-image: url('img.jpg');
    background-attachment: fixed;
}
```

أمثلة:

- `background-attachment: scroll`

```
body {
    background-image: url('image.jpg');
    background-attachment: scroll;
}
```

- `background-attachment: fixed`

```
body {
    background-image: url('image.jpg');
    background-attachment: fixed;
}
```

- `background-attachment: local`

```
div {
    background-image: url('image.jpg');
    background-attachment: local;
}
```

7.5.4 الخاصية background-clip

تُحدد الخاصية `background-clip` أين ستتوقف خلفية العنصر على حدود العنصر، سواءً كـانت الخلفيـة لونًا `<color>` أو صورةً `<image>`.

القيمة	الوصف
border-box	ستمتد الخلفية إلى الحافة الخارجية للإطار (border)، لكنها ستقع تحته.
padding-box	ستمتد الخلفية إلى الحافة الخارجية لمنطقة الحواشي (padding)، ولن تُرسَم الخلفية تحت الإطار.
content-box	ستكون حدود الخلفية هي صندوق المحتوى (content box).
inherit	ترث قيمة الخاصية من العنصر الأب.

مثال:

- ملف HTML

```
<p>No background-origin (padding-box is default):</p>
<div class="example example1">
```

```
      <h2>Lorem Ipsum Dolor</h2>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
diam nonummy nibh euismod
      tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
      <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut
      aliquip ex ea commodo consequat.</p>
  </div>

  <p>background-origin: border-box:</p>
  <div class="example example2">
      <h2>Lorem Ipsum Dolor</h2>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
diam nonummy nibh euismod
      tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
      <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut
      aliquip ex ea commodo consequat.</p>
  </div>

  <p>background-origin: content-box:</p>
  <div class="example example3">
      <h2>Lorem Ipsum Dolor</h2>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
diam nonummy nibh euismod
      tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
      <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut
      aliquip ex ea commodo consequat.</p>
  </div>
```

- ملف CSS

```
  .example {
      width: 300px;
      border: 20px solid black;
      padding: 50px;
      background: url(https://static.pexels.com/photos/6440/magazines-
desk-work-workspace-medium.jpg);
```

```
        background-repeat: no-repeat;
    }
    .example1 {}
    .example2 { background-origin: border-box; }
    .example3 { background-origin: content-box; }
```

7.5.5 الخاصية background-repeat

تُحدد الخاصية `background-repeat` كيف ستتكرر صورة الخلفية؛ إذ يمكن أن تتكرر صورة الخلفية على المحور الأفقي، أو على المحور الرأسي، أو على كلا المحوري، ويمكن ألّا تتكرر أبدًا.

مثال على تكرار الخلفية على المحور الرأسي:

```
div {
    background-image: url("image.jpg");
    background-repeat: repeat-y;
}
```

النتيجة:

7.5.6 الخاصية background-blend-mode

تُحدد الخاصية `background-blend-mode` كيف تمتزج صور الخلفية مع بعضها بعضًا ومع لون الخلفية المُحدَّد للعنصر.

```
.my-div {
    width: 300px;
    height: 200px;
    background-size: 100%;
    background-repeat: no-repeat;
    background-image: linear-gradient(to right, black 0%,white 100%),
    url('https://static.pexels.com/photos/54624/strawberry-fruit-red-sweet-54624-medium.jpeg');
    background-blend-mode:saturation;
}
```

اطّلع على تجربة حيَّة لهذا المثال على JSFiddle.

لمزيد من المعلومات حول الخاصية `background-blend-mode` انظر موسوعة حسوب.

7.6 إضافة تأثير الشفافية على خلفية العنصر

إذا أضفت تأثير الشفافية على العنصر فستتأثر جميع العناصر الأبناء له بهذا التأثير، ولإضافة تأثير الشـفافية على الخلفية فقط يجب استخدام ترميز RGBa للألوان كما هو موضح في المثال أدناه

```
/* Fallback for web browsers that don't support RGBa */
background-color: rgb(0, 0, 0);

/* RGBa with 0.6 opacity */
background-color: rgba(0, 0, 0, 0.6);
```

7.7 الخاصية المختزلة background

يُمكن استخدام الخاصية المُختزلة `background` لتحديد خاصـية واحـدة أو عـدد من الخاصـيات المرتبطـة بالخلفيات.

الخاصية	الوصف	إصدار CSS
`background-image`	الصورة المُراد استخدامها كخلفية.	1+
`background-color`	اللون المُراد استخدامه كخلفية.	1+
`background-position`	موضع الخلفية.	1+
`background-size`	حجم صورة الخلفية.	3+
`background-repeat`	كيفية تكرار صورة الخلفية.	1+

الخاصية	الوصف	إصدار CSS
background-origin	مركز الخلفية.	3+
background-clip	تحدد كيف ستعرض الخلفية نسبةً لصندوق المحتوى أو صندوق الإطارات أو صندوق الحواشي.	3+
background-attachment	تُحدد ما إذا كانت الخلفية ستتحرك مع العنصر الحاوي لها، أم أنها ثابتة.	1+
initial	تُرجع القيمة الابتدائية للخاصية.	3+
inherit	ترث قيمة الخاصية من العنصر الأب	2+

الصيغة العامة:

```
background: [<background-image>] [<background-color>] [<background-position>]/[<background-size>]
[<background-repeat>] [<background-origin>] [<background-clip>] [<background-attachment>]
[<initial|inherit>];
```

جميع الخواص اختيارية، والترتيب غير مهم.

مثال على إضافة خلفية بلون أحمر للعنصر:

```
background: red;
```

مثال على استعمال الخاصية background-clip:

```
background: border-box red;
```

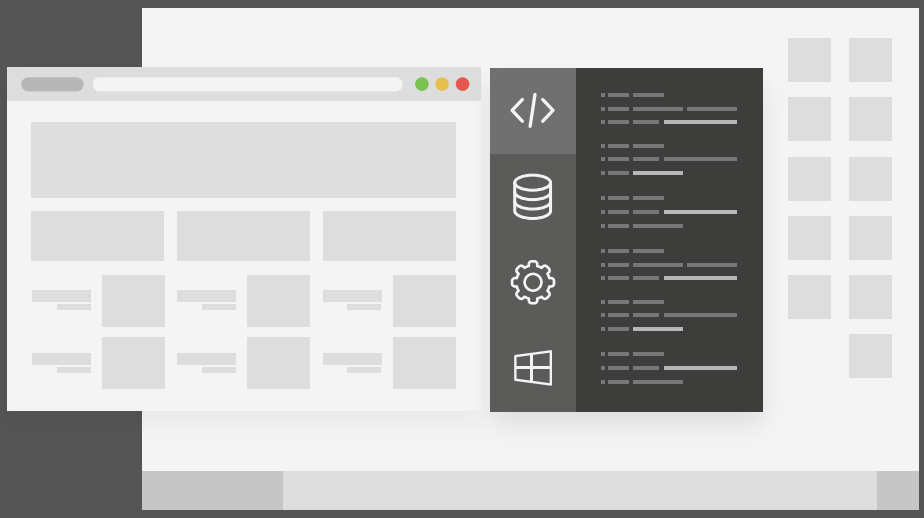
مثال على تحديد تكرار ومركز الخلفية:

```
background: no-repeat center url('somepng.png');
```

مثال على إضافة أكثر من خلفية للعنصر:

```
background: url('pattern.png') green;
background: #000 url(picture.png) top left / 00px auto no-repeat;
```

دورة علوم الحاسوب



دورة تدريبية متكاملة تضعك على بوابة الاحتراف
في تعلم أساسيات البرمجة وعلوم الحاسوب

التحق بالدورة الآن

8. تنسيق الصور

8.1 المُرشِّحات Filters

تُستخدم الخاصية `filter` لتطبيق تأثيرات مثل تغيير اللون وتأثير الضبابية على الصور والخلفيات والإطارات ويوضح الجدول التالي الدوال التي يمكن استخدامها معها.

الدالة	الوصف
`blur()`	تُطبِق تأثير الضبابية على الصورة المُحدَّدة.
`brightness(x)`	تُغيِّر سطوع الصورة.
`contrast(x)`	تُغيِّر تباين الصورة.
`dropshadow(h,v,x,y,z)`	تُطبِّق تأثير الظلال على الصورة.
`gray-scale(x)`	تُحَوِّل ألوان الصورة إلى التدرج الرمادي.
`hue-rotate(x)`	تُدَوِّر القيمة اللونية لجميع ألوان الصورة.
`invert(x)`	تعكس ألوان الصورة،
`opacity(x)`	تُطبِّق تأثير الشفافية على الصورة.
`saturate(x)`	تُغيِّر إشباع الصورة.
`sepia(x)`	تُحَوِّل ألوان الصورة إلى البني الداكن.

يجب استخدام البادئة `-webkit` لدعم خاصية المُرشِّحات `filter` في المتصفحات القديمة.

8.2 الدالة blur

مثال:

- ملف HTML

```
<img src="donald-duck.png" alt="donald duck" title="donald duck" />
```

- ملف CSS

```
img {
    -webkit-filter: blur(1px);
    filter: blur(1px);
}
```

النتيجة:

8.3 الدالة dropshadow

مثال:

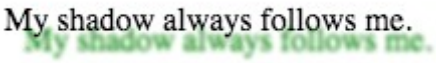
- ملف HTML

```
<p>
    My shadow always follows me.
</p>
```

- ملف CSS

```
p {
    -webkit-filter: drop-shadow(10px 10px 1px green);
    filter: drop-shadow(10px 10px 1px green);
}
```

النتيجة:



8.4 الدالة hue-rotate

مثال:

- ملف HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

- ملف CSS

```
img {
    -webkit-filter: hue-rotate(120deg);
    filter: hue-rotate(120deg);
}
```

النتيجة:

8.5 الدالة invert

مثال:

- ملف HTML

```
<div></div>
```

- ملف CSS

```
div {
    width: 100px;
    height: 100px;
    background-color: white;
    -webkit-filter: invert(100%);
    filter: invert(100%);
}
```

النتيجة:

8.6 إضافة مُرشِّحات متعددة

يُمكن تطبيق عدد من المُرشِّحات والفصل بينها بفاصلة.

مثال:

- ملف HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

- ملف CSS

```
img {
    -webkit-filter: brightness(200%) grayscale(100%) sepia(100%)
invert(100%);
```

```
        filter: brightness(200%) grayscale(100%) sepia(100%)
invert(100%);
  }
```

النتيجة:

8.7 القواطع Clipping والأقنعة Masking

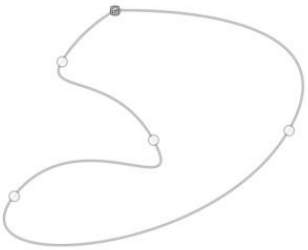
تستخدم خواص القطع `clip-path` والأقنعة `mask` لجعل أجزاء من العنصر شفافة أو معتمة ويوضح الجدول التالي المعاملات التي يمكن استخدامها.

المعامل	الوصف
`clip-source`	رابط يُشير لعنصر SVG أو ملف SVG خارجي يحوي تعريف مسار القطع.
`basic-shape`	تُحدد شكل مسار القطع، وتأخذ القيمة `inset()` أو `circle()` أو `ellipse()` أو `polygon()`.
`clip-geometry-box`	تأخذ إحدى القيم `content-box` أو `padding-box` أوّ أو `margin-box` أو `fill-box` أو `stroke-box` أو `view-box`، وتستعمل حواف الصندوق المحدد كمسار للقطع.
`mask-reference`	تُشير إلى الصورة التي ستُستخدم كقناع.
`repeat-style`	تُحدد كيف ستُكرر صورة القناع على المحورين الأفقي والرأسي، وتأخذ القيمة `repeat-x` أو `repeat-y` أو `repeat` أو `space` أو `round` أو `no-repeat`.
`mask-mode`	تأخذ إحدى القيم alpha أو luminance أو auto.
`position`	تُحدد موضع القناع، وتسلك سلوك مشابه لسلوك الخاصية `background-position`.
`geometry-box`	تُحدد الصندوق الذي سيأخذ شكل القطع، وتأخذ القيمة `content-box` أو `padding-box` أوّ أو `margin-box` أو `fill-box` أو `stroke-box` أو

المعامل	الوصف
	`view-box`. لمزيد من المعلومات انظر توثيق W3C.
`bg-size`	تُحدد حجم صورة القناع، ولها صياغة مماثلة لصياغة الخاصية `background-size`.
`compositing-operator`	تُحدد عملية التركيب التي ستُطبَّق على طبقات القناع، وتأخذ إحدى القيم add أو substract أو exclude أو multiply. لمزيد من المعلومات انظر توثيق W3C.

8.7.1 القطع Clipping

القواطع هي مسارات مُتجِهة، ويكون العنصر خارج هذه المسارات شفافًا، أما الشكل المُحدد بالمسار فيكون معتمًا.



مثال:

```
clip-path: circle(100px at center)
```

يكون الجزء المُعتم (المرئي) من العنصر على شكل دائرة مركزها هو نفس مركز العنصر وقطرها 100 بكسل.

8.7.2 القناع

تستخدم لتحديد قناع يوضع فوق العنصر ويحدد الأجزاء المرئية والمعتمة منه، وهناك نوعان من الأقنعة، أقنعة الإضاءة luminance masks وأقنعة ألفا alpha masks.

أما أقنعة الإضاءة، فيمكنك هذا النوع من وضع قناع ذو تدرج رمادي فوق العنصر، بحيث يكون العنصر تحت الجزء الأسود من القناع معتمًا، بينما يكون الجزء الواقع تحت الجزء الأبيض من القناع شفافًا، ويتدرج العنصر من التعتيم إلى الشفافية حسب تدرج القناع.

مثال:

```
mask: url(masks.svg#rectangle) luminance;
```

وأما أقنعة ألفا، فيكون الجزء من العنصر الواقـع تحت الجـزء الشـفاف من القنـاع مرئيًـا بينمـا يكـون بـاقي العنصر شفافًا.

ا. تحويل الصورة من معتمة إلى شفافة تدريجيًا

في المثال التالي، استُخدِم قناع بتدرج خطي ذو لون أبيض في أقصى اليسـار ويصـبح شـفافًا كلمـا اتجهت لليمين، وتصبح الصورة شفافة في المكان الذي يكون فيه القناع شفافًا.

```
<style>
    div {
        height: 200px;
        width: 200px;
        background: url('http://lorempixel.com/200/200/nature/1');
        mask-image: linear-gradient(to right, white, transparent);
    }
</style>

<div></div>
```

النتيجة:

يجب استخدام البادئة `-webkit` لدعم الخاصية `mask-image` في بعض المتصفحات.

8.8 استخدام الأقنعة لإنشاء ثقب في الصورة

في المثال التالي، أُنشِئت دائرة شفافة في منتصف الصورة باستخدام الخاصية linear-gradient، مما يؤدي لإنشاء ثقب شفاف في الصورة.

```
<style>
    div {
        width: 200px;
        height: 200px;
        background: url(http://lorempixel.com/200/200/abstract/6);
        mask-image: radial-gradient(circle farthest-side at center,
transparent 49%, white 50%);
    }
</style>

<div></div>
```

الصورة الأصلية وبجانبها الصورة بعد إنشاء الثقب:

مثال استخدام الأقنعة لإنشاء صور بأشكال غير منتظمة:

```
<style>
    div {
        height: 200px;
        width: 400px;
        background-image: url(http://lorempixel.com/400/200/nature/4);
        mask-image: linear-gradient(to top right, transparent 49.5%,
white 50.5%), linear-gradient(to top
```

```
        left, transparent 49.5%, white 50.5%), linear-gradient(white,
white);
        mask-size: 75% 25%, 25% 25%, 100% 75%;
        mask-position: bottom left, bottom right, top left;
        mask-repeat: no-repeat;
    }
</style>

<div></div>
```

النتيجة:

8.8.1 القواطع

```
<style>
    div {
        width: 200px;
        height: 200px;
```

```
        background: teal;
        clip-path: polygon(0 0, 0 100%, 100% 50%);
    }
</style>

<div></div>
```

المثال أعلاه يوضح كيفية استخدام الخاصية `clip-path` لقطع الشكل المربع إلى شكل مثلث، يبدأ القطع في الحافة العُليا باتجاه اليسار (عند النقطة 0 0)، ومن ثمة يتجه إلى أسفل اليسار (النقطة 100% 0)، وأخيرًا إلى منتصف الضلع الأيمن (عند النقطة 100% 50%)، وتحدد هذه النقاط الثلاث رؤوس المثلث.

النتيجة:

ا. إنشاء شكل دائري باستخدام القواطع

مثال:

```
<style>
    div {
        width: 200px;
        height: 200px;
        background: teal;
        clip-path: circle(30% at 50% 50%);
    }
</style>

<div></div>
```

المثال أعلاه يوضح كيفية إنشاء شكل دائري باستخدام الخاصية `clip-path`، ويكون الناتج هو شكل دائري نصف قطره 30% من عرض العنصر الأصلي، ومركزه هو نفس مركز العنصر.

اطلع على تجربة حية للمثال أعلاه على JSFiddle.

الصيغة العامة لرسم الشكل الدائري:

```
circle(radius at x y)
```

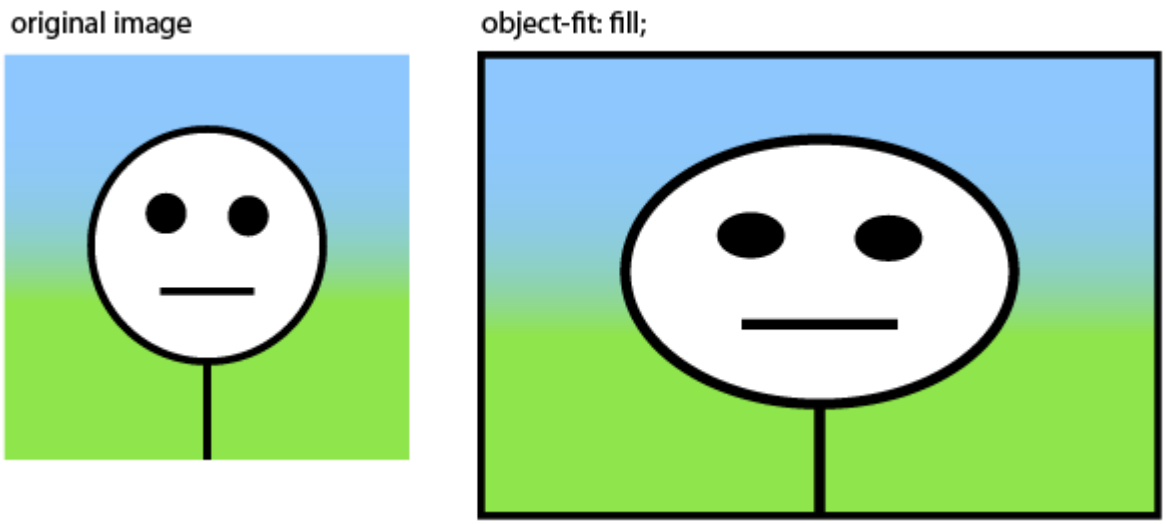
8.9 الخاصية object-fit

تُحدد الخاصية object-fit كيف يتناسب العنصر مع صندوق ذو ارتفاع وعرض محددين، وغالبًا ما تستخدم مع الصور والفيديوهات، ويمكن أن تأخذ إحدى القيم التالية:

مثال مع القيمة fill:

```
object-fit: fill;
```

النتيجة:

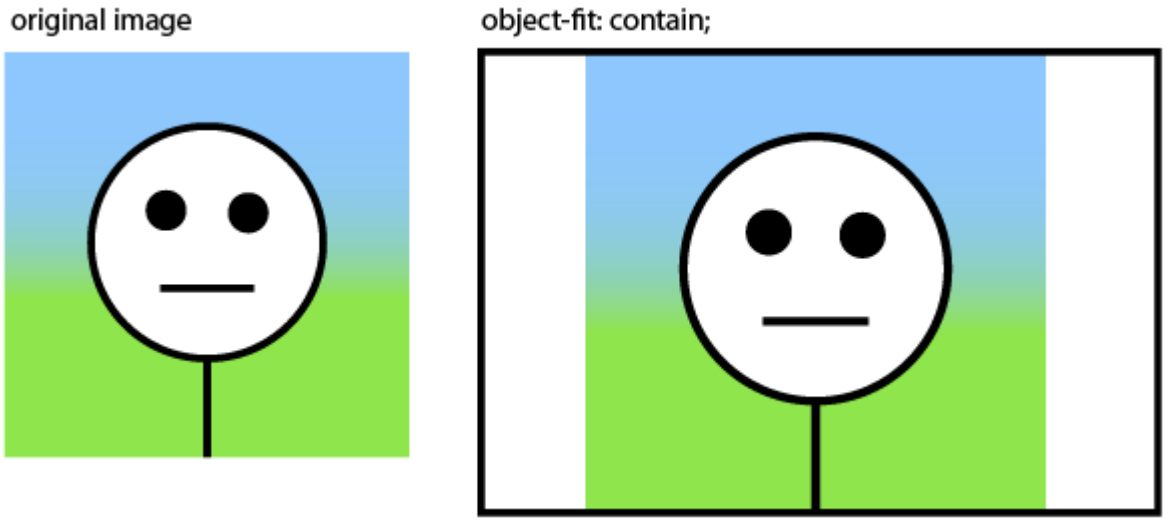


تُمدد `fill` الصورة لملء صندوق المحتوى بغض النظر عن نسبة العرض إلى الارتفاع الأصلية للصورة.

مثال مع القيمة contain:

```
object-fit: contain;
```

النتيجة:

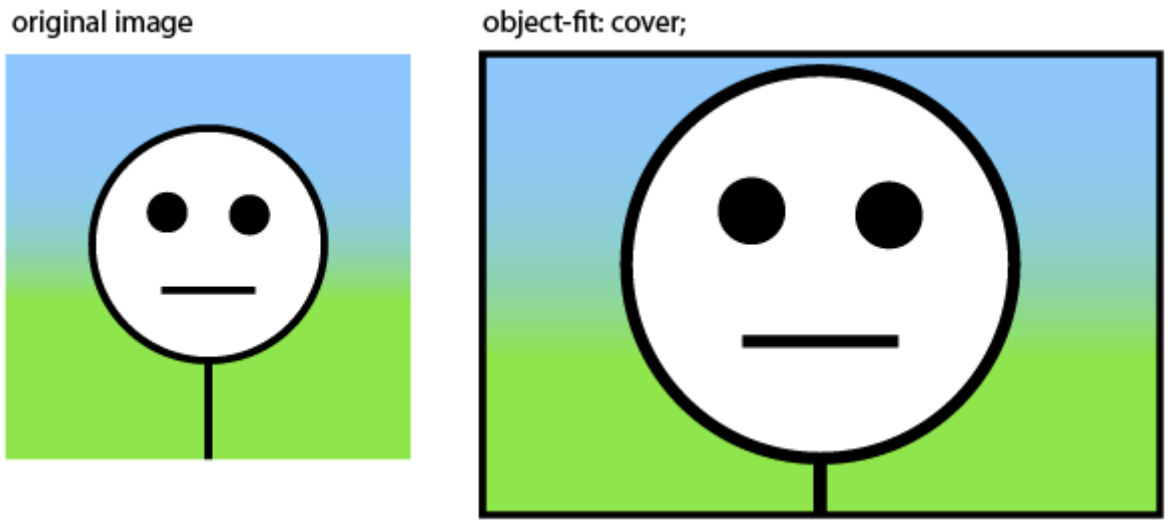


تُحافظ `contain` على نسبة الارتفاع إلى العرض الأصلية للصورة عن طريق ملائمتها لأحد بُعـدي الصـندوق وضبط البعد الآخر بناءً على نسبة الارتفاع إلى العرض الأصلية.

مثال باستخدام القيمة COVER:

```
object-fit: cover;
```

النتيجة:

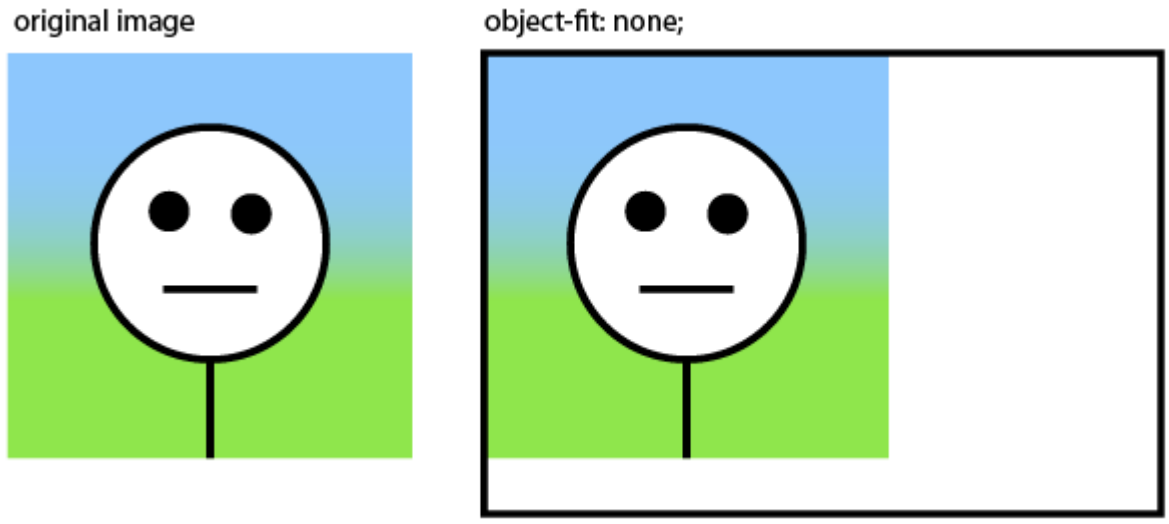


تُمدد `cover` الصورة لملء صندوق المحتوى، مع الحفاظ على نسبة العـرض إلى الارتفـاع الأصـلية للصـورة عن طريق قصها.

مثال باستخدام القيمة none:

```
object-fit: none;
```

النتيجة:

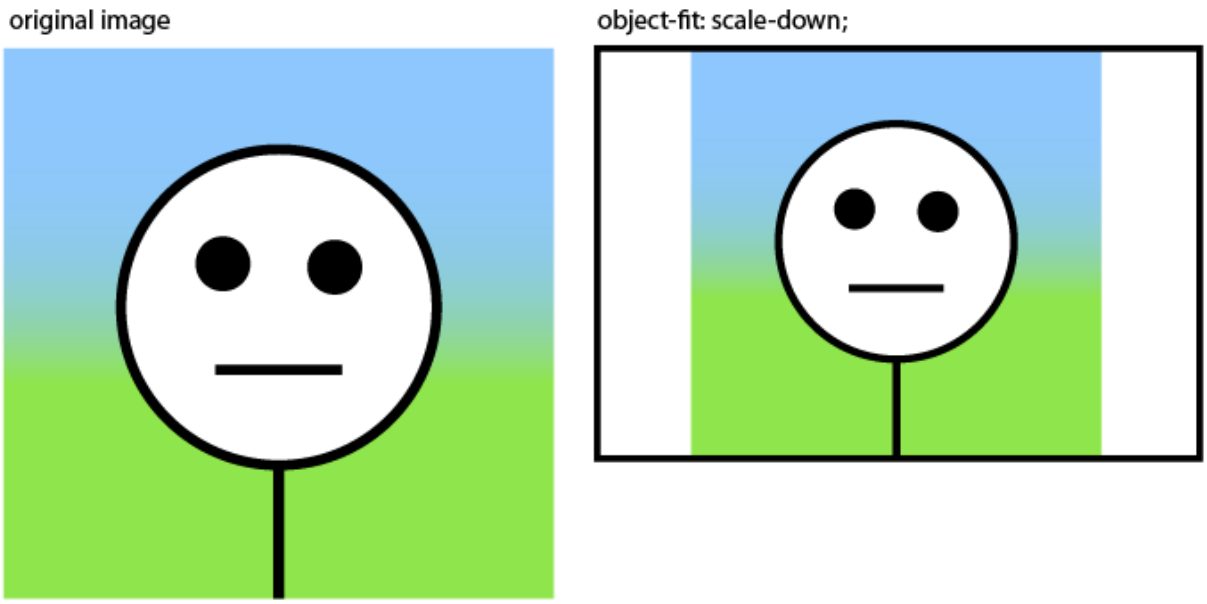


تتجاهل none حجم الصندوق وتُحافظ على الحجم الأصلي للصورة.

مثال باستخدام القيمة scale-down:

```
object-fit: scale-down;
```

النتيجة:



تختار تلقائيًا قيمة object-fit التي تُنتج الصورة الأصغر.

خمسات

لبيع وشراء الخدمات المصغرة

أكبر سوق عربي لبيع وشراء الخدمات المصغرة
اعرض خدماتك أو احصل على ما تريد بأسعار تبدأ من $5 فقط

تصفح الخدمات

9. العناصر العائمة Floats

9.1 تعويم نص حول صورة

من أبسط استعمالات خاصية `float` تعويم نص حول صورة، كما هو موضح بالمثال التالي:

- ملف HTML

```
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer
nec odio. Praesent libero. Sed

  cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum
imperdiet. Duis sagittis

  ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta.
Mauris massa. Vestibulum lacinia

  arcu eget nulla.</p>


  <img src="https://loremflickr.com/200/100" />


  <p>Class aptent taciti sociosqu ad litora torquent per conubia
nostra, per inceptos himenaeos.

  Curabitur sodales ligula in libero. Sed dignissim lacinia nunc.
Curabitur tortor. Pellentesque

  nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed
convallis tristique sem. Proin

  ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis
vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula
lacinia aliquet.</p>
```

- ملف CSS:

```
img {
    float:left;
    margin-right:1rem;
}
```

النتيجة:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet.

اطّلع على تجربة حية لهذا المثال على Codepen.

9.2 الرابط العجيب بين الخاصية clear والخاصية float

ترتبط خاصية `clear` ارتباطًا وثيقًا خاصية `float`، وتستخدم لمنع تعويم العناصر على جانب معين من الحاوية، ويمكن أن تأخذ القيم التالية:

القيمة	الوصف
none	تسمح بتعويم العناصر على الجانبين.
left	تمنع تعويم العناصر على الجانب الأيسر من الحاوية.
right	تمنع تعويم العناصر على الجانب الأيمن من الحاوية.

القيمة	الوصف
both	تمنع تعويم العناصر على الجانبين.
initial	تسترجع القيمة الابتدائية لخاصية float.
inherit	تُوَرِّث قيمة خاصية float من العنصر الأب للابن.

مثال:

```
<html>
    <head>
        <style>
            img {
                float: left;
            }
            p.clear {
                clear: both;
            }
        </style>
    </head>
    <body>
        <img src="https://static.pexels.com/photos/69372/pexels-photo-69372-medium.jpeg" width="100">
        <p>Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem
        ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum </p>
        <p class="clear">Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
        Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum </p>
    </body>
</html>
```

9.3 مفهوم Clearfix

مفهوم clearfix هو مفهوم آخر مرتبط خاصية float، والأمثلة التالية توضح كيفية استخدامه:

- استخدام Clearfix مع هوامش متداخلة:

```
.cf:after {
    content: "";
```

```
    display: table;
}

.cf:after {
    clear: both;
}
```

- استخدام Clearfix لمنع تداخل الهوامش العلوية

```
cf:before,
.cf:after {
    content: " "; /* 1 انظر الملاحظة رقم */
    display: table; /* 2 انظر الملاحظة رقم */
}
.cf:after {
    clear: both;
}
```

اطَّلع على تجربة حيَّة لاستخدامات clearfix على codepen.

ملاحظات:

1. إضافة حرف الفراغ للعناصر الزائفة `:after` و`:before` يمنع ظهور فراغات حول العنصر.
2. يجب استعمال `table` بدلاً عن `block` في حال استخدام `:before` لاحتواء هوامش العنصر الابن.

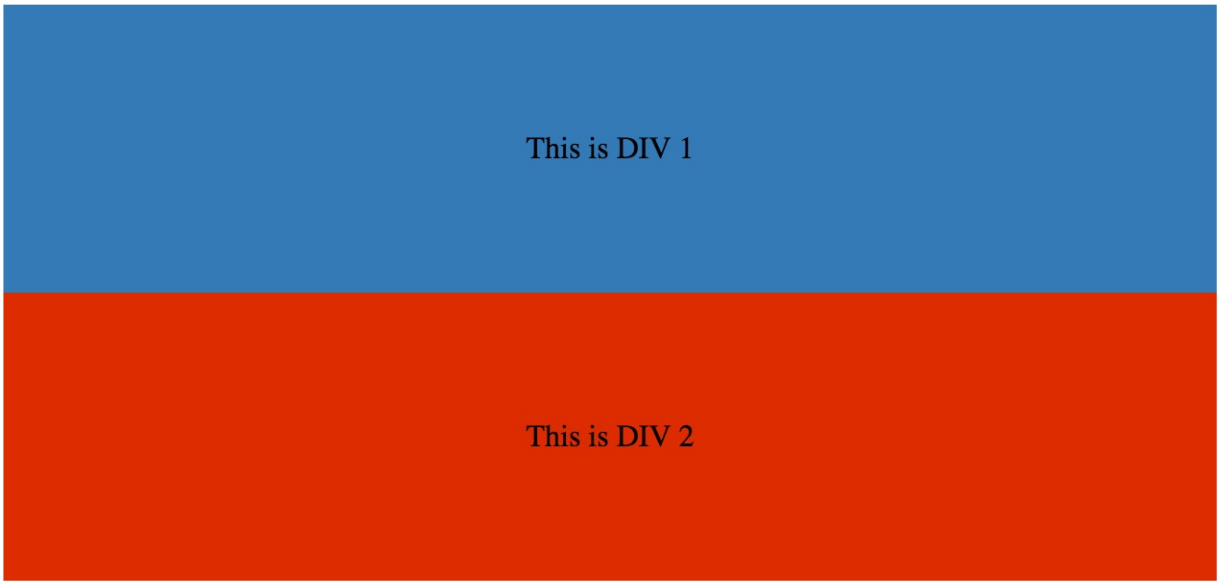
9.4 تحويل حاوية div إلى حاوية سطرية inline باستخدام float

عنصر `div` هو عنصر كُتَلي، أي أنه يأخذ كل عرض الشاشة، وتوضع العناصر الإخوة له رأسيًا فوق بعضها البعض.

مثال:

```
<div>
    <p>This is DIV 1</p>
</div>
<div>
    <p>This is DIV 2</p>
</div>
```

ويكون ناتج الشيفرة أعلاه:



يمكن تحويل عنصر div الى حاوية سطرية (أي تأخذ فقط عرض محتواها وتوضع العناصر الإخوة لها متجاورة أفقيًا) باستخدام خاصية float.

مثال:

- ملف HTML:

```
<div class="outer-div">
    <div class="inner-div1">
        <p>This is DIV 1</p>
    </div>
    <div class="inner-div2">
        <p>This is DIV 2</p>
    </div>
</div>
```

- ملف CSS:

```
.inner-div1 {
    width: 50%;
    margin-right:0px;
    float:left;
    background : #337ab7;
    padding:50px 0px;
```

```
    }

    .inner-div2 {
        width: 50%;
        margin-right:0px;
        float:left;
        background : #dd2c00;
        padding:50px 0px;
    }

    p {
        text-align:center;
    }
```

النتيجة:

اطّلع على تجربة حيَّة لهذا المثال على Codepen.

9.5 إصلاح الطوفان بضبط الطفحان

تحصل مشاكل بين العناصر عندما يطبق الطوفان على عنصر بينها لذا يمكن ضبط خاصية الطفحان overflow إلى القيمة `hidden` أو `auto` أو `scroll` لعنصر محدد لإصلاح تلك المشاكل.

> استخدام القاعدة `overflow: scroll` سيظهر شريط التمرير بشكل دائم.

9.6 إنشاء تخطيط بسيط ذو عمودين بعرض ثابت

المثال أدناه يوضح طريقة إنشاء تخطيط ذو عمودين بعرضين ثابتين باستخدام خاصية `float`.

- ملف HTML

```
    <div class="wrapper">
        <div class="sidebar">
            <h2>Sidebar</h2>
```

```
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Integer nec odio.</p>
        </div>
        <div class="content">
            <h1>Content</h1>
            <p>Class aptent taciti sociosqu ad litora torquent per
conubia nostra, per inceptos himenaeos.
            Curabitur sodales ligula in libero. Sed dignissim lacinia
nunc. Curabitur tortor. Pellentesque
            nibh. Aenean quam. In scelerisque sem at dolor. Maecenas
mattis. Sed convallis tristique sem. Proin
            ut ligula vel nunc egestas porttitor. Morbi lectus risus,
iaculis vel, suscipit quis, luctus non,
            massa. Fusce ac turpis quis ligula lacinia aliquet. </p>
        </div>
    </div>
```

- ملف CSS:

```
    .wrapper {
        width: 600px;
        padding: 20px;
        background-color: pink;

        /* انظر الملاحظة 1 */
        overflow:hidden;
    }

    .sidebar {
        width: 150px;
        float: left;
        background-color: blue;
    }

    .content {
        width: 450px;
        float: right;
        background-color: yellow;
```

```
}
```

العناصر العائمة floating elements ليس لديها ارتفاع محدد ويُجبر استعمال القاعدة `overflow: hidden` العنصر الأب على التمدد وأخذ ارتفاع العناصر الأبناء.

9.7 إنشاء تخطيط ذو ثلاث أعمدة بعرض ثابت

انظر المثال التالي:

- ملف HTML:

```
<div class="wrapper">
    <div class="left-sidebar">
        <h1>Left Sidebar</h1>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
</p>
    </div>

    <div class="content">
        <h1>Content</h1>
        <p>Class aptent taciti sociosqu ad litora torquent per
conubia nostra, per inceptos himenaeos.
        Curabitur sodales ligula in libero. Sed dignissim lacinia
nunc. Curabitur tortor. Pellentesque
        nibh. Aenean quam. In scelerisque sem at dolor. Maecenas
mattis. Sed convallis tristique sem. Proin
        ut ligula vel nunc egestas porttitor. Morbi lectus risus,
iaculis vel, suscipit quis, luctus non,
        massa. </p>
    </div>

    <div class="right-sidebar">
        <h1>Right Sidebar</h1>
        <p>Fusce ac turpis quis ligula lacinia aliquet.</p>
    </div>
</div>
```

- ملف CSS:

```
.wrapper {
```

```
        width:600px;
        background-color:pink;
        padding:20px;
        overflow:hidden;
    }
    .left-sidebar {
        width:150px;
        background-color:blue;
        float:left;
    }
    .content {
        width:300px;
        background-color:yellow;
        float:left;
    }
    .right-sidebar {
        width:150px;
        background-color:green;
        float:right;
    }
```

9.8 إنشاء تخطيط ذو عمودين بعرض ديناميكي (غير ثابت)

المثال التالي يستخدم عنصرًا عائمًا واحد لإنشاء تخطيط ذي عمودين بعرض ديناميكي. تأخذ الحاوية `.sidebar` فقط مساحة محتواها، بينما تأخذ الحاوية `content`. كل المساحة المتبقية من الشاشة.

- ملف HTML:

```
    <div class="sidebar">
        <h1>Sidebar</h1>
        <img src="http://lorempixel.com/150/200/" />
    </div>

    <div class="content">
        <h1>Content</h1>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Integer nec odio. Praesent libero. Sed
```

```
      cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh
elementum imperdiet. Duis sagittis

      ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta.
Mauris massa. Vestibulum lacinia

      arcu eget nulla. </p>

      <p>Class aptent taciti sociosqu ad litora torquent per conubia
nostra, per inceptos himenaeos.

      Curabitur sodales ligula in libero. Sed dignissim lacinia nunc.
Curabitur tortor. Pellentesque

      nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis.
Sed convallis tristique sem. Proin

      ut ligula vel nunc egestas porttitor. Morbi lectus risus,
iaculis vel, suscipit quis, luctus non,

      massa. Fusce ac turpis quis ligula lacinia aliquet. Mauris
ipsum. Nulla metus metus, ullamcorper

      vel, tincidunt sed, euismod in, nibh.</p>

  </div>
```

- ملف CSS:

```
  .sidebar {
      /* `display:table;` shrink-wraps the column */
      display:table;
      float:left;
      background-color:blue;
  }


      .content {
      /* `overflow:hidden;` prevents `.content` from flowing under
`.sidebar` */
      overflow:hidden;
      background-color:yellow;
  }
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

9.9 شكل مساحة التعويم

تُحدد الخاصية `shape-outside` شكل مساحة التعويم، وهي المساحة التي تُحيط بالعنصر المُعَوَّم ويوضح الجدول التالي القيم التي تأخذها:

المعامل	الوصف
none	تعني أن مساحة التعويم float area (المساحة حول العنصر العائم) لا تتأثر.
basic-shape	تُحدد شكل مساحة التعويم وتأخذ إحدى القيم inset() أو circle()، أو ellipse()، أو polygon().
shape-box	تحدد الصندوق الذي سيأخذ الشكل، وتأخذ إحدى القيم margin-box، أو border-box، أو padding-box، أو content-box.
image	تُحدد صورة توضع داخل الشكل.

مثال:

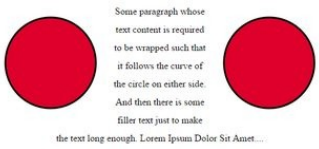
- ملف CSS

```
    img:nth-of-type(1) {
        shape-outside: circle(80px at 50% 50%);
        float: left;
        width: 200px;
    }

    img:nth-of-type(2) {
        shape-outside: circle(80px at 50% 50%);
        float: right;
        width: 200px;
    }
    p {
        text-align: center;
        line-height: 30px;
    }
```

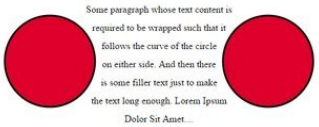
- ملف HTML

```
    <img src="http://images.clipartpanda.com/circle-clip-art-
circlergb.jpg">
    <img src="http://images.clipartpanda.com/circle-clip-art-
circlergb.jpg">
    <p>Some paragraph whose text content is required to be wrapped such
that it follows the curve of
    the circle on either side. And then there is some filler text just
to make the text long enough.
    Lorem Ipsum Dolor Sit Amet....</p>
```

بمـا أن شـكل النمـوذج الصـندوقي للصـورتين هـو دائمًـا مربـع، ففي حـال عـدم اسـتخدام الخاصـية `shape-outside` لن يطفو النص حول الشكل الـدائري للصـورة وإنمـا سـيطفو حـول النمـوذج الصـندوقي ذو الشكل المربع، كما هو موضح في الصورة أدناه.



ولكن عند استخدام الخاصية `shape-outside` يُعاد تعريف الشكل الخارجي للعنصـر ويصـبح دائريًـا ممـا يجعل النص يطفو حول الدائرية التخيلية التي ترسمها هذه الخاصية، كما هو موضح في الصورة التالية:



9.9.1 خاصية شكل الهامش shape-margin

تُسـتخدم الخاصـية `shape-margin` لتعريـف شـكل الهـوامش للعنصـر، وتأخـذ نفس قيم الخاصـية `shape-outside`. مثال:

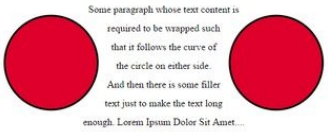
- ملف CSS

```
img:nth-of-type {
    shape-outside: circle(80px at 50% 50%);
    shape-margin: 10px;
    float: left;
    width: 200px;
}
img:nth-of-type(2) {
    shape-outside: circle(80px at 50% 50%);
    shape-margin: 10px;
    float: right;
    width: 200px;
}
p {
    text-align: center;
    line-height: 30px;
}
```

- ملف HTML

```
  <img src="http://images.clipartpanda.com/circle-clip-art-
circlergb.jpg">
  <img src="http://images.clipartpanda.com/circle-clip-art-
circlergb.jpg">
  <p>Some paragraph whose text content is required to be wrapped such
that it follows the curve of
  the circle on either side. And then there is some filler text just
to make the text long enough.
      Lorem Ipsum Dolor Sit Amet....</p>
```

في هذا المثال أُضيف هامش بعرض 10 بكسلات حول العنصر، وكما هو معروف أن شكل صندوق الهوامش مربع، لذلك يجب استخدام الخاصية `shape-margin` لإعادة تعريف الشكل وجعله دائريًا.



دورة تطوير تطبيقات الويب
باستخدام لغة Ruby

دورة تدريبية متكاملة من الصفر وحتى الاحتراف
تمكنك من التخصص في هندسة الويب ودخول سوق العمل

التحق بالدورة الآن

10. الانتقالات Transitions والحركات Animations

10.1 الانتقالات عبر الخاصية transition

يوضح الجدول التالي خاصيات الانتقال `transition` التفصيلية:

الخاصية	الوصف
`transition-property`	تُستعمل لتحديد خاصيات CSS التي ستخضع لتأثير الانتقال.
`transition-duration`	تُحدِّد الزمن الذي سيستغرقه تأثير الانتقال للوصول إلى القيمة النهائية للخاصية، والقيمة الافتراضية `0s` ستلغي تأثير الانتقال
`transition-timing-function`	تُستعمل لوصف كيف ستتأثر القيم الوسطية لخاصيات CSS بتأثير الانتقال، أي أنها تسمح لك بتحديد منحني التسارع acceleration curve لتأثير الانتقال. انظر التوثيق لمزيد من المعلومات.
`transition-delay`	تُحدِّد الزمن الذي يجب انتظاره بين وقوع الحدث الذي سيؤدي إلى بدء تأثير الانتقال وبين بداية التأثير

إليك مثال بسيط لتغيير لون العنصر من اللون الأحمر إلى اللون الأخضر تدريجًا باستعمال الخاصية `transition` خلال ثانية واحدة عند تمرير مؤشر الفأرة فوقه:

- ملف CSS

```
div {
    width: 150px;
    height: 150px;
    background-color: red;
```

```
        transition: background-color 1s;
    }

    div:hover {
        background-color: green;
    }
```

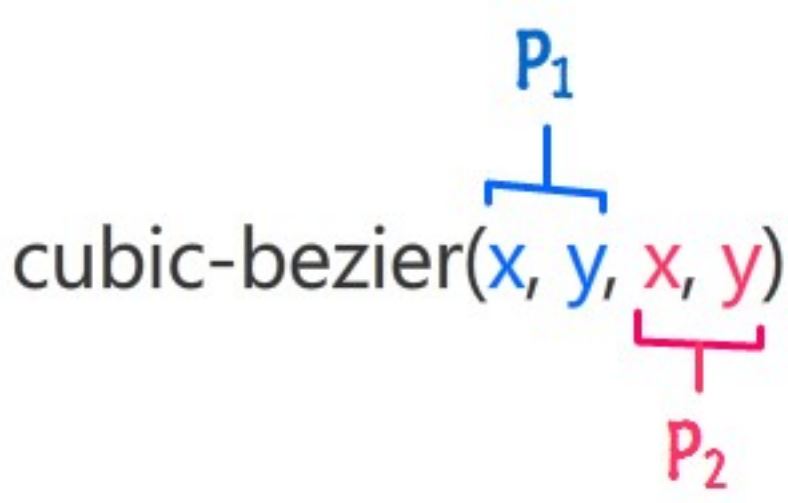
- ملف HTML

```
    <div></div>
```

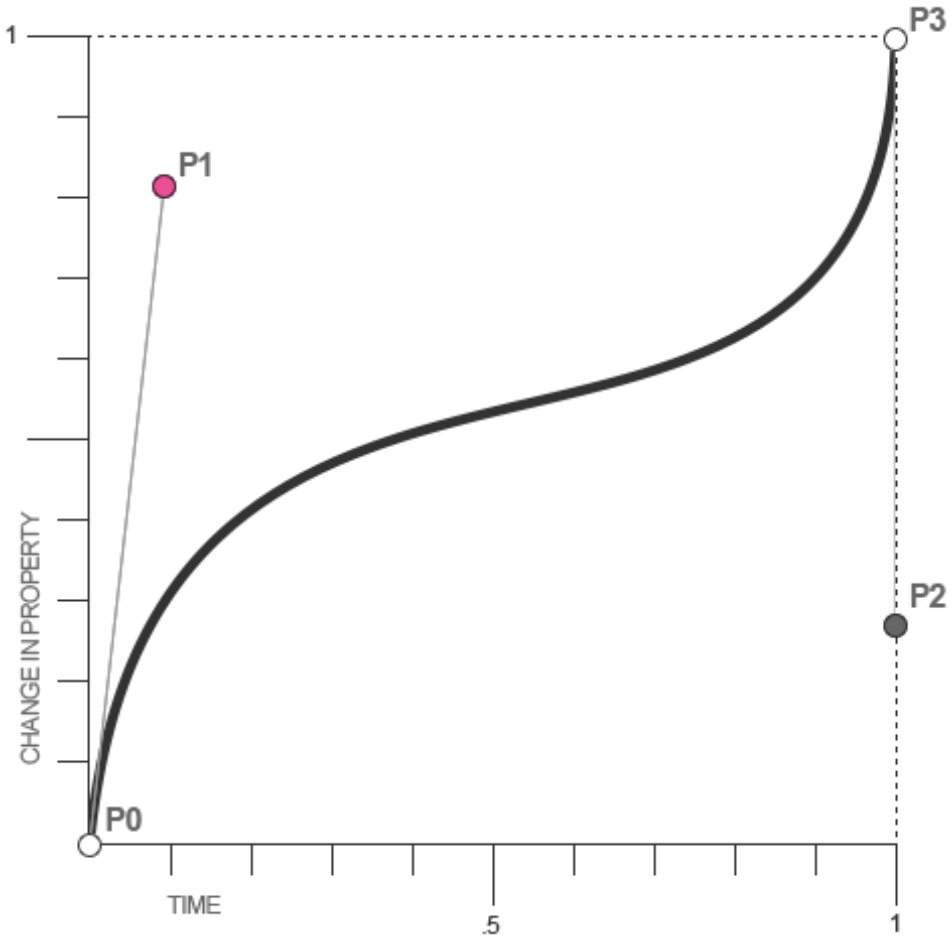
10.1.1 دالة التسارع cubic-bezier

تسمح لك دالة `cubic-bezier` بتخصيص منحى التسارع للحصول على انتقالات سلسة، وتأخذ أربعة معاملات:

```
  cubic-bezier(P1_x, P1_y, P2_x, P2_y)
```



تُرسم هذه المعاملات على مُنحنى bezier، ودائمًا ما يكون المعاملين P0 و P3 ثابتين، يكون P0 في النقطة (0،0) و P3 في النقطة (1،1)، وبالتالي يجب أن تكون قيم معاملات الدالة `cubic-bezier` بين صفر وواحد.



يُمكن التعبـير عن جميـع منحنيـات التسـار ع باسـتخدام دالـة `cubic-bezier` كمـا هـو موضـح في المثال التالي:

```
linear: cubic-bezier(0.0, 0.0, 1.0, 1.0)

ease-in: cubic-bezier(0.42, 0.0, 1.0, 1.0)

ease-out: cubic-bezier(0.0, 0.0, 0.58, 1.0)

ease-in-out: cubic-bezier(0.42, 0.0, 0.58, 1.0)
```

مثال:

- ملف CSS

```
  div {
      height: 100px;
      width: 100px;
```

```
        border: 1px solid;
        transition-property: height, width;
        transition-duration: 1s, 500ms;
        transition-timing-function: linear;
        transition-delay: 0s, 1s;
    }

    div:hover {
        height: 200px;
        width: 200px;
    }
```

- ملف HTML

```
    <div></div>
```

شرح الخاصيات:

- `transition-property`: تُحدِّد هذه الخاصية الخاصيات التي ستخضع لتأثير الانتقال. وفي هذا المثال سيزيد ارتفاع وعرض الحاوية تدريجيًا عند تمرير مؤشر الفأرة فوقها.
- `transition-duration`: تُحدِّد الزمن الذي سيستغرقه تأثير الانتقال للوصول إلى القيمة النهائية للخاصية. وفي هذا المثال تستغرق الخاصية `height` ثانية واحدة للانتقال من القيمة 100 بكسل إلى 200 بكسل، بينما تستغرق الخاصية `width` نصف ثانية (500 ملي ثانية).
- `transition-timing-function`: تُحدد منحى التسارع للانتقال. القيمة الخطية (linear) تعني أن لعملية الانتقال سرعة ثابتة.
- `transition-delay`: تُحدِّد الزمن الذي يجب انتظاره بين وقوع الحدث الذي سيؤدي إلى بدء تأثير الانتقال وبين بداية التأثير. وفي هذا المثال يبدأ تأثير الانتقال للخاصية `height` لحظيًا (أي فور تمرير مؤشر الفأرة فوق الحاوية)، بينما يبدأ تأثير الانتقال للخاصية `width` بعد مرور ثانية واحدة من ذلك.

10.2 إنشاء الحركات باستخدام خاصية transition

يُمكن استخدام الخاصية `transition` لإنشاء الحركات البسيطة والغير معقدة.

مثال:

```
.example {
    height: 100px;
    background: #fff;
}

.example:hover {
    height: 120px;
    background: #ff0000;
}
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

عند تمرير مؤشر الفأرة على أي عنصر يمتلك الصنف `.example` يتحول الارتفاع مباشرة إلى 120 بكسل ويتغير لون الخلفية إلى اللون الأحمر `#ff0000`، وبإضافة خاصية `transition` يُمكن التحكم في المدة الزمنية التي يستغرقها هذا الانتقال.

```
.example {
    ...
    transition: all 400ms ease;
}
```

اطَّلع على تجربة حية لهذا المثال على JSFiddle.

10.3 دعم المتصفحات لخاصية transition

خاصية `transition` مدعومة في جميع المتصفحات الأساسية وبعض الإصدارات القديمة من متصفح Firefox، ويُمكن استعمال البوادئ الخاصة بهذه المتصفحات لدعم خاصية `transition` فيها.

```
.example {
    transition: all 400ms ease;
    -moz-transition: all 400ms ease;
    -webkit-transition: all 400ms ease;
}
```

10.4 تحريك العناصر عبر الخاصية animation

تستعمل الخاصية `animation` لتطبيق حركة محددة على عنصر، وتأخذ إحدى القيم التالية:

المعامل	الوصف
`property`	تحدِّد خاصيات CSS التي ستتأثر بالحركات.
`duration`	تحدِّد الزمن اللازم لإكمال دورة كاملة من الحركة.
`timing-function`	تُستَعمل لوصف كيف ستتأثر القيم الوسطية لخاصيات CSS بالحركة، أي أنها تسمح لك بتحديد منحني التسار ع acceleration curve للحركة خلال دورة واحدة. انظر دليل منحنيات التسار ع.
`delay`	تُحدِّد متى ستبدأ الحركة، ويمكنك تأخير بدء الحركة فترةً من الزمن، أو البدء مباشرةً، أو البدء مباشرةً لكن مع تجاوز جزء من دورة الحركة.

10.4.1 إنشاء الحركات مخصصة باستخدام @keyframes

تستخدم القاعدة `@keyframes` للتحكم بالخطوات البينية (intermediate steps) في سلسلة حركات CSS عبر تعريف أنماط للإطارات المفتاحية (keyframes)، مما يعطينا تحكمًا كبيرًا في أنماط العنصر في الخطوات البينية بالموازنة مع الانتقالات `transition`.

مثال:

```
@keyframes rainbow-background {
{ background-color: #ff0000; }
    8.333%  { background-color: #ff8000; }
    16.667% { background-color: #ffff00; }
    25.000% { background-color: #80ff00; }
    33.333% { background-color: #00ff00; }
    41.667% { background-color: #00ff80; }
    50.000% { background-color: #00ffff; }
    58.333% { background-color: #0080ff; }
    66.667% { background-color: #0000ff; }
    75.000% { background-color: #8000ff; }
    83.333% { background-color: #ff00ff; }
    91.667% { background-color: #ff0080; }
    100.00% { background-color: #ff0000; }
}

.RainbowBackground {
    animation: rainbow-background 5s infinite;
}
```

اطَّلع على تجربة حيَّة للمثال أعلاه على JSFiddle.

لاحظ أن القاعدة `@keyframes` تُكتب بالصيغة الموضحة بالشيفرة أدناه حُيث `rainbow-background` هو اسم الحركة.

```
@keyframes rainbow-background {
{background-color: #ff0000; }
}
```

تمثل القيمة 0% نُقطة بداية الحركة، أي الأنماط التي ستُطبق عند بدء الحركة، وتنتقل الحركة تلقائيًا للنُقطة الثانية عند مرور 8.333% من زمن الحركة، ومن ثم إلى النقطة الثالثة عند مرور 16.667% من زمن الحركة وهكذا.

```
.Rainbowbackground {
    animation: rainbow-background 5s infinite;
}
```

تُضيف هذه الشفرة الحركة `rainbow-background` لكل العناصر التي تمتلك الصنف `..RainbowBackground`

تأخذ خاصية `animation` عدد من المعاملات هي:

- `animation-name` : يُحدِّد اسم الحركة.
- `animation-duration` : يُحدِّد الزمن اللازم لإكمال دورة كاملة من الحركة.
- `animation-iteration-count` : (اختياري) يُحدِّد عدد تكرارات الحركة، ويأخذ القيمة `infinite` في حال كانت الحركة لانهائية (مستمرة).
- `animation-delay` : (اختياري) يُحدِّد متى ستبدأ الحركة، ويُمكن أن يأخذ قيم موجبة أو سالبة أو صفر وهي القيمة الافتراضية له. القيم السالبة تعني أن الحركة ستبدأ مُتقدمة قليلًا حسب المقدار المحدد.
- `animation-timing-function` : (اختياري) يُحدِّد منحني التسارع (acceleration curve) للحركة خلال دورة واحدة.

تأخذ الخاصية `background-color` في المثال التالي القيمة `#FF0000` عند النقطتين 0% و 100% لذلك يُمكن تعريف هاتين النقطتين بسطر واحد كما هو موضح بالشيفرة التالية:

```
0%, 100% { background-color: #ff000- }
```

انتبه، إن أردت استعمال قاعدة `@keyframes` في المتصفحات القديمة يجب إضافة البادئة `-webkit` كما هو موضح في الشيفرة التالية:

```
@-webkit-keyframes {}

-webkit-animation: ...
```

10.5 أمثلة لاستعمال الخاصية animation

مثال على الصيغة المختزلة لخاصية `animation`:

```
animation: 3s ease-in 1s 2 reverse both paused slidein;

/* ترتيب الخاصيات */
animation: <duration> <timing-function> <delay> <iteration-count>
<direction> <fill-mode> <play-state> <name>
```

مثال على حذف بعض الخاصيات الاختيارية للخاصية `animation`:

```
animation: 3s linear 1s slidein;

/* ترتيب الخاصيات */
animation: <duration> <linear> <delay> <name>
```

مثال على أبسط تعريف لخاصية `animation`:

```
animation: 3s slidein;

/* ترتيب الخاصيات */
animation: <duration> <name>
```

مثال على كتابة خاصيات الحركات منفصلة:

```
animation-duration: 3s;
animation-timing-function: ease-in;
animation-delay: 1s;
animation-iteration-count: 2;
animation-direction: reverse;
animation-fill-mode: both;
animation-play-state: paused;
```

```
animation-name: slidein
```

10.6 تحسين أداء عملية التحريك عبر will-change

تستعمل خاصية `transition` وقاعدة `@keyframes` معالج الرسومات GPU استعمالًا ثقيلًا، ويمكن تحسين أداء هاتين الخاصيتين باستخدام الخاصية `will-change` التي تُخبر المتصفح أن هناك جزء من الصفحة قابل للتغيير في لحظة زمنية معينة. تأخذ الخاصية `will-change` أسماء الخاصيات التي تتأثر بالخاصية `transition` أو بالقاعدة `@keyframes` وتخبر المتصفح أن هذه الخاصيات ستتغير في لحظة ما.

```
.example {
    ...
    will-change: transform, opacity;
}
```

10.7 التحويلات ثُنائية الأبعاد 2D Transforms

تُستخدم الخاصية `transform` لعمل تحويلات ثنائية أو ثلاثية الأبعاد للعناصر، يوضح الجدول أدناه القيم التي تأخذها هذه الخاصية لإنشاء تحويلات ثنائية الأبعاد، ويُغطي الفصل القادم التحويلات ثلاثية الأبعاد.

الدالة أو المعامل	الوصف
`rotate(x)`	تُستخدم لتدوير العنصر حول المحور Z.
`translate(x,y)`	تُستخدم لتحريك العنصر على المحورين X و Y.
`translateX(x)`	تُستخدم لتحريك العنصر على المحور X.
`translateY(y)`	تُستخدم لتحريك العنصر على المحور Y.
`scale(x,y)`	تُستخدم لإعادة تحجيم أحد العناصر على المحورين X و Y مما يؤدي إلى تكبيره أو تصغيره.
`scaleX(x)`	تُستخدم لإعادة تحجيم أحد العناصر على المحور X.
`scaleY(y)`	تُستخدم لإعادة تحجيم أحد العناصر على المحور Y.
`skew(x,y)`	تستخدم لجعل العنصر منحرفًا، أي كأنَّنا أمسكنا بضلعين متقابلين من المستطيل وحركناهما باتجاهين مختلفين، مما يؤدي إلى تحويل المستطيل إلى متوازي أضلاع.
`skewX(x)`	تستخدم لجعل العنصر منحرفًا في الاتجاه الأفقي (محور X).
`skewY(y)`	تستخدم لجعل العنصر منحرفًا في الاتجاه الرأسي (محور Y).
`matrix()`	تستخدم الدالة `matrix()` لتطبيق مصفوفة تحويلات في الفضاء ثنائي الأبعاد، ويمكن أن تحتوي المصفوفة على أكثر من تحويل في آنٍ واحد.

الدالة أو المعامل	الوصف
`angle`	تُحدد زاوية التحويل التي ستُمرَّر إلى الدالة، ويمكن أن تكون بالدرجات أو بالتقدير الدائري (راديان) أو بعدد الدورات.
`length-or-percentage`	تُحدد مسافة التحويل التي ستُمرَّر إلى الدالة.
`scale-factor`	يُحدد عدد مرات تكبير أو تصغير العنصر.

10.7.1 الدالة rotate

تستخدم الدالة `rotate()` لتدوير العنصر في الفضاء ثنائي الأبعاد بمقدار الزاوية `<angle>` المُمـرَّرة إليـه، القيم الموجبة للزاوية ستؤدي إلى تدوير العنصر باتجاه عقارب الساعة، أما القيم السالبة تُـدَوِّر العنصـر في عكس اتجاه عقارب الساعة.

مثال:

- ملف HTML

```
<div class="rotate"></div>
```

- ملف CSS

```
.rotate {
    width: 100px;
    height: 100px;
    background: teal;
    transform: rotate(45deg);
}
```

ويكـون مركـز الـدوران بشـكل افتراضـي في منتصـف العنصـر، ويمكن تغيـيره عن طريـق الخاصـية `transform-origin` كما هو موضح في المثال التالي:

```
transform-origin: 100% 50%;
```

10.7.2 الدالة scale

تستخدم الدالة `scale()` لإعادة تحجيم أحد العناصر مما يـؤدي إلى تكبـيره أو تصـغيره، وتأخـذ هـذه الدالـة قيمتين رقميتين `<number>`، القيمة الأولى sx تؤدي إلى إعادة تحجيم العنصر على المحور X، أما القيمة الثانيـة sy فتُعيد تحجيمه على المحور Y، وإذا وفّرنا قيمةً واحدةً سيفترض المتصفح أن القيمتان sx و sy متساويتان.

إذا كانت القيمة أكـبر من 1، سـيزداد حجم العنصـر على المحـور المُحـدَّد، وأمّـا إذا كـانت مسـاويةً للـرقم 1 فسيبقى العنصر على حاله (بدون تصغير أو تكبير) على المحور المُحدَّد، أما إذا كانت القيمة بين الصـفر والواحـد

فستؤدي إلى تصغير العنصر، أما القيمة 0 فستخفي العنصر تمامًا من الصفحة؛ ويُسمَح باستخدام القيم السالبة، لكنها لن تؤدي إلى إعادة تحجيم العنصر وإنما قلبه (flip) في الاتجاه المُحدَّد.

مثال:

- ملف HTML

```
<div class="scale"></div>
```

- ملف CSS

```
.scale {
    width: 100px;
    height: 100px;
    background: teal;
    transform: scale(0.5, 1.3);
}
```

10.7.3 الدالة skew

تستخدم الدالة `skew()` لجعل العنصر منحرفًا، أي كأنَّنا أمسكنا بضلعين متقابلين من المستطيل وحرَّكناهما باتجاهين مختلفين، مما يردي إلى تحويل المستطيل إلى متوازي أضلاع.

تأخذ هذه الدالة قيمةً واحدةً أو قيمتين، وكلا القيمتين هي زاوية ، تؤدي القيمة الأولى إلى انحراف العنصر على المحور X، وتؤدي الثانية إلى انحراف العنصر على المحور Y، إذا لم تتوافر القيمة الثانية فستُعدّ أنها صفر.

مثال:

- ملف HTML

```
<div class="skew"></div>
```

- ملف CSS

```
.skew {
    width: 100px;
    height: 100px;
    background: teal;
    transform: skew(20deg, -30deg);
}
```

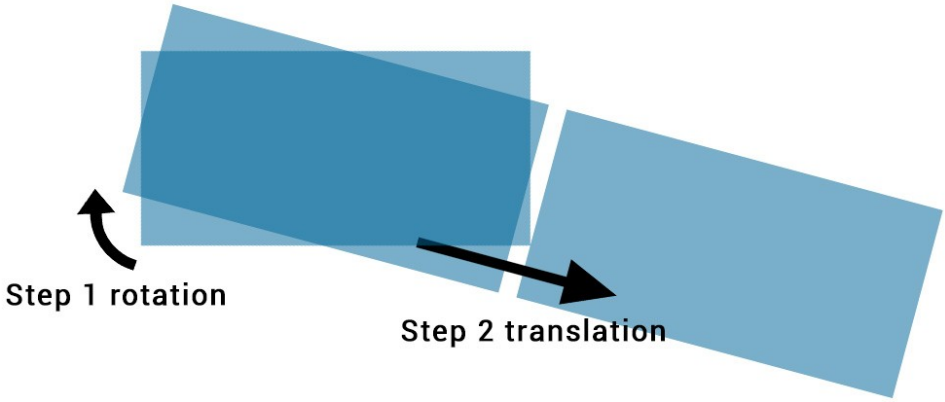
اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

10.7.4 التحويلات المُتعددة

يُمكن إضافة عدد من التحويلات للعنصر كما هو موضح في المثال التالي:

```
transform: rotate(15deg) translateX(200px);
```

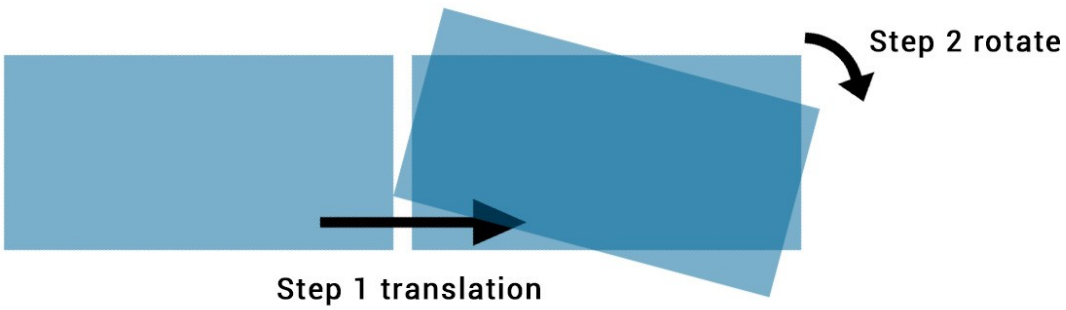
هذه الشيفرة تُدَوِّر العنصر 15 درجة في اتجاه عقارب الساعة ومن ثم تحركه لاتجاه اليمين بمقدار 200 بكسل، ويجب ملاحظة أن المحاور ستُدور مع العنصر مما يجعل اتجاه التحريك مائل بزاوية 15 درجة.



لتحريك العنصر أفقيًا يجب تغيير ترتيب الخواص كما هو موضح في المثال التالي:

```
<style>
.transform {
    transform: translateX(200px) rotate(15deg);
}
</style>

<div class="transform"></div>
```

النتيجة:



10.7.5 دالة translate

تستخدم الدالة `translate()` لتحريك العنصر وفق القيم المُمرَّرة إلى الدالة، إذ تمثل القيمة الأولى التحريك على المحور X، وتمثل القيمة الثانية التحريك على المحور Y، وإذا لم تُحدد القيمة الثانية فسيتحرَّك العنصر على المحور X فقط؛ ويمكن أن تكون كلا القيمتين طولًا مطلقًا `<length>` أو نسبةً مئويةً `<percentage>` من أبعاد صندوق العنصر.

القيم الموجبة ستؤدي إلى تحريك العنصر في الاتجاه الموجب للمحور، والقيم السالبة ستحركه عكس ذلك.

مثال:

- ملف HTML

```
<div class="translate"></div>
```

- ملف CSS

```
.translate {
    width: 100px;
    height: 100px;
    background: teal;
    transform: translate(200px, 50%);
}
```

ويمكن تحريك العنصر على محور X فقط باستعمال الخاصية `translateX` أو على محور Y فقط باستعمال الخاصية `translateY`.

مثال على تحريك العنصر على محور X فقط:

```
.translate {
    transform: translateX(200px);
}
```

مثال على تحريك العنصر على محور Y فقط:

```
.translate {
    transform: translateY(50%);
}
```

10.7.6 الخاصية transform-origin

تسمح الخاصية `transform-origin` بتحديد مبدأ الإحداثيات للتحويلات التي ستجرى على العنصر، والقيمة الافتراضية لمبدأ الإحداثيات هي مركز العنصر، وتأخذ هذه الخاصية قيمتان تحددان الإحداثيات على المحورين X و Y على التوالي.

مثال:

- ملف HTML

```
<div class="transform originl"></div>
<div class="transform origin2"></div>
```

- ملف CSS

```
transform {
    display: inline-block;
    width: 200px;
    height: 100px;
    background: teal;
    transition: transform 1s;
}
.origin1 {
    transform-origin: 0 0;
}
.origin2 {
    transform-origin: 100% 0;
}
.transform:hover {
    transform: rotate(30deg);
}
```

10.8 التحويلات ثلاثية الأبعاد 3D Transforms

10.8.1 إنشاء شكل مؤشر البوصلة باستخدام التحويلات ثلاثية الأبعاد

مثال:

- ملف CSS

```
    div.needle {
        margin: 100px;
        height: 150px;
        width: 150px;
        transform: rotateY(85deg) rotateZ(45deg);
        background-image: linear-gradient(to top left, #555 0%, #555
40%, #444 50%, #333 97%);
        box-shadow: inset 6px 6px 22px 8px #272727;
    }
```

- ملف HTML

```
    <div class="needle"></div>
```

عندما نُطبق خاصية التدوير `rotate`، يحدث الدوران حول محور Z فقط، وفي أفضل الأحوال يكون الشكل الناتج هو شكل الماس (diamond shape)، لذلك نُضيف الخاصية `rotateY` لضغط العنصر في محور Y لإنتاج شكل الإبرة أو مؤشر البوصلة.

ناتج هذا المثال هو إبرة (مؤشر بوصلة) ترتكز على طرفها، ولإنشاء إبرة ترتكز على قاعدتها يجب استبدال الخاصية `rotateY` بالخاصية `rotateX` مع إبقاء نفس قيم الزوايا.

الصور أدناه توضح ناتج هذا المثال:

- صورة العنصر قبل تطبيق التحويلات

- صورة العنصر في حال استخدام التحويلات ثنائية الأبعاد

- صورة العنصر بعد تطبيق التحويلات ثلاثية الأبعاد

10.9 إنشاء نص ثلاثي الأبعاد مع تأثير الظل

مثال:

- ملف HTML

```
<div id="title">
    <h1 data-content="HOVER">
        Hover
    </h1>
</div>
```

- ملف CSS

```
*{margin:0;padding:0;}
html,body{
    height:100%;
    width:100%;
    overflow:hidden;
    background:#0099CC;
}
#title{
    position:absolute;
    top:50%; left:50%;
    transform:translate(-50%,-50%);
    perspective-origin:50% 50%;
    perspective:300px;
}
h1{
    text-align:center;
    font-size:12vmin;
    font-family: 'Open Sans', sans-serif;
    color:rgba(0,0,0,0.8);
    line-height:1em;
    transform:rotateY(50deg);
    perspective:150px;
    perspective-origin:0% 50%;
}
h1:after{
    content:attr(data-content);
    position:absolute;
    left:0;top:0;
    transform-origin:50% 100%;
    transform:rotateX(-90deg);
    color:#0099CC;
}
#title:before{
    content:'';
    position:absolute;
    top:-150%; left:-25%;
```

```
        width:180%; height:328%;
        background:rgba(255,255,255,0.7);
        transform-origin: 0 100%;
        transform: translatez(-200px) rotate(40deg) skewX(35deg);
        border-radius:0 0 100% 0;
    }
```

النتيجة:

اطَّلع على تجربة حيَّة لهذا المثال على Codepen.

10.9.1 الخاصية backface-visibility

تُحدد الخاصية `backface-visibility` ما إذا كان الوجه الخلفي للعنصر ظاهرًا للمستخدم، ويكون الوجه الخلفي للعنصر ذو خلفية شفافة، ويسمح بعرض صورة معكوسة (mirrored) للوجه الأمامي للعنصر، ويمكن أن تأخذ إحدى القيم الآتية:

- `visible`: يكون الوجه الخلفي للعنصر ظاهرًا.
- `hidden`: تُخفي الوجه الخلفي للعنصر.
- `inherit`: ترث قيمة الخاصية من العنصر الأب.
- `initial`: تُرجِع القيمة الابتدائية للخاصية.

مثال:

```
<style>
.flip {
    -webkit-transform: rotateY(180deg);
    -moz-transform: rotateY(180deg);
    -ms-transform: rotateY(180deg);
    -webkit-backface-visibility: visible;
    -moz-backface-visibility: visible;
    -ms-backface-visibility: visible;
}
.flip.back {
    -webkit-backface-visibility: hidden;
    -moz-backface-visibility: hidden;
    -ms-backface-visibility: hidden;
}
</style>

<div class="flip">Loren ipsum</div>
<div class="flip back">Lorem ipsum</div>
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

10.9.2 رسم مُكعَّب باستخدام التحويلات ثُلاثية الأبعاد

مثال:

- ملف HTML

```
  <div class="cube">
      <div class="cubeFace"></div>
      <div class="cubeFace face2"></div>
  </div>
```

- ملف CSS

```
  body {
      perspective-origin: 50% 100%;
      perspective: 1500px;
      overflow: hidden;
  }
```

```
.cube {
    position: relative;
    padding-bottom: 20%;
    transform-style: preserve-3d;
    transform-origin: 50% 100%;
    transform: rotateY(45deg) rotateX(0);
}
.cubeFace {
    position: absolute;
    top: 0;
    left: 40%;
    width: 20%;
    height: 100%;
    margin: 0 auto;
    transform-style: inherit;
    background: #C52329;
    box-shadow: inset 0 0 0 5px #333;
    transform-origin: 50% 50%;
    transform: rotateX(90deg);
    backface-visibility: hidden;
}
.face2 {
    transform-origin: 50% 50%;
    transform: rotatez(90deg) translateX(100%) rotateY(90deg);
}
.cubeFace:before, .cubeFace:after {
    content: '';
    position: absolute;
    width: 100%;
    height: 100%;
    transform-origin: 0 0;
    background: inherit;
    box-shadow: inherit;
    backface-visibility: inherit;
}
.cubeFace:before {
```

```
        top: 100%;
        left: 0;
        transform: rotateX(-90deg);
    }
    .cubeFace:after {
        top: 0;
        left: 100%;
        transform: rotateY(90deg);
    }
```

اطَّلع على تجربة حيَّة لهذا المثال على JSFiddle.

10.10 البوادئ ودعم المتصفحات

يوضح الجدول التالي البوادئ الخاصية بكل متصفح:

البادئة	المتصفحات
`-webkit-`	Google Chrome, Safari, الإصدارات الأحدث من Opera 12 وما بعد, Android, Blackberry, UC
`-moz-`	Mozilla Firefox
`-ms-`	Internet Explorer, Edge
`-o-` `-xv-`	Opera حتى الإصدار 12

مثال:

- خاصية `transition`

```
    div {
        -webkit-transition: all 4s ease;
        -moz-transition: all 4s ease;
        -o-transition: all 4s ease;
        transition: all 4s ease;
    }
```

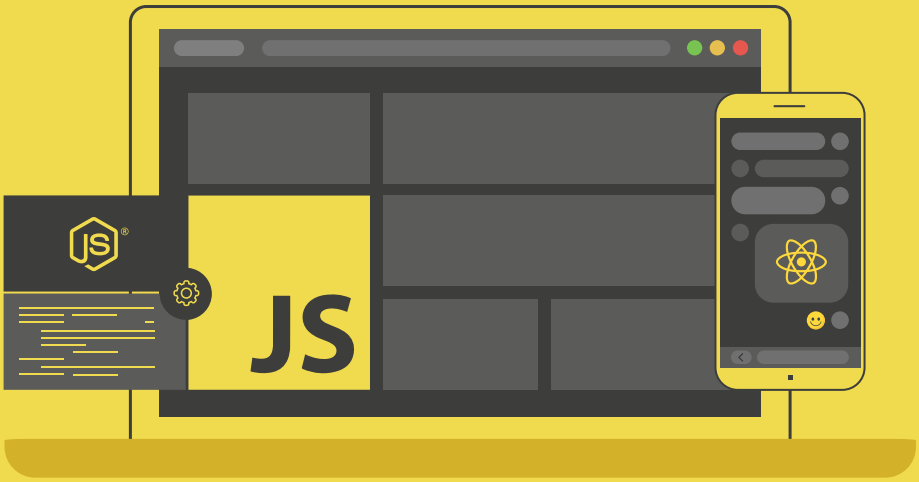
- خاصية `transform`

```
    div {
        -webkit-transform: rotate(45deg);
```

```
        -moz-transform: rotate(45deg);
        -ms-transform: rotate(45deg);
        -o-transform: rotate(45deg);
        transform: rotate(45deg);
    }
```

للمزيد من التفاصيل حول التحريك، ننصحك بقراءة هذه السلسلة، المرجع الشامل إلى التحريك عبر CSS.

دورة تطوير التطبيقات
باستخدام لغة JavaScript



احترف تطوير التطبيقات بلغة جافا سكريبت
انطلاقًا من أبسط المفاهيم وحتى بناء تطبيقات حقيقية

التحق بالدورة الآن

11. استعلامات الوسائط Media Queries

تسمح استعلامات الوسائط media queries بتطبيق أنماط CSS مختلفة بناءً على نوع جهاز العرض (شاشة أو طابعة أو غير ذلك) وحجمه، فيُحدَّد نوع الجهاز عن طريق نوع الوسط `media type`، بينما تُحدد مواصفات الجهاز الأخرى مثل اللون وأبعاد شاشة العرض باستخدام خاصيات الوسط `media features`.

الصيغة العامة لكتابة وسائط الاستعلامات media queries:

```
@media [...] {
    /* القواعد التي ستُطبق عند استيفاء شروط استعلامات الوسيط */
}
```

تحديد نوع الجهاز:

```
@media print {
    /* القواعد التي ستُطبق عند استيفاء شروط استعلامات الوسيط */
}
```

تحديد نوع وخصائص الجهاز:

```
@media screen and (max-width: 600px) {
    /* القواعد التي ستُطبق عند استيفاء شروط استعلامات الوسيط */
}
```

تحديد وضعية الجهاز (أفقي أم رأسي):

```
@media (orientation: portrait) {
    /* القواعد التي ستُطبق عند استيفاء شروط استعلامات الوسيط */
}
```

مثال:

```
@media screen and (min-width: 720px) {
    body {
        background-color: skyblue;
    }
}
```

تُحدد استعلامات الوسائط في المثال أعلاه شرطين لتطبيق قواعد CSS هما:

1. يجب عرض الصفحة على شاشة، أي ألَّا تكون مطبوعة أو ما شابه ذلك.
2. يجب أن يكون عرض الشاشة 720 بكسل على الأقل.

تُطبق القاعدة `background-color: skyblue` في حال تحقق الشرطين أعلاه ويُصبح لون الصفحة أزرق سماوي.

تطبق قواعد استعلامات الوسائط بشكل ديناميكي، أي أنه إذا عُرضت الصفحة على جهاز يستوفي شروطها فإنها تُطبق قواعد CSS الخاصة به، ويتوقف تطبيق هذه القواعد عند تغيير جهاز العرض.

في المثال أعلاه، إذا كان عرض شاشة العرض في البداية أقل من 720 بكسل تُطبق القاعدة ويُصبح لون الصفحة أزرق سماوي. إذا قَلَّص المستخدم عرض الشاشة (عن طريق تصغير حجم المتصفح مثلًا) إلى أقل من 720 بكسل يتوقف تطبيق القاعدة ويتغير لون الصفحة للون المحدد خارج media query.

ملخص القول، المعاملات التي تأخذها القاعدة `@media` هي باختصار:

المعامل	الوصف
`mediatype`	(اختياري) يُعرِّف الوسط (الجهاز) الذي تُعرض عليه الصفحة.
`not`	(اختياري) يُطبِّق أنماط CSS على كل الأجهزة عدا الجهاز المحدد به.
`media feature`	يُحدِّد خاصيات استعلامات الوسائط. الجدول التالي يبين الخاصيات المتاحة.

11.1 تحديد نوع الوسط أو الجهاز عبر media type

يستخدم المعامل `media type` لتحديد نوع الجهاز المُراد تطبيق قواعد CSS عليه، وهو معامل اختياري يُوضع مباشرة بعد التعريف `@media`. يوضح المثال التالي كيفية استخدامه:

```
@media print {
    html {
        background-color: white;
    }
}
```

تُعطي الشيفرة أعلاه صفحة HTML خلفية بيضاء عند الطباعة.

يُمكن إضافة المعامل `not` قبل نوع الجهاز لتطبيق قواعد CSS على كل الأجهزة عدا الجهاز المحدد، أو المعامل `only` والذي يُطبق قواعد CSS على الجهاز المحدد فقط كما هو موضح في الأمثلة التالية:

```
@media not print {
    html {
        background-color: green;
    }
}
@media only screen {
    .fadeInEffects {
        display: block;
    }
}
```

يوضح الجدول التالي أنواع الأجهزة:

نوع الجهاز أو الوسط	الوصف
`all`	كل الأجهزة.
`screen`	أجهزة الحاسوب.
`print`	الطابعات، وتستخدم لعمل نسخة مطبوعة من الصفحة.
`handheld`	تشمل الهواتف المحمولة والأجهزة ذات الشاشات الصغيرة.
`projection`	تستعمل لعمل العروض المرئية presentations، مثل جهاز الإسقاط projector
`aural`	الأجهزة الصوتية.
`braille`	أجهزة برايل اللمسية.
`embossed`	طابعات برايل.
`tv`	أجهزة التلفاز.
`tty`	الأجهزة ذات شبكات الأحرف الثابتة.

11.2 استعلامات الوسائط للشاشات الشبكية وغير الشبكية

على الـرغم من أن هـذا النـوع من اسـتعلامات لوسـائط يعمـل فقـط على مصـفحات المبنيـة على مُحـرِّك Webkit إلَّا انه قد يكون مفيدًا في بعض الأحيان، يوضح المثال التالي كيفية استعمالها:

```
/* ---- الشاشات غير الشبكية ---- */
@media screen
    and (min-width: 1200px)
    and (max-width: 1600px)
    and (-webkit-min-device-pixel-ratio: 1) {
        /* CSS rules */
}

/* ---- الشاشات الشبكية ---- */
@media screen
    and (min-width: 1200px)
    and (max-width: 1600px)
    and (-webkit-min-device-pixel-ratio: 2)
    and (min-resolution: 192dpi) {
        /* CSS rules */
}
```

معلومات أساسية:

- يوجـد نوعـان من البكسـلات المُسـتخدمة في شاشـات العـرض، البكسـلات المنطقيـة والبكسـلات الفيزيائية، وغالبًا ما تكون البكسلات الفيزيائية ثابتة لكل أجهزة العرض بينما تتغير البكسلات المنطقيـة مع وضوح الشاشة، فكلَّما زاد وضوح الشاشة تزيد جودة البكسلات.
- نسبة بكسلات الجهاز (device pixel ratio) هي نسبة البكسلات الفيزيائية إلى المنطقية. على سبيل المثال الأجهـزة MacBook Pro Retina، و iPhone4 لهـا نسـبة بكسـلات 2، أي أن عـدد البكسـلات الفيزيائية ضعف عدد البكسلات المنطقية.

11.3 خصائص الوسيط أو الجهاز media features

يوضح الجدول التالي خاصيات الوسيط أو الجهاز media features التي يمكن التعامل معها:

الخاصية	الوصف
`aspect-ratio`	نسبة ارتفاع الجهاز إلى عرضه.
`color`	تُشير إلى عدد البِتَّات (الثنائيات) المستعملة لتمثيل اللون في جهاز العرض المستهدف، وتكون قيمتها صفر إذا كان الجهاز غير ملون.
`color-index`	تشير إلى عدد المُدخلات في جدول ألوان جهاز العرض المستهدف.
`gird`	تُحدِّد ما إذا كان الجهاز شبكي أم انه يستخدم الصور النقطية.
`height`	تُحدد ارتفاع مساحة العرض الخاصة بالجهاز.
`max-width`	تُحدد أكبر عرض تُطبق عليه قواعد CSS.
`min-width`	تُحدد أقل عرض تُطبق عليه قواعد CSS.
`max-height`	تُحدد أكبر ارتفاع تُطبق عليه قواعد CSS.
`min-height`	تُحدد أقل ارتفاع تُطبق عليه قواعد CSS.
`monochrome`	تشير إلى عدد البِتَّات (الثنائيات) في البكسل الواحد على جهاز عرض ذو تدرج رمادي.
`orientation`	تُحدد اتجاه التدوير (أفقي أو رأسي) الذي تُطبَّق فيه قواعد CSS.
`resolution`	تشير إلى وضوح (كثافة بكسلات) جهاز العرض.
`scan`	تشير إلى عملية المسح الضوئي لأجهزة العرض التلفازية.
`width`	تُحدد عرض مساحة العرض الخاصة بالجهاز.

يوضح الجدول التالي بعض الخاصيات الملغية:

الخاصية	الوصف
`device-aspect-ratio`	تُطبق أنماط CSS فقط على الأجهزة التي تطابق نسبة ارتفاعها إلى عرضها النسبة المحددة بهذه الخاصية.
`max-device-width`	مماثلة لخاصية max-width لكنها تقيس عرض الشاشة الكُلي بدلًا عن عرض شاشة المتصفح.
`min-device-width`	مماثلة لخاصية min-width لكنها تقيس عرض الشاشة الكُلي بدلًا عن عرض شاشة المتصفح.
`max-device-height`	مماثلة لخاصية max-height لكنها تقيس ارتفاع الشاشة الكُلي بدلًا عن ارتفاع شاشة المتصفح.
`min-device-height`	مماثلة لخاصية min-height لكنها تقيس ارتفاع الشاشة الكُلي بدلًا عن ارتفاع شاشة المتصفح.

11.4 العرض Width مقابل إطار العرض Viewport

من المهم تحديد عرض الصفحة viewport ليكون موافقًا لعرض الجهاز device-width باستخدام البيانات الوصفية `meta tags`. توضع البيانات الوصفية داخل رمز

وتُكتب بالطريقة الموضحة في المثال التالي:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

تُحدد خاصية `width` في استعلامات الوسائط عرض المساحة الفعلية المستعملة لعرض المحتوى، بينما تحدد `view-port` عرض الجهاز نفسه.

تستخدم العديد من الأجهزة عدد من البكسلات الفيزيائية لتمثيل بكسل فيزيائي واحد. على سبيل المثال وضوح شاشة جهاز iPhone 6 Plus هو 2208 × 1242 لكنَّ مساحة العرض الفعلية هي 736 × 414، أي أن كل ثلاثة بكسلات منطقية تُمثل بكسل فيزيائي واحد.

عدم ضبط البيانات الوصفية meta tag ضبطًا صحيحًا يؤدي إلى عرض الصفحة بوضوحها الحقيقي مما يتسبب في ظهورها مُصغَّرة (نصوص وصور بحجم أصغر).

11.5 استخدام استعلامات الوسائط لاستهداف شاشات محددة

غالبًا ما تستخدم استعلامات الوسائط في إنشاء التصميم المستجيب لمواقع الويب Responsive web design، وذلك لأنها تمكن من تطبيق قواعد CSS مختلفة بناءً على حجم ونوع الجهاز المستخدم لعرض الصفحة.

```
@meida only screen and (min-width: 300px) and (max-width: 767) {
    /*
      تطبق القواعد المحددة في هذا الجزء فقط إذا كان عرض الشاشة بين 300 بكسل و 767 بكسل
    */
    .site-title {
        font-size: 80%;
    }
}

@meida only screen and (min-width: 768px) and (max-width: 1023) {
    /*
      تطبق القواعد المحددة في هذا الجزء فقط إذا كان عرض الشاشة بين 768 بكسل و 1023 بكسل
    */
```

```
    .site-title {
        font-size: 80%;
    }
}

@media only screen and (min-width: 1024) {
    /*
        تطبق القواعد المحددة في هذا الجزء فقط إذا كان عرض الشاشة أكبر من 1024 بكسل
    */
    .site-title {
        font-size: 120%;
    }
}
```

11.6 إنشاء استعلامات الوسائط عبر العنصر link

يُحمَّل ملف CSS المحدد في الرابط أعلاه في كل الحالات ولكن لا يُطبق إلا إذا كان عرض شاشة الجهاز أكـبر من 600 بكسل.

```
<link rel="stylesheet" media="min-width: 600px" href="example.css" />
```

11.7 تقسيم الصفحات عبر استعلامات الوسائط

يوضح الجدول التالي القيم التي يمكن استخدامها عبر استعلامات الوسائط:

القيمة	الوصف
`auto`	تُضاف فواصل الصفحات تلقائيًا.
`always`	تُضيف فواصل الصفحات دائمًا.
`avoid`	تتجنب إضافة فواصل الصفحات ما أمكن.
`left`	تُضيف فواصل الصفحات، وتعتبر الصفحة التالية صفحة يُسرى.
`right`	تُضيف فواصل الصفحات، وتعتبر الصفحة التالية صفحة يُمنى.
`initial`	تُرجع القيمة الابتدائية للخاصية.
`inherit`	ترث قيمة الخاصية من العنصر الأب.

تستخدم هذه الخواص عند طباعة صفحات الويب، لذلك من الشائع استخدامها مع استعلامات الوسائط

مثال:

```
@media print {
    p {
        page-break-inside: avoid;
    }
    h1 {
        page-break-before: always;
    }
    h2 {
        page-break-after: avoid;
    }
}
```

ملاحظات:

- القاعدة الأولى تمنع إضافة فواصل الصفحات داخل الفقرة، وبمعنى آخر أنها تمنع فصل الفقرة على صفحتين.
- القاعدة الثانية تضيف تبدأ صفحة جديدة قبل كل عناصر h1 أي أن كل عنصر h1 سيبدأ في صفحة جديدة.
- القاعدة الثالثة تمنع بدء صفحة جديدة بعد عناصر h2.

11.8 وسائط الميزات Feature Queries

يوضح الجدول التالي المعاملات التي يمكن استخدامها مع وسائط الميزات:

المعامل	الوصف
`(property: value)`	تكون صحيحة إذا كانت الخاصية المحددة مدعومة في المتصفح الذي تُعرض عليه الصفحة.
`and`	تكون صحيحة إذا وفقط إذا تحقق الشرطان معًا.
`not`	تكون صحيحة إذا لم يتحقق الشرط.
`or`	تكون صحيحة إذا تحقق أحد الشروط.
`(...)`	مجموعة الشروط.

11.9 استخدام @supports

تستعمل `@supports` لمعرفة ما إذا كانت الخاصية مدعومة في المتصفح أم لا وتطبيقها إذا كانت مدعومة، مثال:

```
@supports (display: flex) {
    .my-container {
    display: flex;
    }
}
```

11.10 الاستعلام عن عدد من الميزات

تستخدم العمليات `and` و`not` و`or` للاستعلام عن عدد من الميزات.

```
@supports (transform: translateZ(1px)) and (transform-style: preserve-
3d) and (perspective: 1px) {
    /* Probably do some fancy 3d stuff here */
}


@supports (display: flex) or (display: table-cell) {
    /* Will be used if the browser supports flexbox or display: table-
cell */
}
@supports not (-webkit-transform: translate(0, 0, 0)) {
    /* Will *not* be used if the browser supports -webkit-transform:
translate(...) */
}
```

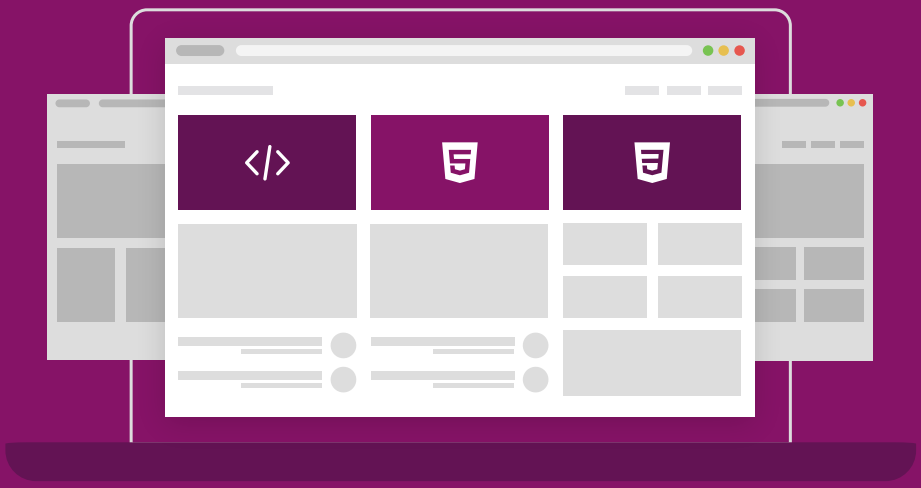
ولتسهيل قراءة الشيفرة، من الجيد وضع الجمل المنطقية بين قوسين كما هو موضح في المثال التالي

```
@supports ((display: block) and (zoom: 1)) or ((display: flex) and
(not (display: table-cell))) or
(transform: translateX(1px)) {
    /* ... */
}
```

تُطبَّق الشيفرة المحددة داخل هذا الاستعلام إذا كان:

1. يدعم المتصفح الخاصيتين `display: block` و `zoom: 1`، أو إذا كان
2. يدعم الخاصية `display: flex` ولا يدعم الخاصية `display: table-cell`، أو إذا كان
3. يدعم الخاصية `transform: translate(1px)`.

دورة تطوير واجهات المستخدم



ابدأ مسارك المهني كمطور واجهات المواقع والمتاجر الإلكترونية
فور انتهائك من الدورة

التحق بالدورة الآن

12. مواضيع متقدمة في CSS

12.1 الدوال Functions

12.1.1 الدالة calc

تسمح الدالة `calc()` بإجراء عمليات حسابية على قيم خاصيات CSS، ويمكن استخدامها في أي موضع يُسمَح فيه باستخدام أنواع البيانات `<length>` و `<percentage>` و `<angle>` و `<time>` و `<number>` و `<integer>`.

مثال على استخدام دالة `calc()` لحساب عرض الحاوية:

```
#div1 {
    position: absolute;
    left: 50px;
    width: calc(100% - 100px);
    border: 1px solid black;
    background-color: yellow;
    padding: 5px;
    text-align: center
}
```

استخدام الدالة `calc()` لتحديد موضع صورة الخلفية:

```
background-position: calc(50% + 17px) calc(50% + 10px), 50% 50%;
```

استخدام الدالة `calc()` لحساب ارتفاع العنصر:

```
height: calc(100% - 20px);
```

لمزيد من المعلومات انظر توثيق calc.

12.1.2 الدالة attr

تستخدم الدالة `attr()` للحصول على قيمة إحدى خاصيات العنصر المُحدَّد واستخدامها في أنماط CSS، ويمكن استعمالها أيضًا على العناصر الزائفة pseudo-elements وفي هذه الحالة ستُستخدَم قيمة خاصية العنصر الأصل.

مثال:

- ملف HTML

```
<blockquote data-mark=""></blockquote>
```

- ملف CSS

```
blockquote[data-mark]::before,
blockquote[data-mark]::after {
    content: attr(data-mark);
}
```

لمزيد من المعلومات انظر توثيق attr.

12.1.3 الدالة var

تستخدم الدالة `var()` للوصول لمتغيرات CSS.

مثال:

```
:root {
    --primary-color: blue;
}

selector {
    color: var(--primary-color);
}
```

لمزيد من المعلومات انظر توثيق var.

12.1.4 الدالة radial-gradient

تستخدم الدالة `radial-gradient()` لإنشاء تـدرج لـوني بين لـونين أو أكـثر على شـكل أشـعة تبـدأ من المركز، ويمكن أن يكون شكلها دائريًا أو على شكل قطع ناقص.

مثال:

```
radial-gradient(red, orange, yellow);
```

لمزيد من المعلومات انظر توثيق radial-gradient.

12.1.5 الدالة linear-gradient

تستخدم الدالة `linear-gradient()` لإنشاء تدرج لوني بين لونين أو أكثر على امتداد خط مستقيم،

مثال:

```
linear-gradient( 0deg, red, yellow 50%, blue);
```

لمزيد من المعلومات انظر توثيق linear-gradient.

12.2 المتغيرات: الخاصيات المُخصصة في CSS

أسـماء الخاصـيات الـتي تُسـبَق بشـرطتين `--` مثـل `--example-name` تُمثِّل الخاصـيات المخصصـة custom properties التي تُسنَد لها قيمةٌ يمكن إعادة استخدامها في المستند عبر الدالة `var()`.

مثالًا من الشائع إعادة استخدام نفس اللون في عدة أمـاكن في الصـفحة، لـذلك من الأفضـل إنشـاء متغـير يحمل قيمة هذا اللون واستخدامه مما يُسهِّل البحث عنه وتعديله في المستقبل.

```
:root {
    --red: #b00;
    --blue: #4679bd;
    --grey: #ddd;
}

.Bx1 {
    color: var(--red);
    background-color: var(--grey);
    border: 1px solid var(--red);
}
```

مثال:

```
:root {
    --W200: 200px;
    --W100: 10px;
}

.Bx2 {
    width: var(--W200);
    height: var(--W200);
    margin: var(--W10);
}
```

12.2.1 إعادة تعريف قيم المتغيرات

مثال:

- ملف HTML

```
  <a class="button">Button Green</a>
  <a class="button button_red">Button Red</a>
  <a class="button">Button Hovered On</a>
```

- ملف CSS

```
  .button {
      --color: green;
      padding: .5rem;
      border: 1px solid var(--color);
      color: var(--color);
  }

  .button:hover {
      --color: blue;
  }

  .button-red {
      --color: red;
  }
```

النتيجة:

12.2.2 قواعد استخدام المتغيرات في CSS

يجب أن تحتوي أسماء المتغيرات على الحروف وعلامة (-) فقط، مثال:

```
/* Invalid variables names */
--123color: blue;
--#color: red;
--bg_color: yellow
--$width: 100px;

/* Valid variable names */
--color: red;
--bg-color: yellow
--width: 100px
```

أسماء المتغيرات حساسة لحالة الأحرف، مثال:

```
/* The variable names below are all different variables */
--pcolor: ;
--Pcolor: ;
--pColor: ;
```

يجب أن يكون لكل متغير قيمة (حرف المسافة (space) قيمة صالحة للمتغير)، مثال:

```
/* Invalid */
--color:;

/* Valid */
--color: ; /* space is assigned */
```

متغيرات CSS لا تدعم التسلسل، أي لا يمكن استخدام الدالة var() داخل دالة أخرى، مثال:

```
/* Invalid */
.logo{
    --logo-url: 'logo';
```

```
    background: url('assets/img/' var(--logo-url) '.png');
}

/* Invalid */
.logo{
    --logo-url: 'assets/img/logo.png';
    background: url(var(--logo-url));
}

/* Valid */
.logo{
    --logo-url: url('assets/img/logo.png');
    background: var(--logo-url);
}
```

يجب وضع الوحدات داخل تعريف المتغير:

```
/* Invalid */
--width: 10;
width: var(--width)px;

/* Valid */
--width: 10px;
width: var(--width);

/* Valid */
--width: 10;
width: calc(1px * var(--width)); /* multiply by 1 unit to convert */
width: calc(1em * var(--width));
```

12.2.3 استخدام المتغيرات مع استعلامات الوسائط

يُمكن إعادة تعريف قيم المتغيرات في استعلامات الوسائط للحصول على صفحات متجاوبة.

مثال:

- ملف HTML

```
<div></div>
```

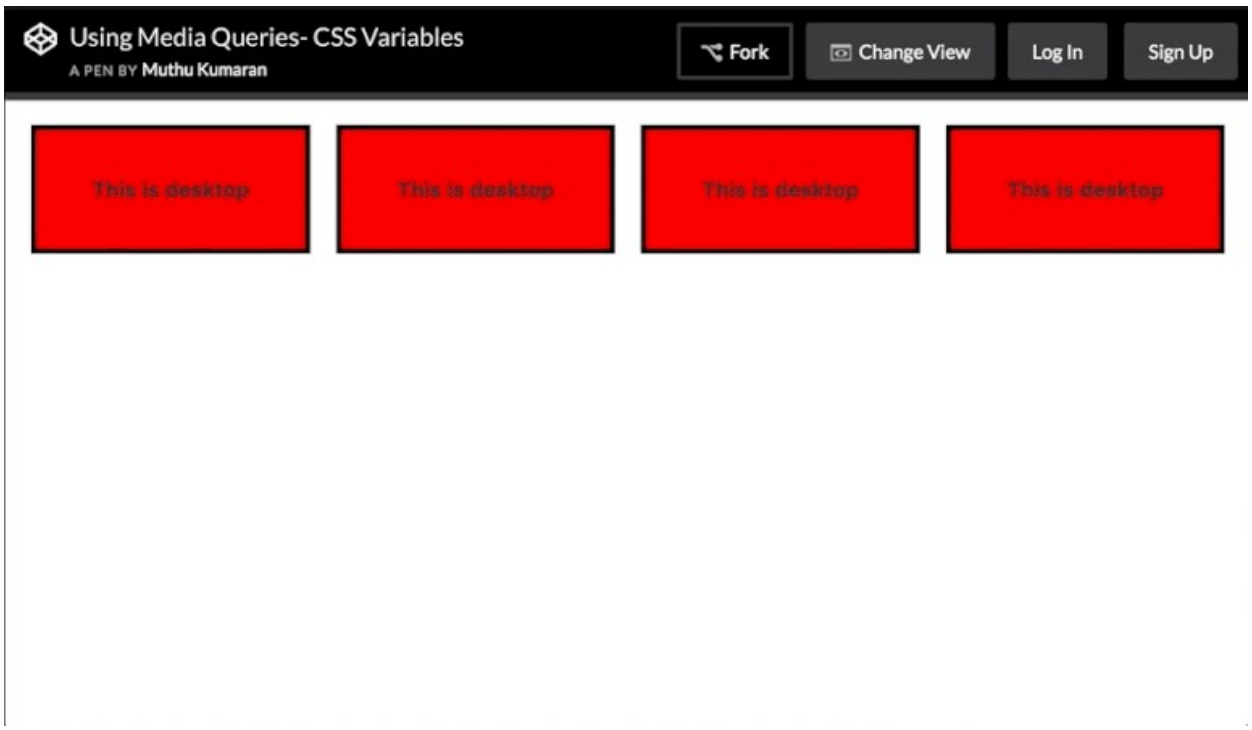
```
<div></div>
<div></div>
<div></div>
```

- ملف CSS

```
:root{
    --width: 25%;
    --content: 'This is desktop';
}

@media only screen and (max-width: 767px){
    :root{
    --width:50%;
    --content: 'This is mobile';
    }
}

@media only screen and (max-width: 480px){
    :root{
    --width:100%;
    }
}

div{
    width: calc(var(--width) - 20px);
    height: 100px;
}
div:before{
    content: var(--content);
}

/* Other Styles */
body {
    padding: 10px;
}
div{
```

```
        display: flex;
        align-items: center;
        justify-content: center;
        font-weight:bold;
        float:left;
        margin: 10px;
        border: 4px solid black;
        background: red;
    }
```

النتيجة (الصورة متحركة اضغط عليها لرؤيتها كاملةً في المتصفح):



اطَّلع على تجربة حيَّة لهذا المثال على CodePen.

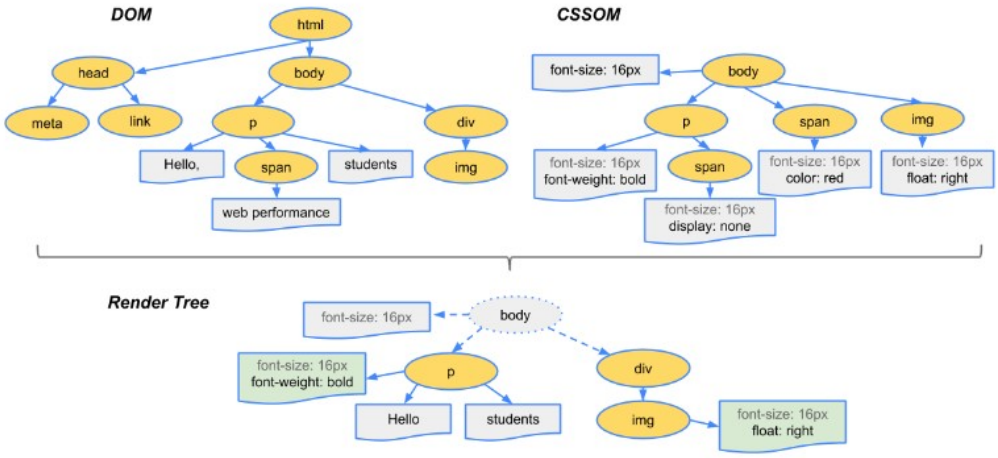
12.3 نموذج الكائنات CSSOM

يحدد المتصفح جميع الرموز في ملف CSS ويحولها إلى عُقد مرتبطة في شكل بنية شجرية، وتُكَوِّن مجموعة هذه العقد ما يُعرف بنموذج كائنات CSS أو CSSOM وهو اختصار CSS Object Model .

لعرض أي صفحة ويب، ينفذ المتصفح الخطوات التالية:

1. يفحص المتصفح وثيقة HTML ويبني DOM.
2. يفحص المتصفح وثيقة CSS ويبني نموذج الكائنات CSSOM.

3. يدمج المتصفح بين DOM و CSSOM لإنشاء شجرة العرض التي تُمثل الصفحة.



إليك المثال التالي حول إضافة صورة الخلفية background-image باستخدام نموذج الكائنات CSSOM.

أولًا يجب الحصول على نقاط مرجعية تمثل بداية ونهاية القواعد في وثيقة CSS.

```
var stylesheet = document.stylesheet[0].cssRules;
var end = stylesheet.length - 1;
```

ومن ثم تُضاف القاعدة background-image للعنصر body في نهاية وثيقة CSS.

```
stylesheet.insertRule("body { background-image:
url('http://cdn.sstatic.net/Sites/stackoverflow/img/favicon.ico'); }",
end);
```

12.4 أنماط التصميم CSS Design Patterns

أنماط التصميم Design Patterns هي أساليب نموذجية تحل المشاكل التي تتكرر كثيرًا في تصميم البرمجيات، ويمكن تشبيهها بالمخططات التي تستخدمها -وتُعدَّلها أيضًا- من أجل إصلاح مشكلة بعينها في شيفرتك.

بعض أنماط التصميم الشائعة في CSS:

- BEM
- OOCSS
- SMACSS

12.4.1 نمط BEM

هي منهجية صُممت من قبل شركة التكنولوجيا الروسية Yandex، ومن ثم اكتسبت شعبية كبيرة بين مبرمجين الويب في الولايات المتحدة و أوروبا الغربية ومنها إلى بقية العالم، وتُشير BEM إلى الكلمات Block، وElements وModifiers على التوالي.

- Blocks: أو الكتل، وهي العناصر أو الكيانات المُستقلة ولها معنى محدد، مثل العناصر `header`، و `container`، و`menu`، وغيرها.
- Elements: أو العناصر، وهي أجزاء من الكُتل والتي ليس لها معنى محدد، مثل `menu item`، و`list` `item` وغيرها.
- Modifiers: أو الصفات التعريفية، وتستخدم لتغيير طريقة عرض أو سلوك العنصر، مثل الخواص `disabled`، و`highlighted`، و`checked`، وغيرها.

هناك بعض القواعد لتطبيق منهجية أو نمط BEM وهي:

- أنماط الكتل لا تعتمد بأي شكل من الأشكال على بقية العناصر في الصفحة.
- يجب أن تكون أسماء الكُتل قصيرة، مع تجنب استخدام المحارف (-) و (_).
- يجب أن تأخذ أسماء مُحددات العناصر الشكل `blockname__elementname`.
- يجب أن تأخذ أسماء مُحددات الصفات التعريفية الشكل `blockname--modifiername` أو `blockname__element-name--modifiername`.

مثال:

```
<style>
form { } // Block
.form--theme-xmas { } // Block + modifier
.form--simple { } // Block + modifier
.form__input { } // Block > element
.form__submit { } // Block > element
.form__submit--disabled { } // Block > element + modifier
</style>

<form class="form form--theme-xmas form--simple">
    <input class="form__input" type="text" />
```

```
    <input class="form__submit form__submit--disabled" type="submit"
/>
</form
```

للتفصيل أكثر حول BEM، ننصحك بالإطلاع على مقال «دليلك إلى منهجية BEM».

أكبر موقع توظيف عن بعد في العالم العربي

ابحث عن الوظيفة التي تحقق أهدافك وطموحاتك
المهنية في أكبر موقع توظيف عن بعد

تصفح الوظائف الآن

13. تنسيقات المتصفحات المخصصة وأداءها

تُضيف المتصفحات تلقائيًا بعض الأنماط على عناصر HTML، وتختلف هذه الأنماط باختلاف المتصفح مما يتسبب في ظهور الصفحة بأشكال مختلفة عند عرضها على متصفحات مختلفة، ولتفادي هذا الأمر يجب توحيد أنماط المتصفحات، ومن الشائع استخدام مكتبة normalize.css والتي تضبط بعض الأنماط وتُصحح بعض الأخطاء الشائعة في بعض المتصفحات.

13.1 الفرق بين normalize.css و reset.css

الفرق الأساسي بين مكتبة normalize.css ومكتبة reset.css هو أن الأولى تضبط بعض الأنماط لتوحيدها في جميع المتصفحات، بينما تُزيل الثانية جميع الأنماط الافتراضية من العناصر، مما يعني أنك ستضطر لكتابة جميع القواعد من الصفر.

مثال على استخدام Eric Meyer's reset.css:

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,GoalKicker.com – CSS
Notes for Professionals 224
```

```
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
```

مزيد من المعلومات حول Eric Meyer's reset.css.

مثال على استخدام normalize.css:

```
/**
* 1. توحيد نوع الخط في جميع المتصفحات
* 2. تصحيح ارتفاع السطر وتوحيده في جميع المتصفحات
* 3. Safari منع تعديل حجم الخط عند تغيير اتجاه العرض في متصفح
*/
/* Document
========================================================================
==== */
html {
    font-family: sans-serif; /* 1 */
    line-height: 1.15; /* 2 */
    -ms-text-size-adjust: 100%; /* 3 */
    -webkit-text-size-adjust: 100%; /* 3 */
}

/*
========================================================================
==== */
/**
* إزالة الهوامش
*/
body {
```

```
    margin: 0;
}
/**
* تصحيح طريقة العرض
*/
article,
aside,
footer,
header,
nav,
section {
    display: block;
}
/**
* تصحيح حجم الخط والهوامش
*/
h1 {
    font-size: 2em;
    margin: 0.67em 0;
}
```

13.2 وضع التباين العالي

أُضيف محدد الوسائط `-ms-high-contrast` في متصفحات Internet Explorer ابتداءً من الإصدار العاشر والإصدارات التي تليه، مما يسمح للمبرمج بضبط الأنماط بناءً على ما إذا كان المستخدم يستخدم وضع التباين العالي أم لا.

للمحدد `-ms-high-contrast` ثلاث حالات هي:

- `active`
- `black-on-white`
- `white-on-black`

مثال:

```
@media screen and (-ms-high-contrast: active), (-ms-high-contrast:
black-on-white) {
    .header {
```

```
        background: #fff;
        color: #000;
    }
}
```

في هذا المثال يتغير لون خلفية العنصر `header` إلى اللون الأبيض ولون الكتابة للون الأسود إذا كان وضع التباين العالي مُفعَّلًا وفي الحالة `black-on-white`.

مثال:

```
@media screen and (-ms-high-contrast: white-on-black) {
    .header {
        background: #000;
        color: #fff;
    }
}
```

في هذا المثال يتغير لون خلفية العنصر `header` إلى اللون الأسود ولون الكتابة للون الأبيض إذا كان وضع التباين العالي مُفعَّلًا وفي الحالة `white-on-black`.

لمزيد من المعلومات زر توثيق مايكروسوفت.

13.3 أداء المتصفحات

13.3.1 استخدام الخاصيات transform و opacity لتجنب تنبيه التخطيط

تغيير بعض الخاصيات في CSS يُنبِّه المتصفح لإعادة حساب القيم النهائية للأنماط و التخطيط، مما يتسبب في تبطيء تحميل الصفحة.

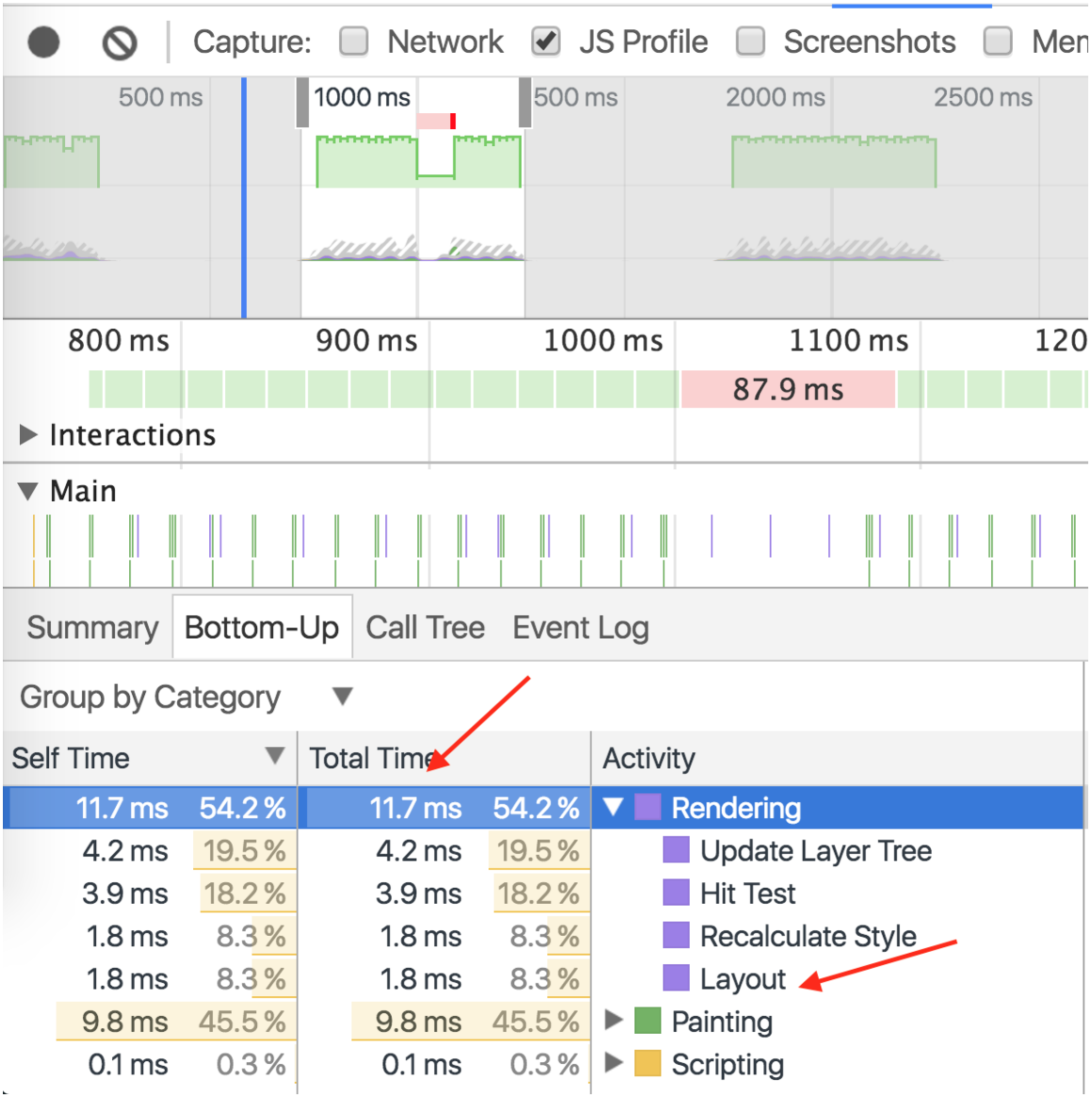
مثال على التحريك باستخدام `left` و `top`:

```
#box {
    left: 0;
    top: 0;
    transition: left 0.5s, top 0.5s;
    position: absolute;
    width: 50px;
    height: 50px;
    background-color: gray;
```

```
}

#box.active {
    left: 100px;
    top: 100px;
}
```

اطلع على تجربة حية لهذا المثال على JSFiddle.

تستغرق الصفحة حوالي 11.7 ملي ثانية للتحميل، ومن ثم تستغرق 9.8 ملي ثانية لإعادة رسم الصندوق في الموضع الجديد.

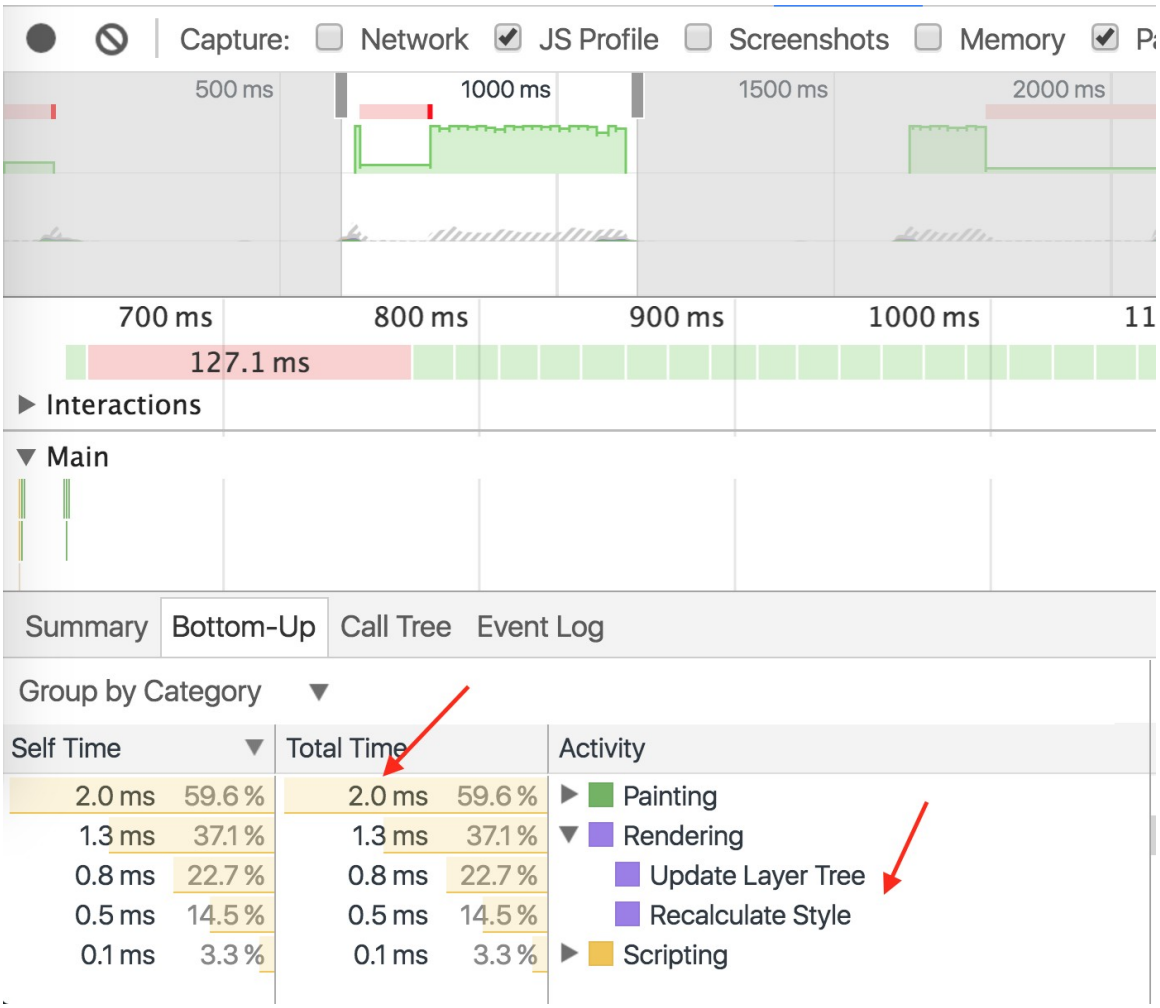


ولتحسين الأداء وتسريع تحميل الصفحة يجب استخدام الخواص `transform` و`opacity` للحصول على نفس النتيجة كما هو موضح في المثال التالي:

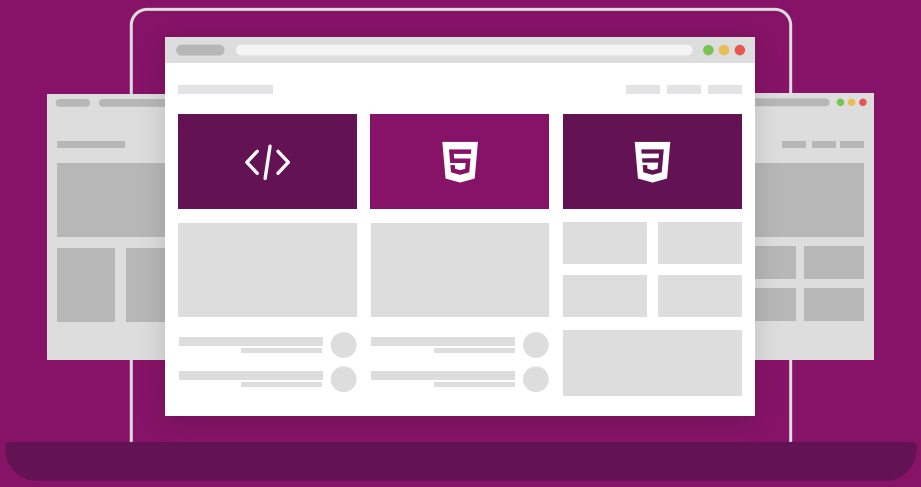
```
#box {
    left: 0;
    top: 0;
    position: absolute;
    width: 50px;
    height: 50px;
    background-color: gray;
    transition: transform 0.5s;
    transform: translate3d(0, 0, 0);
}
#box.active
    transform: translate3d(100px, 100px, 0);
}
```

اطلع على تجربة حية لهذا المثال على JSFiddle.

تستغرق الصفحة حوالي 1.3 ملي ثانية للتحميل، ومن ثم تستغرق 2.0 ملي ثانية لإعادة رسم الصندوق في الموضع الجديد.


Capture:
Network
JS Profile
Screenshots
Memory
500 ms
1000 ms
1500 ms
2000 ms
700 ms
800 ms
900 ms
1000 ms
127.1 ms
Interactions
Main
Summary
Bottom-Up
Call Tree
Event Log
Group by Category
Self Time
Total Time
Activity
2.0 ms 59.6 %
2.0 ms 59.6 %
Painting
1.3 ms 37.1 %
1.3 ms 37.1 %
Rendering
0.8 ms 22.7 %
0.8 ms 22.7 %
Update Layer Tree
0.5 ms 14.5 %
0.5 ms 14.5 %
Recalculate Style
0.1 ms 3.3 %
0.1 ms 3.3 %
Scripting

دورة تطوير واجهات المستخدم



مميزات الدورة

- بناء معرض أعمال قوي بمشاريع حقيقية
- وصول مدى الحياة لمحتويات الدورة
- تحديثات مستمرة على الدورة مجانًا
- شهادة معتمدة من أكاديمية حسوب
- إرشادات من المدربين على مدار الساعة
- من الصفر دون الحاجة لخبرة مسبقة

اشترك الآن

أحدث إصدارات أكاديمية حسوب