



CSS Fundamentals

.NET

CSS (Cascading Style Sheets) is a language that describes the style that the browser will give to HTML elements on screen. CSS rulesets are stored in a .css file called a stylesheet.

[HTTPS://WWW.W3SCHOOLS.COM/CSS/CSS_INTRO.ASP](https://www.w3schools.com/css/css_intro.asp)

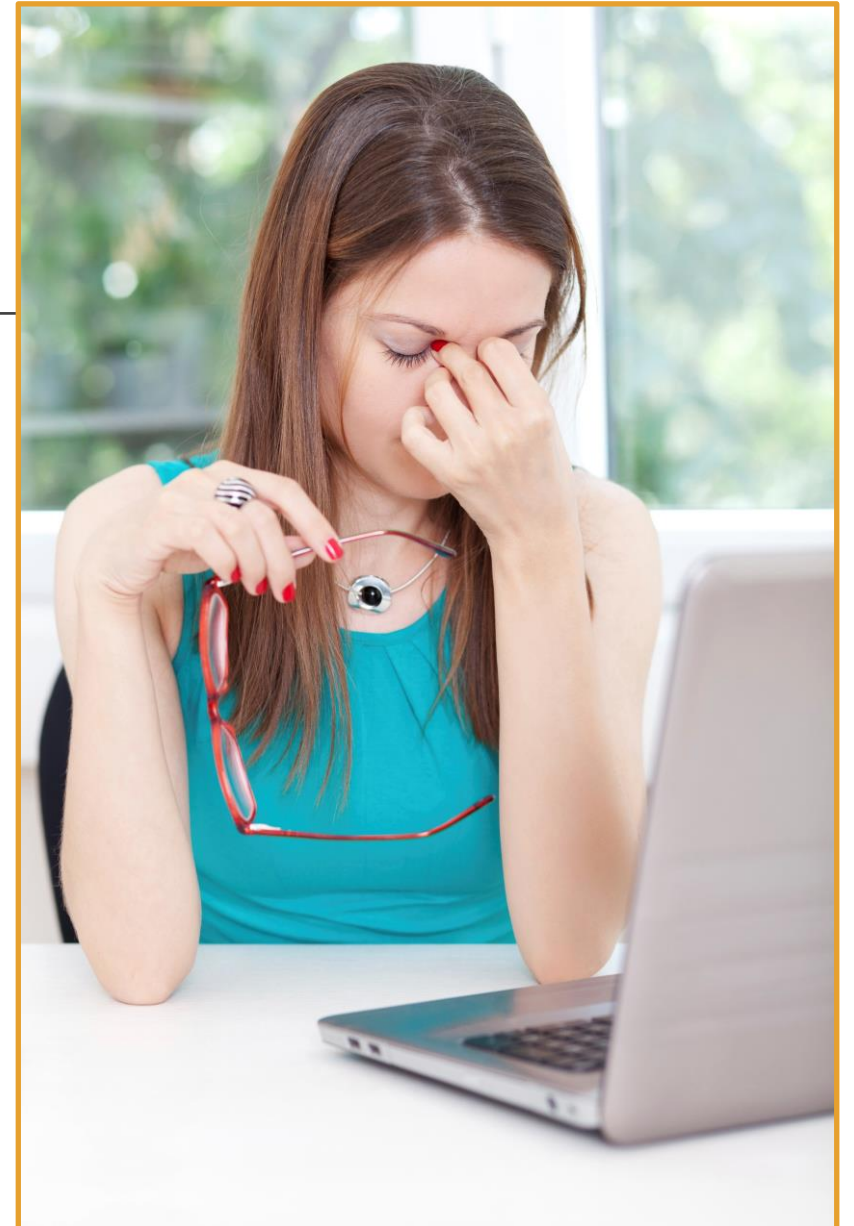
Preparation

- Create a directory, **HtmlAndCssPractice**, in your personal repo.
- Right-Click the directory to open it in VS Code.
- Create a .html file, **HtmlPractice.html**.
- In VS Code, download the extension 'Live Server' by Ritwick Dey
- Reload your VS Code window with CTRL + Shift + P (to open Command Palette) and Type 'reload window'.
- Type 'doc' + tab to auto-fill the **.html** page with the HTML template.
- Add `<link rel="stylesheet" type="text/css" href="CssPractice.css" />` to the `<head>` section of your .html just below the `<title>`.
- Create a file, **CssPractice.css**, in the same folder as your **HtmlPractice.html** file.
- Right-Click on the **HtmlPractice.html** file in the explorer window and click 'Open with Live Server'. This will open the .html file in your browser and automatically update the browser any time your **HtmlPractice.html** file is saved.

CSS – Why use CSS?

<https://www.w3.org/Style/CSS/learning>

- **HTML** is not intended to be responsible for style and formatting.
- Style Formatting **tags** were added with HTML 3.2.
- Development of large websites became a laborious process with fonts and color information added to every page.
- The **World Wide Web Consortium (W3C)** created CSS to enable **separation of concerns** between structure and presentation of documents (HTML + CSS).



CSS - Adding Styling (1 / 3) - External

https://www.w3schools.com/css/css_howto.asp

Styling Precedence – *Inline* overrides *Internal*. *Internal* overrides *External*. *External* overrides *Browser default* styling.

External **.css** file (Recommended):

Include a reference to the relative location of the **.css** file inside a **<link>** element in the **<head>** section.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Benefits of External CSS.

- Separation of concerns - HTML and CSS are in separate documents, you don't have to mix them
- Reusability - the same stylesheet can be used to style many HTML files
- Central location - Change styling in one place!
- Readability - HTML is less cluttered, and CSS styling is easier to understand.

CSS - Adding Styling (2/3) - Internal

https://www.w3schools.com/css/css_howto.asp

Internal CSS :

- defined inside the **<style>** element inside the **<head>** section.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
<body>
```

CSS - Adding Styling (3 / 3) - Inline

https://www.w3schools.com/css/css_howto.asp

Inline CSS :

- used to apply a unique style to a single *element*,
- add a **style** attribute inside relevant *element* opening.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

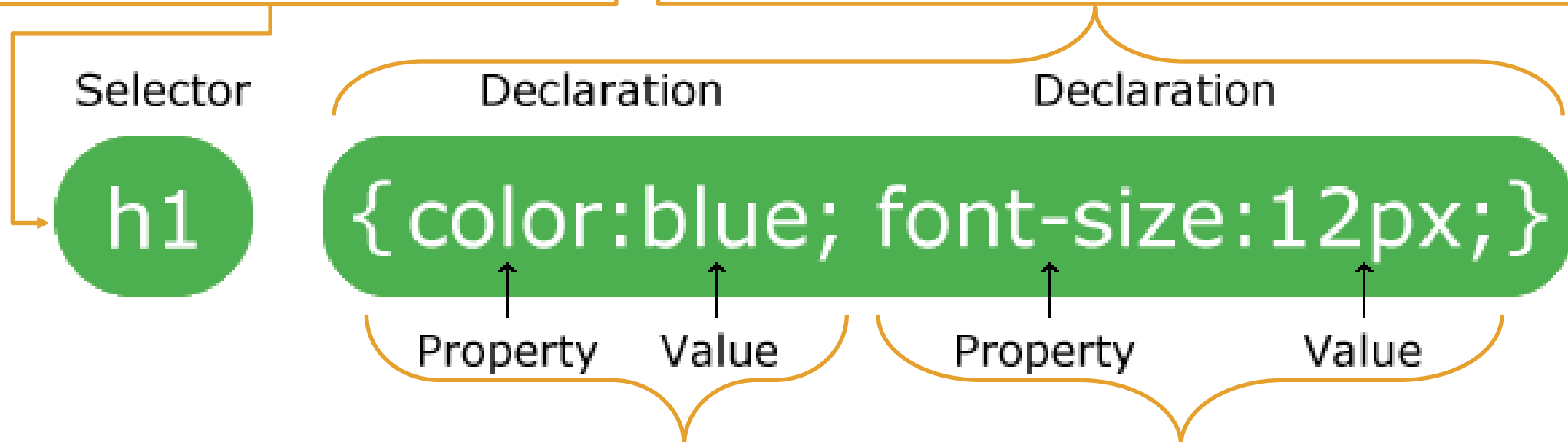
CSS – Syntax

https://www.w3schools.com/css/css_syntax.asp

A **rule** or "rule set" is a statement that tells browsers how to render particular elements on an HTML page.

The **selector** points to the HTML element you want to style.

The **declaration** block is bounded by '{ }', contains **declarations** separated by ';'.
Declaration Declaration



Each declaration includes a CSS **property** name and a **value**, separated by a colon.

CSS – Selector

https://www.w3schools.com/css/css_selectors.asp

Selector Type	Example RuleSet	Description
element Selector	p { declarations }	Selects HTML elements based on the tag name.
id Selector	#id1 { declarations }	Selects HTML elements based on the id attribute name.
class Selector	.center { declarations }	Selects HTML elements based on the class attribute name.
Universal Selector	* { declarations }	Selects all HTML elements on the page.
Selector List	h1, h2, p { declarations }	Selects all indicated HTML tags in the document.
Concatenation	p .center { declarations }	Selects all <u>center</u> class elements inside a <p> element.

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

```
#para1 {  
  text-align: center;  
  color: red;  
}
```

```
.center {  
  text-align: center;  
  color: red;  
}
```

```
p.center {  
  text-align: center;  
  color: red;  
}
```

```
h1, h2, p {  
  text-align: center;  
  color: red;  
}
```

```
* {  
  text-align: center;  
  color: blue;  
}
```

CSS – Pseudo-Selector

https://www.w3schools.com/css/css_pseudo_classes.asp

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Pseudo-classes_and_pseudo-elements

There are two types of *Pseudo-Selectors*.

<i>Pseudo-Class</i> Selectors	<i>Pseudo-Element</i> Selectors
Used to define a special state of an element , like display a visited link differently or change an elements color when the mouse hovers over it.	A CSS pseudo-element is used to style specified parts of an element , like the first letter (or line) of an element.

CSS – Pseudo Classes

https://www.w3schools.com/css/css_pseudo_classes.asp

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Pseudo-classes_and_pseudo-elements

A ***pseudo-class*** is a keyword at the end of a CSS ***selector*** that is used to specify a ***style*** for the selected ***element*** but only when it is in a certain state.

- when the mouse hovers over the element.
- when a checkbox is checked.

Use a colon (:) in front of the ***pseudo-class***. These are some of the over 30 ***pseudo-classes*** available.

- :active
- :visited
- :checked
- :disabled
- :first
- :nth-child()
- :hover

CSS – Pseudo Classes Examples

https://www.w3schools.com/css/css_pseudo_classes.asp



This is a link

This link was clicked

**The mouse is over
this link**



This link is selected

```
<head>
<style>
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
</style>
</head>
<body>
```

CSS – Pseudo Classes with HTML Classes

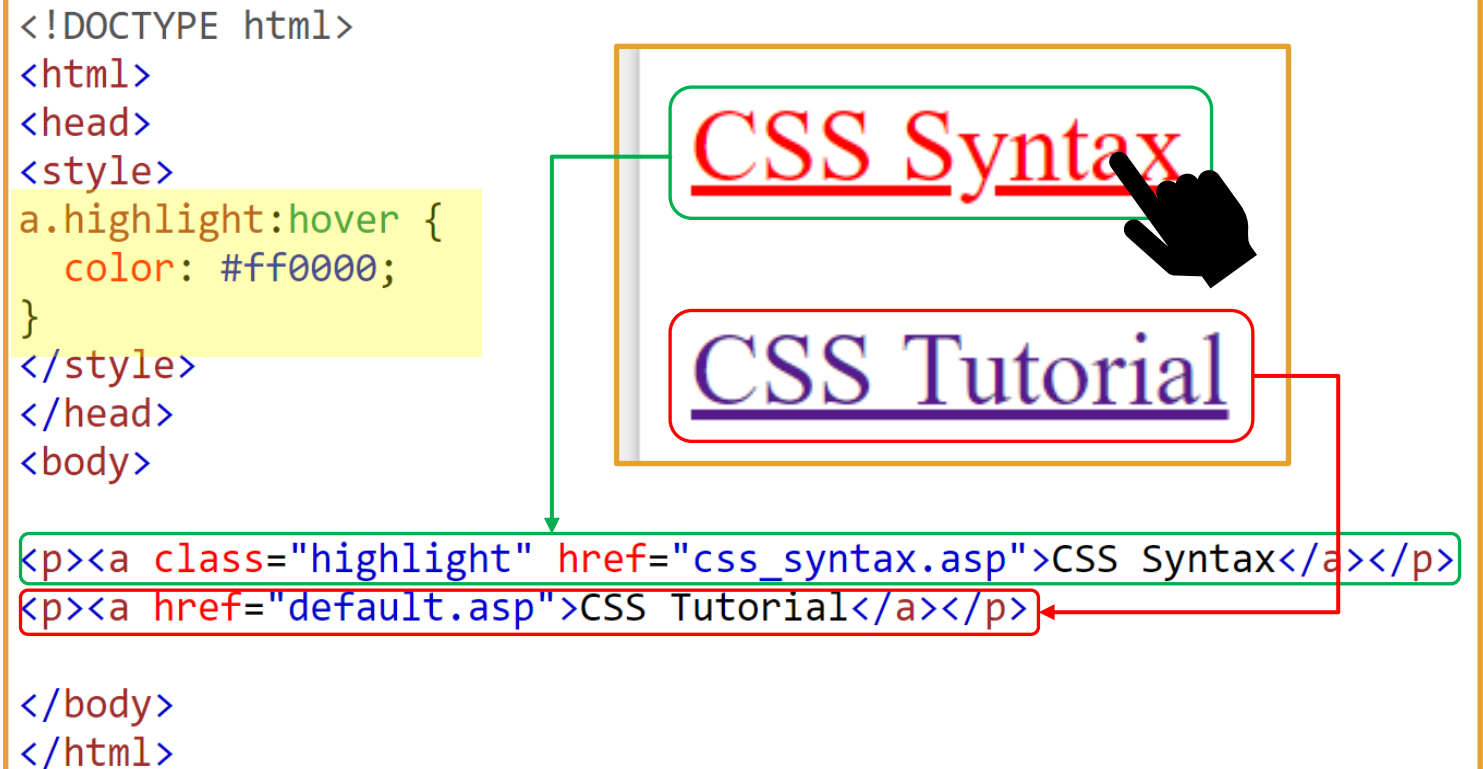
https://www.w3schools.com/css/css_pseudo_classes.asp

Pseudo Classes can be placed on HTML elements to be specific as to which elements you want to style.

```
<!DOCTYPE html>
<html>
<head>
<style>
a.highlight:hover {
  color: #ff0000;
}
</style>
</head>
<body>

<p><a class="highlight" href="css_syntax.asp">CSS Syntax</a></p>
<p><a href="default.asp">CSS Tutorial</a></p>

</body>
</html>
```



CSS – Pseudo Elements

https://www.w3schools.com/css/css_pseudo_elements.asp

A CSS *pseudo-element* is used to style specified parts of an *element*. To use it, place two colons (::) between the *element* and the *pseudo element*.

- Style just the first letter (or line) of an element.
- Insert other content before or after the content of an element

- `::before`
- `::after`
- `::first-line`
- `::first-letter`

This example
formats the
first line of the
text in all <p>
elements:

```
p::first-line {  
  color: #ff0000;  
  font-variant: small-caps;  
}
```

[illegible]

YOU CAN USE THE `::FIRST-LINE` PSEUDO-ELEMENT TO ADD A SPECIAL effect to the first line of a text. Some more text. And even more, and more, and more, and more, and more, and more, and more, and more, and more, and more.

CSS – Combinators

https://www.w3schools.com/css/css_combinators.asp

A **combinator** describes the relationship between two **selectors**. The **selector** part of a CSS **RuleSet** can contain more than one **selector**. Between the **selectors**, we can insert one of four available **combinators**.

Selector	Symbol	Description
descendant selector	(space)	Matches any descendant element (nested inside) of a specified element. This includes grandchildren, etc
child selector	>	Selects only elements that are the children of a specified element.
Adjacent (next) sibling selector	+	Selects the designated element if it occurs immediately after this selector. Sibling elements have the same parent element.
general sibling selector	~	Selects all sibling elements that follow this element.

CSS Combinator Example - Descendent

https://www.w3schools.com/css/css_combinators.asp

In this example, ***Descendent*** means any `<p>` element nested inside a `<div>` to unlimited depth.

```
<style>
div p {
  background-color: yellow;
}
</style>
</head>
<body>
```

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>
```

```
<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>
```

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.

CSS Combinators Example - Child

https://www.w3schools.com/css/css_combinators.asp

In this example, **Child** means any `<p>` element inside the `<div>` but only one level deep. This means children of children are not styled.

```
<head>
<style>
div > p {
  background-color: yellow;
}
</style>
</head>
<body>
```

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
  <p>Paragraph 4 in the div.</p>
</div>
```

```
<p>Paragraph 5. Not in a div.</p>
<p>Paragraph 6. Not in a div.</p>
```

```
</body>
```

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4 in the div.

Paragraph 5. Not in a div.

Paragraph 6. Not in a div.

<!-- not Child but Descendant -->

CSS Combinators Example – Adjacent Sibling

https://www.w3schools.com/css/css_combinators.asp

In this example,
Adjacent Sibling
means the first **<p>**
sibling immediately
following a **div**.

```
<head>
<style>
div + p {
  background-color: yellow;
}
</style>
</head>
<body>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
</div>

<p>Paragraph 3. Not in a div.</p>
<p>Paragraph 4. Not in a div.</p>

</body>
```

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. Not in a div.

Paragraph 4. Not in a div.

CSS Combinators Example – General Sibling

https://www.w3schools.com/css/css_combinators.asp

In this example,
General Sibling means
all **<p>** siblings
occurring after the **div**.

```
<head>
<style>
div ~ p {
  background-color: yellow;
}
</style>
</head>
<body>

<p>Paragraph 1.</p>

<div>
  <p>Paragraph 2.</p>
</div>

<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>

</body>
```

Paragraph 1.

Paragraph 2.

Paragraph 3.

Some code.

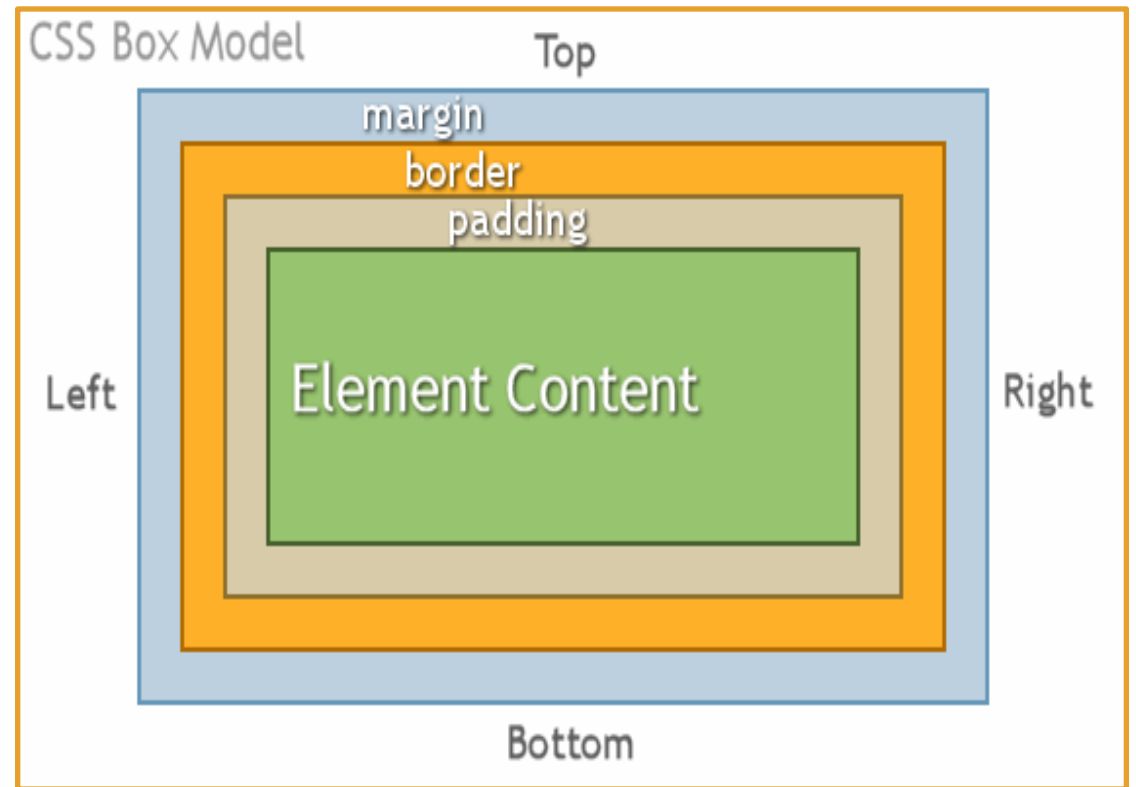
Paragraph 4.

CSS - Box Model

https://www.w3schools.com/css/css_boxmodel.asp

The CSS '**Box Model**' is the way that all HTML *elements* are defined. The **Box Model** is made up of 4 different concentric boxes.

- **Content** - this is the actual text or image
- **Padding** - space between content and border
- **Border** - space between the padding and the margin. The border width, style, and color properties may be set
- **Margin** - the invisible space between the end of one element and the start of another
- Remember:
 - C.P.B.M. =>Cows Poop Brown...MOO!



CSS - Box Model Example

https://www.w3schools.com/css/css_boxmodel.asp

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
</style>
</head>
<body>

<h2>Demonstrating the Box Model</h2>

<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

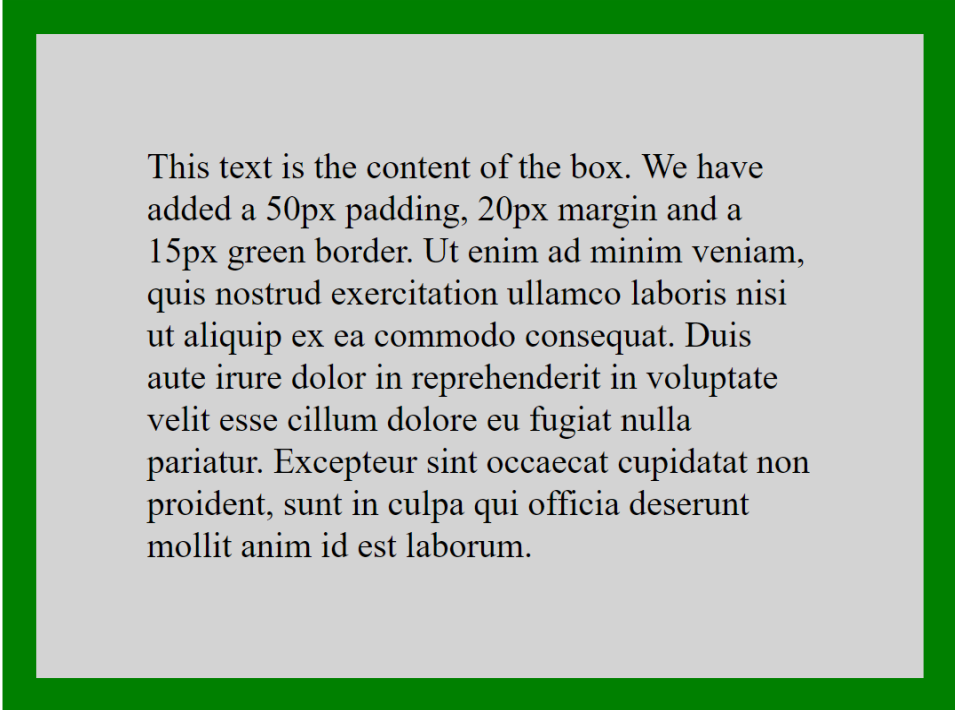
<div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</div>

</body>
</html>
```

Demonstrating the Box Model

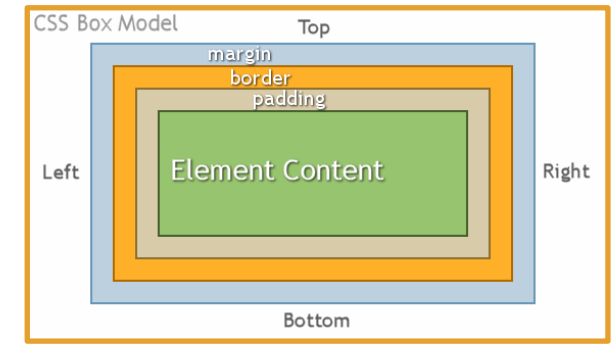
The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.



This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. 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.

CSS Box Model – Example

https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand_properties



In order to correctly set the width and height of an element, you must know how the box model works.

```
div {  
  background-color: lightgrey;  
  width: 300px;  
  border: 15px solid green;  
  padding: 50px;  
  margin: 20px;  
}
```

300px (width)
+ 30px (left + right border)
+ 100px (left + right padding)
+ 40px (left + right margin)
= 470px total element width

Margin is the only part of the Box Model that can have a negative value.

margins, and the actual content.

margin

border

padding

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. 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.

width

CSS – Units of measurement


https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units

There are two ways to give measurement in CSS. Both ***Absolute*** and ***Relative*** have appropriate uses.

Absolute Measurements		Relative Measurements	
px	pixels	%	percentage relative to parent element's width
mm	millimeters	em	1em is the same as the font size(scales with it) of the current element
cm	centimeters	rem	stands for "root em" and works the same except refers to the base font size
In	inches	vh, vw	these are 1/100th of the height and width of the viewport, respectively
Pt	points(1/72 of an inch)		
Pc	picas (12 points)		

CSS – Colors

https://developer.mozilla.org/en-US/docs/Web/HTML/Applying_color

There are four ways  can be set in CSS.

Colors	
keywords	'red', 'blue', 'aquamarine', etc
hex value	The # sign followed by six hex numbers (0-F) with each pair representing rgb values for up to 256 different values of each.
rgb(r,g,b)	This function specifies values from 0 to 255 for red (r), green (g), and blue (b). Ex. rgb(224,176,255) . There is an optional alpha (transparency) value, but this isn't used much.
hsl(h,s,l)	This function to specifies hue, saturation, and lightness values, 0 to 255, to define a color. Ex. hsl(240,100%,50%) . There is an optional alpha (transparency) value, but this isn't used much.

CSS - Rule Set Conflicts

<https://www.w3.org/Style/LieBos2e/enter/Overview.en.html>

What happens if two CSS rules conflict with each other?

The order that ***selectors*** are chosen for styling depends on three things:

- Importance
- Specificity
- Source order

What is this cascade?

By combining importance, origin, specificity, and the source order of the style concerned, the CSS cascade assigns a weight to each declaration. This weight is used to determine exactly, and without conflict, which style declarations should be applied to a specific element: the declaration with the highest weight takes precedence.

<http://reference.sitepoint.com/css/cascade>

CSS – Specificity (1 / 4)

<https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity>

Specificity is the means by which browsers decide which CSS **property** values are chosen to style an **element**.

Specificity is a weight given to each RuleSet. When multiple RuleSets have the same weight, the last occurring RuleSet wins.

A **Ruleset** with a more specific selector combination wins over a less specific combination.

Specificity only comes into play when the same **element** is targeted by multiple declarations. Directly targeted **elements** will always take precedence over inherited styling.

```
div#test span
{ color: green; }

div span
{ color: blue; }

span
{ color: red; }
```

```
<div id="test">
  <span>
    Text
  </span>
</div>
```

Although, the rule for **blue** is more specific than the rule for **red**, the rule for **green** is the most specific and will be applied no matter what order the **RuleSets** occur in.

CSS – Specificity of Selectors (2/4)

<https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity>

These selectors are listed by increasing specificity:

- Type selectors (e.g., `h1`) and pseudo-elements (e.g., `::before`).
- Class selectors (e.g., `.example`), attributes selectors (e.g., `type="radio"`) and pseudo-classes (e.g., `:hover`).
- ID selectors (e.g., `#example`).

The following CSS styles determine which rule will be applied.

- `div { color: red; }` /* least specific - won't be applied */
- `#yellow { color: yellow; }` /* most specific - will be applied */
- `.green { color: green; }` /* in the middle - won't be applied */

```
#specificityId {  
  background-color: ■ red;  
}  
  
div {  
  background-color: ■ whitesmoke;  
}  
  
.specificityClass {  
  background-color: ■ brown;  
}
```

```
<div>  
  <p class="specificityClass"  
    id="specificityId"  
    style="background-color:lightgreen;"  
    >This is a specificity example</p>  
</div>
```

CSS – Specificity and Source Order (3/4)




<https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity>

If two selectors have equal importance level and specificity, then the latter rule wins over earlier one.

For example:

- `div { color: blue; } /* This comes first, will lose */`
- `div { color: red; } /* comes last, so is applied */`

Internal styling wins over external styling.

```
#specificityId {  
  background-color:  red;  
}  
  
div {  
  background-color:  whitesmoke;  
}  
  
.specificityClass {  
  background-color:  brown;  
}
```

```
<div>  
  <p class="specificityClass"  
    id="specificityId"  
    style="background-color:lightgreen;">  
    This is a specificity example</p>  
</div>
```

CSS – Specificity and !important (4 / 4)

<https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity>

The **!important** flag can be used to raise a RuleSet's specificity and ensure that a style always gets applied

div { color: red !important; } <!-- - always applied - ->

The only way to override an **!important** flag is to apply another **!important** flag on a selector with the same specificity later in the document or on a selector with greater specificity.

*!important is not best practice. Use Specificity instead.

CSS –*Inherit* Keyword

https://www.w3schools.com/cssref/css_inherit.asp

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance

<https://developer.mozilla.org/en-US/docs/Web/CSS/inheritance>

```
<!DOCTYPE html>
<html>
<head>
<style>
span {
  color: blue;
  border: 1px solid black;
}
```

```
.extra span {
  color: inherit;
}
</style>
</head>
<body>
```

```
<div>
  Here is <span>a span element</span> which is blue, as span
  elements are set to be.
</div>
```

```
<div class="extra" style="color:green">
  Here is <span>a span element</span> which is green, because
  it inherits from its parent.
</div>
```

```
<div style="color:red">
  Here is <span>a span element</span> which is blue, as span
  elements are set to be.
</div>
```

```
</body>
</html>
```

Here is a span element which is blue, as span elements are set to be.
Here is a span element which is green, because it inherits from its parent.
Here is a span element which is blue, as span elements are set to be.

The width, margin, padding, and border of an *element* is not automatically inherited from its parent. Some *elements* inherit but others do not.

The *inherit* keyword specifies that a property shall *inherit* its styling from its parent element.

The *inherit* keyword can be used for any CSS property, and on any HTML *element*.

CSS – display Property

https://www.w3schools.com/cssref/pr_class_display.asp

The ***display*** property defines the display type of an ***element***. This overrides whatever default display values the property has.

The difference between **display** and **visibility** is that **display: none** removes the element from the page completely whereas **visibility: hidden** means that the tag is still given space on the page, but just isn't seen.

- **inline** - display on the same line as other elements
- **block** - start on a new line and
- **flex** - make the element a flexbox
- **none** - do not display the element at all

```
p.ex1 {display: none;}  
p.ex2 {display: inline;}  
p.ex3 {display: block;}  
p.ex4 {display: inline-block;}
```

CSS – ***position*** Property (1/2)

https://www.w3schools.com/cssref/pr_class_position.asp

The ***position*** property is an important property related to the layout of a webpage. It governs the flow of ***elements***. You can even take ***elements*** out of the normal flow of the page to appear where you want.

Important position property values:

- ***static*** – (default) the element is placed in the normal document flow
- ***relative*** - the element occurs in its normal place, but can be moved around with ***top***, ***bottom***, ***left***, and ***right*** properties
- ***absolute*** - removes the element from the normal flow of the document, fixes it in place relative to the html element or nearest positioned ancestor
- ***fixed*** - fixes element in place relative to browser window.
- ***sticky*** - hybrid between ***relative*** and ***fixed***, allowing relative positioning until scrolled to a 'threshold' point

CSS – *position* Property (2/2)

https://www.w3schools.com/cssref/pr_class_position.asp

```
<!DOCTYPE html>
<html>
<head>
<style>
h2.pos_left {
  position: relative;
  left: -30px;
}

h2.pos_right {
  position: relative;
  left: 50px;
}
</style>
</head>
<body>
```

```
<h1>The position Property</h1>
```

```
<p>Relative positioning moves an element RELATIVE to its original position.</p>
```

```
<p>The style "left: -30px;" subtracts 30 pixels from the element's original left position.</p>
```

```
<p>The style "left: 50px;" adds 50 pixels to the element's original left position.</p>
```

```
<h2 class="pos_left">This heading is moved left according to its normal position</h2>
```

```
<h2 class="pos_right">This heading is moved right according to its normal position</h2>
```

```
</body>
```

```
</html>
```

The position Property

Relative positioning moves an element RELATIVE to its original position.

The style "left: -30px;" subtracts 30 pixels from the element's original left position.

The style "left: 50px;" adds 50 pixels to the element's original left position.

This heading is moved left according to its normal position

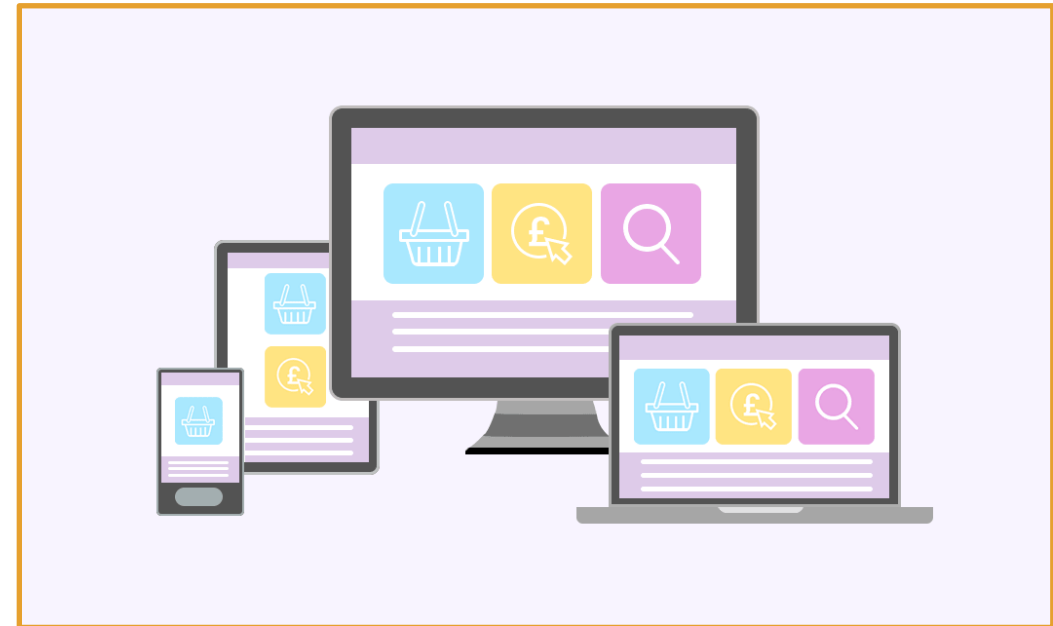
This heading is moved right according to its normal position

CSS - Responsive Web Design

https://developer.mozilla.org/en-US/docs/Tools/Responsive_Design_Mode

Responsive Web Design is a way of designing web pages so that they render well on any window, screen, or device size. **RWD** has become increasingly important due to the share of internet traffic conducted from mobile phones and devices. There are a few ways to design web pages to change based on the size of the screen.

- Use fluid grid systems - make your element containers sized with **relative** units like percentages instead of absolute units
- Use flexible images which are also sized relatively to prevent rendering outside their container
- Use media queries to change CSS based on the window size



CSS - Media Queries

https://en.wikipedia.org/wiki/Media_queries

https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries

Media queries are a way of adapting the display of CSS for different screen sizes. They can be used to respond to different media types and features. An example of the syntax is shown below:

A media type can be declared in the **head** of an HTML document using the "**media**" **attribute** inside of a **<link>** element. The value of the "media" **attribute** specifies how on each device the linked document will be displayed

```
@media screen and (display-mode: fullscreen) {  
  /* Code in here only applies to screens in fullscreen */  
}
```

```
@media all and (orientation: landscape) {  
  /* Code in here only applies in landscape orientation */  
}
```

```
@media screen and (min-device-width: 500px) {  
  /* Code in here only applies to screens equal or greater than 500 pixels wide */  
}
```

CSS – ‘at-rules’

<https://developer.mozilla.org/en-US/docs/Web/CSS/At-rule>

A *media query* is one type of **@rule**. These tell CSS what to do in some way. The most important ones are:

- **@media** - used for media queries
- **@font-face** - used for defining a custom font
- **@keyframes** - used for animations in CSS

```
@font-face {  
  font-family: "Open Sans";  
  src: url("/fonts/OpenSans-Regular-webfont.woff2") format("woff2"),  
       url("/fonts/OpenSans-Regular-webfont.woff") format("woff");  
}
```



Details about CSS3

<https://developer.mozilla.org/en-US/docs/Archive/CSS3>
<https://makeawebsitehub.com/css3-mega-cheat-sheet/>

CSS Level 2 needed 9 years to reach the “Recommendation” status. This was because a few secondary features held back the whole specification.

In order to accelerate the standardization of non-problematic features, the [CSS Working Group](#) of the W3C divided CSS in smaller components called ***modules***.

Each of these ***modules*** is now an independent part of the language and moves towards standardization at its own pace.

While some ***modules*** are already W3C Recommendations, others are still early working drafts. New ***modules*** are added when new needs are identified.

CSS3 is still evolving but some features are available such as:

- rounded corners, shadows, gradients, transitions and animations, flexbox layouts