# CSS Fundamentals

.NET

CSS (Cascading Style Sheets) is a language that describes the style HTML elements are to be displayed in on screen. **CSS** is stored in a stylesheet, which is a **.css** file

CSS – Why use CSS?

- *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).
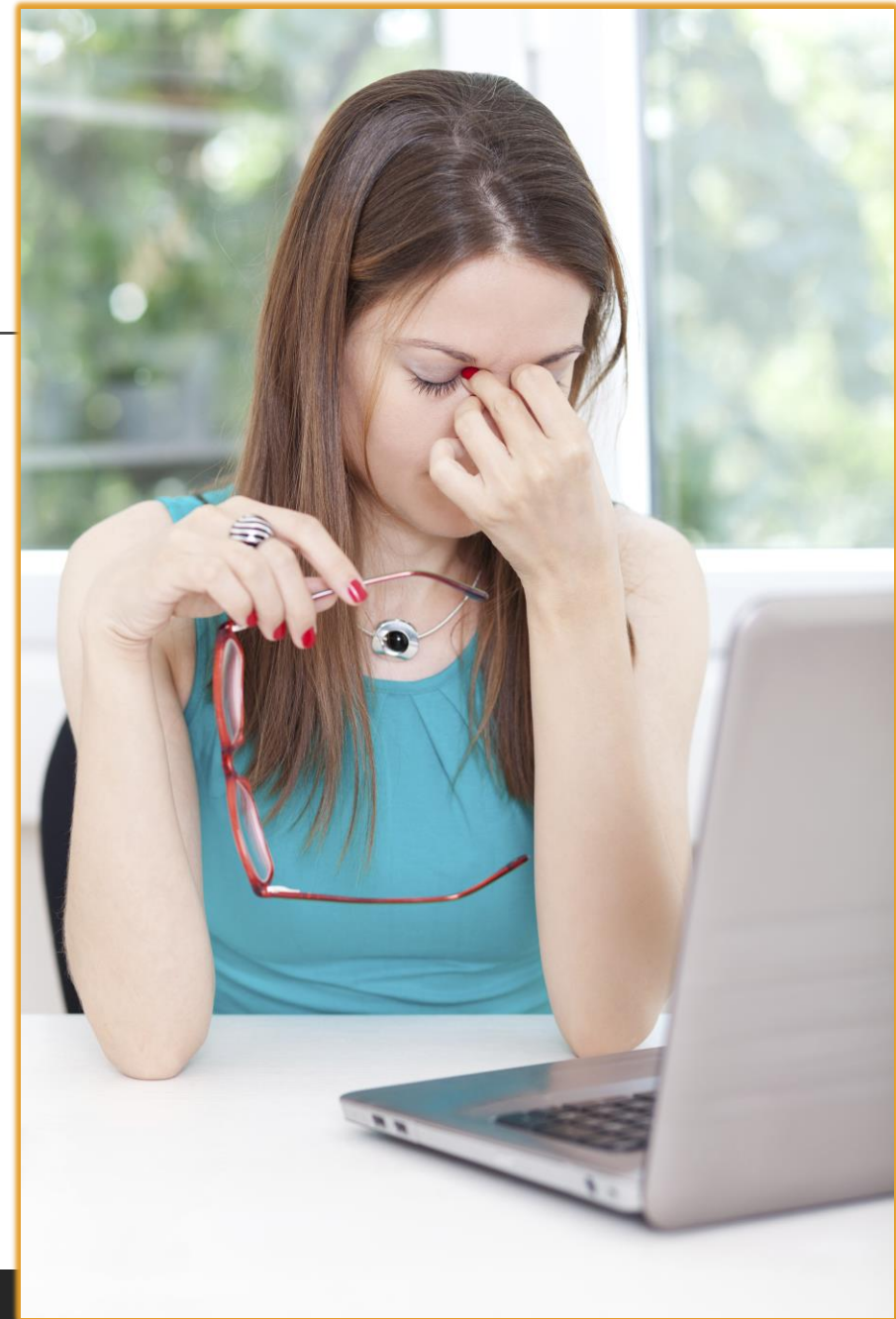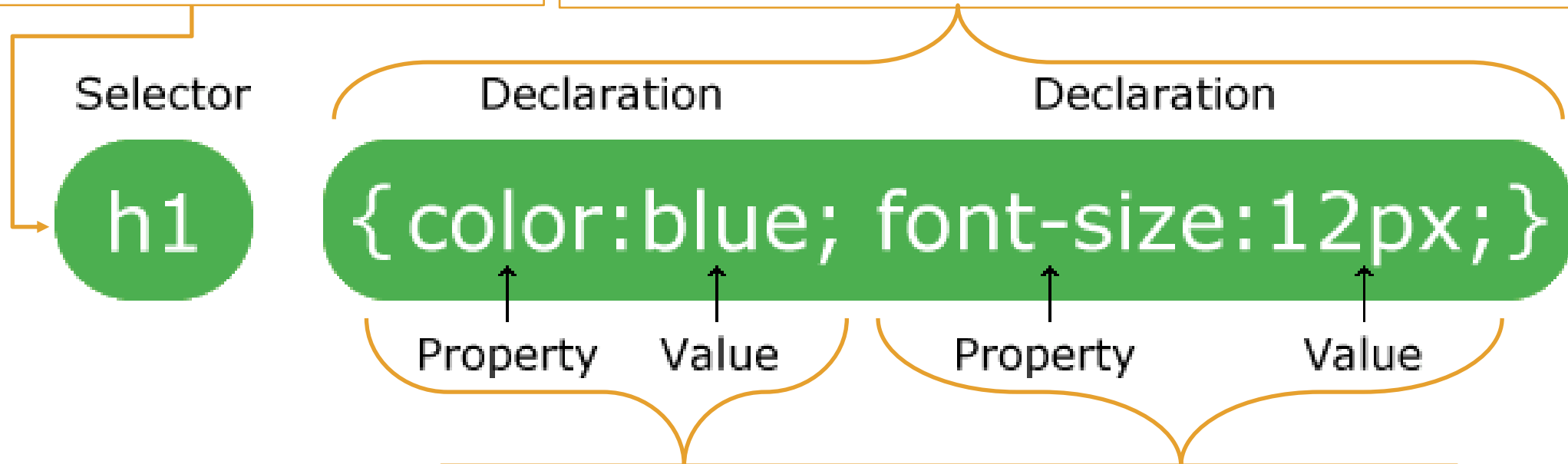
CSS – Syntax

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 begins with '{', contains *declarations* separated by ';', and ends with '}'.

Selector

Declaration   Declaration

h1   { color:blue; font-size:12px;}

Property   Value   Property   Value

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

CSS – Selector

Selector Type	Example	Description
element Selector	p { declarations }	Selects HTML elements based on the element name.
id Selector	#para1 { declarations }	Selects HTML elements based on the id name.
class Selector	.center { declarations }	Selects HTML elements based on the class attribute.
Universal Selector	* { declarations }	Selects all HTML elements on the page.
Selector List	h1, h2, p { declarations }	Selects all named HTML elements in the document.
Concatenation	p .center { declarations }	Selects all <u>center</u> class elements inside a <u><p></u> element in the document.

```
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-Selectors

There are two types of *Pseudo-Selectors*.

Pseudo-Class Selectors and *Pseudo-Element* Selectors.

- A CSS *pseudo-element* is used to style specified parts of an *element*, like the first letter (or line) of an element.
- A CSS *pseudo-class* is 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.

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 specific keyword at the end of a *selector* used to specify that you want to *style* the selected *element* but only when it is in a certain state.

- when the user is hovering the mouse over the element
- when a checkbox is disabled or checked.

The syntax is to use a colon (:) to precede the *pseudo-class*. These are among over 30 *pseudo-classes* available.

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

CSS –
Pseudo Classes Examples

This is a link

This is a link

This is a link

This is a link

```
<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

Pseudo Classes can be combined with HTML Classes to be more specific as to which elements you want to style.

```html
<!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 Syntax

CSS Tutorial

CSS – Pseudo Elements

A CSS pseudo-element is used to style specified parts of an element. You must have two '::' between the element and the pseudo element.

- Style the first letter, or line, of an element
- Insert 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;
}
```

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, and more, and more, and more, and more.

CSS – Combinators

A *combinator* describes the relationship between two **selectors**. The *selector* part of a CSS *selector/declaration* block can contain more than one *selector*. Between the *selectors*, we can include a *combinator*. There are four combinators to choose from.

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 sibling selector	(+)	Selects the designated element if it occurs immediately after this selector. Sibling elements must have the same parent element.
general sibling selector	(~)	Selects all sibling elements.

CSS – Combinators - Descendent

```
<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 – Child Combinators

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

```html
<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> <!-- not Child but Descendant -->
  <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.

CSS – Adjacent Sibling Combinators

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

```
<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.

General Sibling Combinators

```html
<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

The CSS *box model* is the way that all HTML *elements* are defined. There are 4 different concentric boxes, which are

- *Content* - this is the actual text or image
- *Padding* - space between content and border
- *Border* - space between padding and margin - its width, style, and color properties may be set
- *Margin* - usually invisible space between the end of one element and the start of another
- The size of each edge is usually given in pixels.
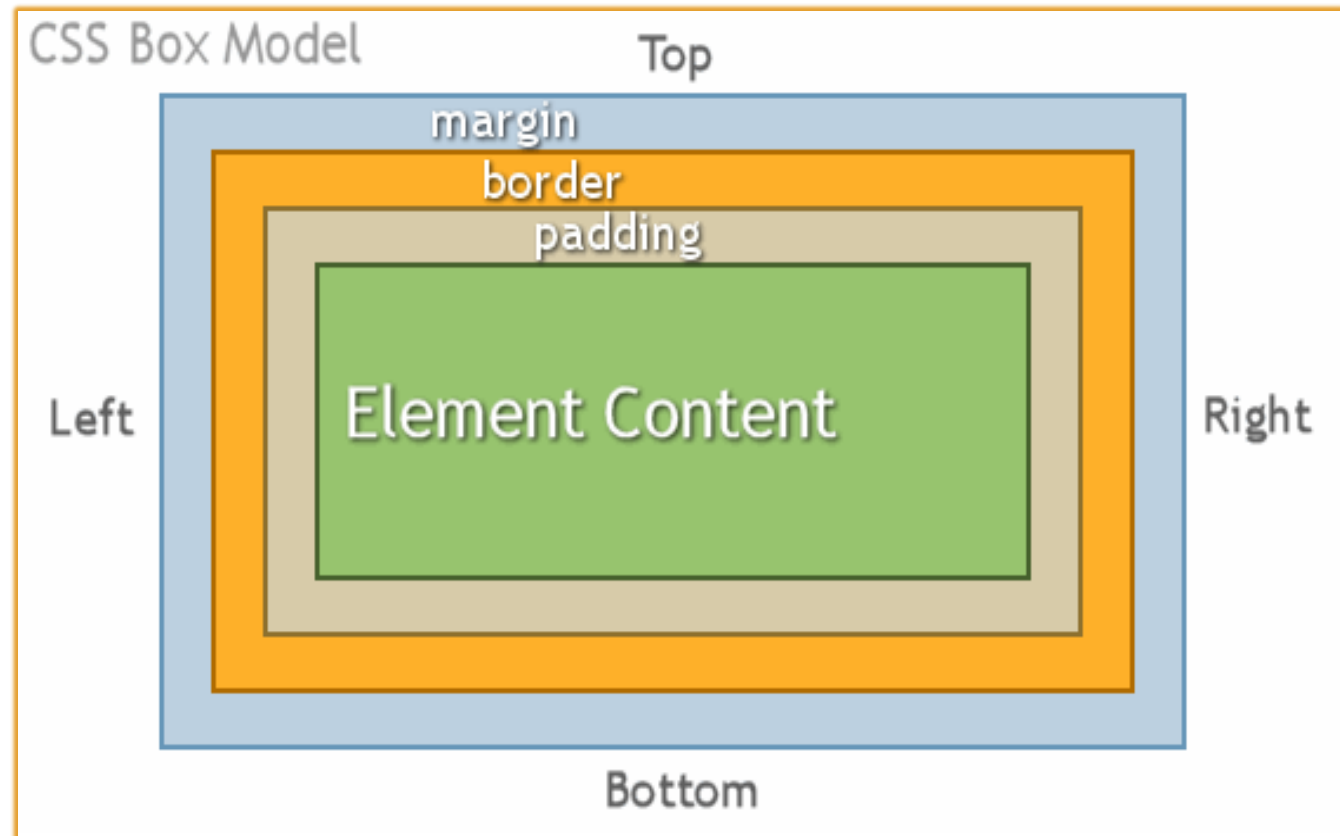
CSS - Box Model Example

```
<!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 ev
consists of: borders, padding, margins, and the actual conten

<div>This text is the content of the box. We have added a 50p
15px green border. Ut enim ad minim veniam, quis nostrud exerc
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolo
voluptate velit esse cillum dolore eu fugiat nulla pariatur. E
cupidatat non proident, sunt in culpa qui officia deserunt mol
</div>

</body>
</html>
```
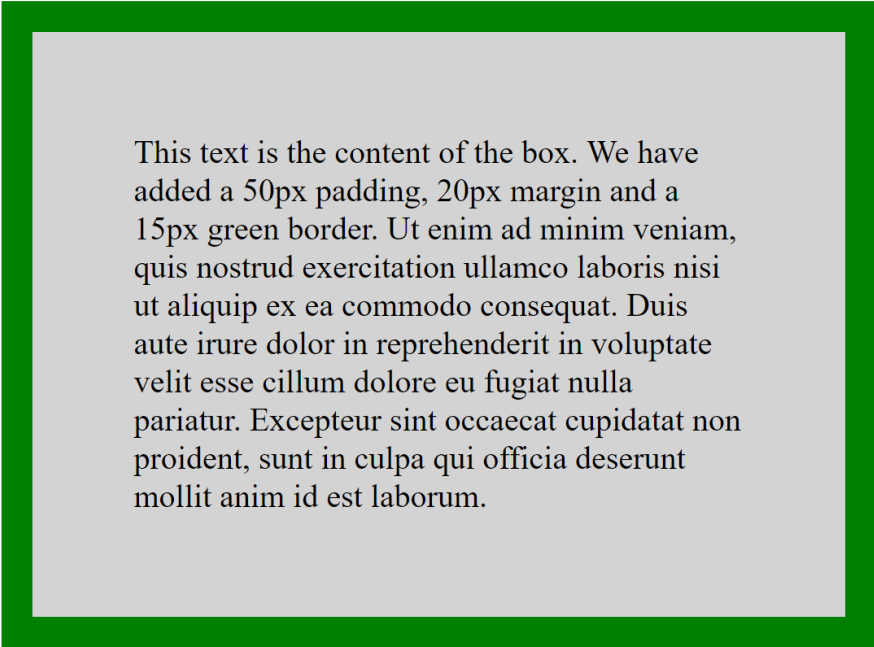
Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML elemen 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.
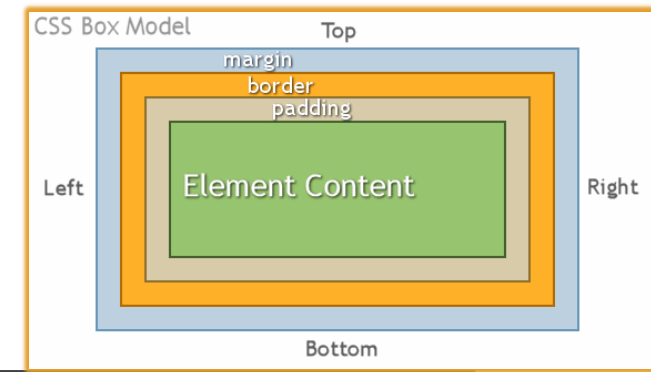
CSS Box Model – Example

CSS Box Model — Top, margin, border, padding, Element Content, Left, Right, Bottom
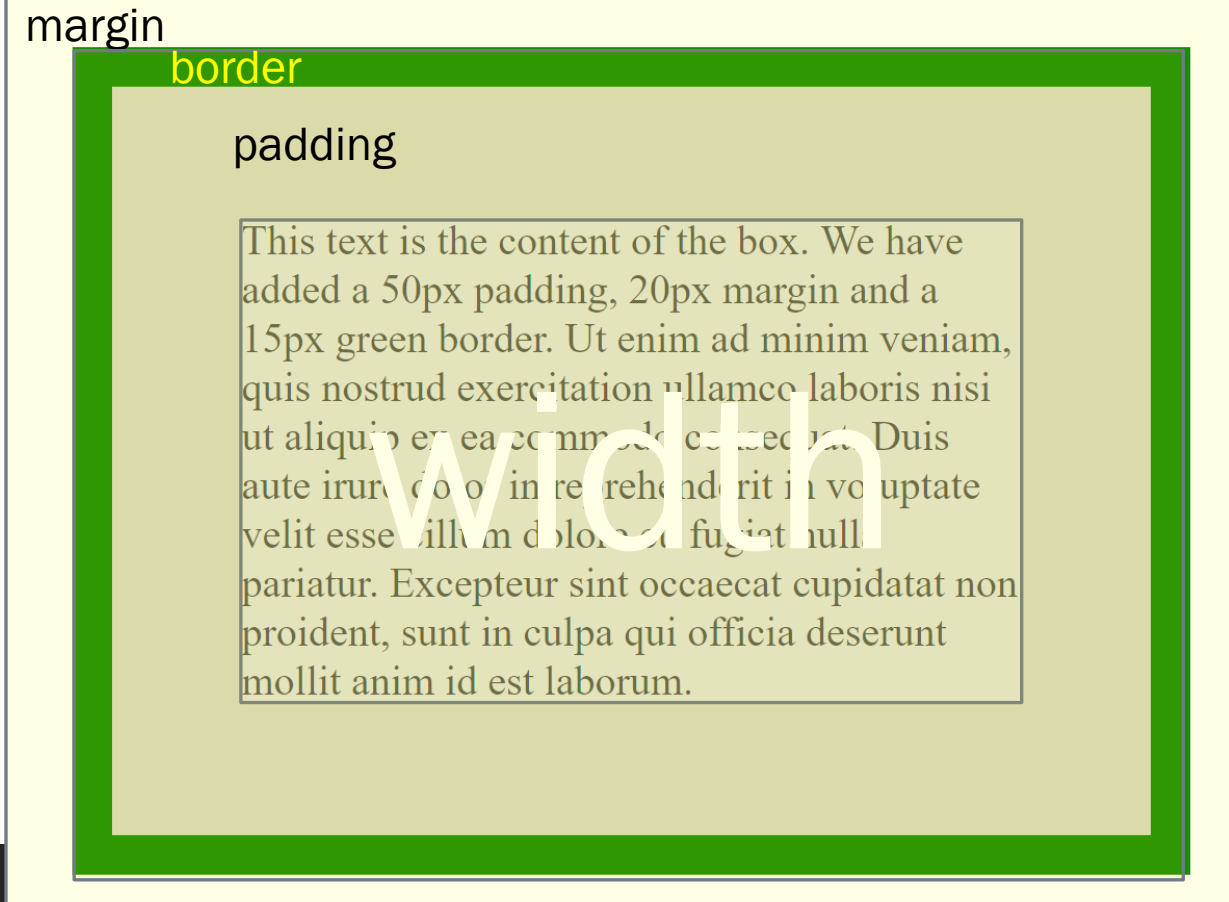
In order to correctly set the width and height of an element, you must know how the box model works.

```
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

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

There are two ways to give measurement in CSS. Both absolute and Relative have their uses.

Absolute	
px	pixels
mm	millimeters
cm	centimeters
In	inches
Pt	points(1/72 of an inch)
Pc	picas (12 points)

Relative	
%	percentage relative to parent element's width
em	1em is the same as the font size(scales with it) of the current element
rem	stands for "root em" and works the same except refers to the base font size
vh, vw	these are 1/100th of the height and width of the viewport, respectively

Most common

CSS – Colors

There are four ways color can be set in CSS.

Colors	
keywords	'red', 'blue', 'aquamarine', etc
hex value	# 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)	a function to specify values for red, green, and blue. Ex. rgb(224,176,255). Optional alpha (transparency) value.
hsl(h,s,l)	a function to specify hue, saturation, and lightness values to define a color. Ex. hsl(240,100%,50%). Optional alpha (transparency) value. This isn't used very much.

CSS - Three Ways To Include Styling

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

External CSS file
(Recommended) :
Include a reference to the relative
location of the .css file inside a <link>
element, inside 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 - Three Ways To Include Styling

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 - Three Ways To Include Styling

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 - Rule Set Conflicts

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

What happens when you have two CSS rules that 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

7

TMW Code Club

CSS – Importance (!important)

There is a special CSS flag you can use to ensure that a style always gets applied - the *!important* flag.

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 - Specificity

Specificity is a measure of how specific a selector is - basically, how many elements the selector could match. The selector with the <u>most</u> specificity will be applied over other selectors.

Each kind of selector is listed below in order of least to most specific:
- No selector / inline (declared inside style attribute)
- ID selector
- Class / attribute / pseudo-class selector
- Element or pseudo-element selector

The following CSS styles determine which *rule* will be applied.
- div { color: red; }   /* least specific - won't be applied */
- #yellow { color: yellow; } /* most specific - this style will be applied */
- .green { color: green; }   /* in the middle - won't be applied */

CSS – Source Order

If two selectors have the same 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. This style is applied */

If you mix an external stylesheet with internal styling, the internal styling will win.

https://www.w3schools.com/cssref/css_inherit.asp
https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance

```html
<!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 widths, margins, padding, and borders of elements do not automatically inherit styling from their parents.

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

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

CSS – display Property

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

Note: you can also use the *visibility: hidden* property to hide an element. The difference is that *display: none* removes the element from the page completely whereas **visibility: hidden** means that the tag is allocated 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

The *position* property specifies the type of positioning method used for an element.

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

Here are some position property values you should know:
- *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 example

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.

his heading is moved left according to its normal position

This heading is moved right according to its normal position

CSS - Responsive Web Design

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

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 on what 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'

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

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

CSS3

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, other still are 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