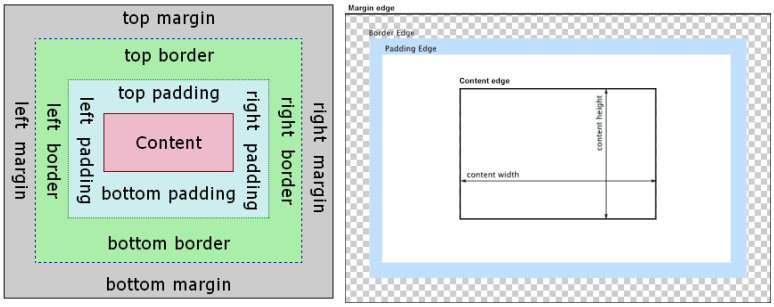
CSS Fundamentals

Box Model

- **Box Model:** the CSS box model lays out how the HTML elements are modeled in browser engines and how the dimensions of those HTML elements are derived from CSS properties. It is a fundamental concept for the composition of HTML webpages. The box model design and layout can be broken down into the following parts:



  - Content - the inner element of your HTML element. The content area, bounded by the content edge, is where your text, image, or video player would be displayed.
    - Background-color and background-image appear here as well.
    - The content's area size can be explicitly defined with the width, min-width, max-width, height, min-height, and max-height properties.
  - Padding - space around an element's content inside of any defined borders. The padding area is bounded by the padding edge.
    - Padding-box width and padding box height set the padding area.
    - The thickness is determined by padding-top, padding-right, padding-bottom, padding-left, and shorthand padding properties.
  - Border - goes around the padding and content. The border is bounded by the border edge.
    - Thickness of the border is set by border-box width and border-box height.
    - Area of the border is defined with width, min-width, max-width, height, min-height, and max height properties.
    - Border styles = dotted, dashed, solid, double, groove, ridge, inset, outset, none, and hidden. The border-style property can have from one to four values for the top border, right border, bottom border, bottom border, and the left border. Border-radius can make the border rounded.
  - Margin - clears an area outside the border. The margin area is bounded by the margin edge.
    - Size of margin area is determined by margin-top, margin-right, margin-bottom, margin-left, and shorthand margin properties.

```html
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lavender;
  width: 300px;
  border: 15px solid purple;
  padding: 100px;
  margin: 30px;
}
</style>
</head>
<body>

<h2>Demonstrating the Box Model</h2>

<div>Well here it is in all its glory.</div>

</body>
</html>
```

Demonstrating the Box Model

Well here it is in all its glory.

Responsive Web Design

- **Responsive Web Design:** a design layout using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it appear correctly on multiple devices and screen sizes.

  - Advantages:
    - Consistent visual experience, better user experience, no need for redirects, lower maintenance needs, high web loading speed, no extra fees for creating and maintaining different versions, and easy analytics reporting.
  - Disadvantages:
    - Not fully optimized, can slow down performance, may suffer from web browser incompatibility, makes it challenging to run advertising campaigns, and it can be difficult to offer different things to different users depending on the device used.
  - Key features when creating a responsive web design:
    - **Viewport** - user's visible area of a web page. The viewports will be bigger for laptops and tv's and smaller for tablets and phones. Browsers on smaller devices need to scale down the entire web page to fit the screen.

- RWD Implementation:
    - Use the <meta> tag in all of your webpages to control the viewport. The device-width sets the screen width of the page to follow the screen width of the device. The initial-scale=1.0 sets the initial zoom on the device when it is first loaded by the browser.

```
<meta name="viewport" content="width=device-width,
                               initial-scale=1.0">
```

- **Flexible layout** - a flexible grid built with relative units of measurement (percentage, em's) rather than absolute units such as pixels or points.
    - RWD Implementation:
        - Using the relative size formula (Target size/context = relative size) converts a design layout using pixels and converts it into percentages.

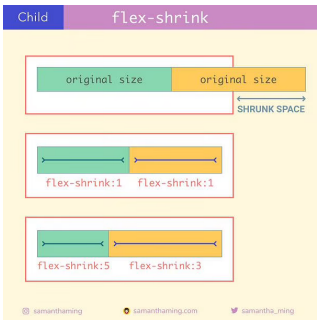
```
.col {
  width: 6.25%; /* 60 / 960 = 0.0625 */
}
```

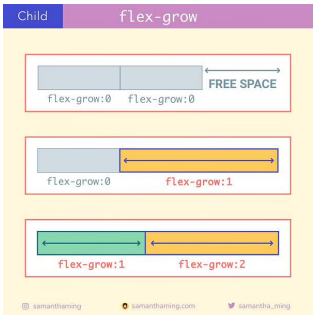
```
***(Target column = 60 pixels) / (context (or container) = 960 pixels) = 0.0625 → 6.25%
```

        - Use a multiple-column layout - how many columns do you want your column to be split into? Depending on what you set the browser will change according to screen size.

```
.container {
  column-count: 3;
}
```

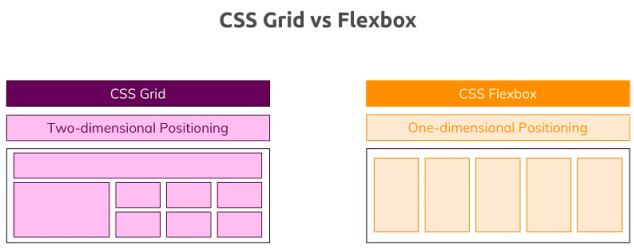
        - Use Flexbox (CSS Flexible Box Layout) - the flex layout allows responsive elements within a container to be automatically arranged depending on screen size. Values flex-grow and flex-shrink are used to modify the space around the items.





        - Use CSS Grid Layout - creates complex responsive web design layout using columns and rows. The spacing is

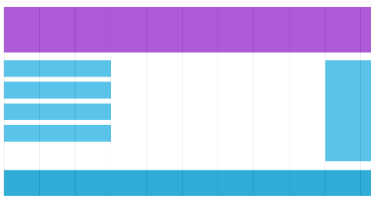
achieved by using grid - template - areas or grid - template - column.



- **Flexible images** - images should move and scale along with a flexible grid, container, or column. Multiple sizes of an image should be used to show the best resolution for all device sizes.
- **Flexible Videos** - To make a video player responsive you can set the width to scale the video up and down. You can also set the width to be max-width. Max-width will allow you to scale the video down, but never scale up to be larger.

```
video {
  width: 100%;
  height: auto;
}
```

- **Grid view** - when designing a webpage it is beneficial to use a 12 column style gridview. This makes it easier to place elements on a page.



- **Frameworks** - Bootstrap is an RWD framework composed of HTML, CSS and jQuery.
- **Media queries** - allow building different layouts within one project by tweaking your whole design or parts to best suit the screen size.
    - RWD Implementation:
        - The @media rule in CSS includes a block of CSS properties only if certain conditions are true. The example below shows the @media rule in effect. Notice there's a mobile device column width at 100%, a 600px tablet with a column width modified, and a 768px desktop with a column width modified. Whenever you are designing a webpage it is best to style the code to fit a small device and then scale it to a bigger device. Also, add breakpoints to your code to differentiate the anticipated size needed for a small, medium, large, or extra large device.

```
/* For mobile phones: */
[class*="col-"] {
  width: 100%;
}

@media only screen and
(min-width: 600px) {
  /* For tablets: */
  .col-s-1 {width: 8.33%;}
  .col-s-2 {width:
16.66%;}
  .col-s-3 {width: 25%;}
  .col-s-4 {width:
33.33%;}
  .col-s-5 {width:
41.66%;}
  .col-s-6 {width: 50%;}
  .col-s-7 {width:
58.33%;}
  .col-s-8 {width:
66.66%;}
  .col-s-9 {width: 75%;}
  .col-s-10 {width:
83.33%;}
  .col-s-11 {width:
91.66%;}
  .col-s-12 {width: 100%;}
}

@media only screen and
(min-width: 768px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}

  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
```

Specificity

Specificity is a weight given to a CSS declaration, and determines what styling will be applied. Specificity only applies when the same element is targeted by multiple declarations. The rule here is that the most specific style wins out.

- So what is the most specific way of making a CSS declaration?
  - Inline style is more specific than an external CSS file.
  - ID's
  - Classes and Attributes
  - Elements

Specificity Point System

Selector :	Example :	Points Worth :
Inline Style Attribute	 I am blue 	1000
ID's	#shepard <p ID="shepard">	100
Classes and Attributes	.terrier <p class="terrier">	10
Elements	h1, h2, p, div	1

CSS Rule Sets

A rule set is a selector with a declaration.
The declaration includes a property and a value.

Syntax is same for external stylesheet or between html <style> tags

h1 { color: blue; }

selector property value

Compound Selectors

h1, h2	selects all h1 and h2 elements		img:hover	selects any image that has a mouse over
a img	selects all images inside an anchor		div + img	selects all images that immediately follow a div
img.puppy	selects all images with class="puppy"		#cuddly	select all elements where class="cuddly"

Syntax for inside html element tags:

Regular HTML image tag:
<img src="example.jpg" height=300px>

HTML image tag with CSS Styling:

<img src="example.jpg" style="height:300px">

Class & ID Selectors

.class #ID

<img src="puppy.jpg" class="onsale" ID="bulldog">

.onsale { width : 100px }

#bulldog { width: 300px }