



UNIT3: STYLE SHEETS

Yuba Raj Devkota

CSS - INTRODUCTION

CSS stands for Cascading Style Sheets.

CSS saves a lot of work. It can control the layout of multiple web pages all at once.

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

The word **cascading** means that a style applied to a parent element will also apply to all children elements within the parent. So, if you set the color of the body text to "blue", all headings, paragraphs, and other text elements within the body will also get the same color (unless you specify something else)!

Using CSS

CSS can be added to HTML documents in 3 ways:

- **Inline** - by using the `style` attribute inside HTML elements
- **Internal** - by using a `<style>` element in the `<head>` section
- **External** - by using a `<link>` element to link to an external CSS file

The most common way to add CSS, is to keep the styles in external CSS files. However, in this tutorial we will use inline and internal styles, because this is easier to demonstrate, and easier for you to try it yourself.

Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the `style` attribute of an HTML element.

The following example sets the text color of the `<h1>` element to blue, and the text color of the `<p>` element to red:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1 style="color:blue;">A Blue Heading</h1>
```

```
<p style="color:red;">A red paragraph.</p>
```

```
</body>
```

```
</html>
```

A Blue Heading

A red paragraph.

Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the `<head>` section of an HTML page, within a `<style>` element.

The following example sets the text color of ALL the `<h1>` elements (on that page) to blue, and the text color of ALL the `<p>` elements to red. In addition, the page will be displayed with a "powderblue" background color:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

This is a heading

This is a paragraph.

External CSS

An external style sheet is used to define the style for many HTML pages.

To use an external style sheet, add a link to it in the `<head>` section of each HTML page:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

This is a heading

This is a paragraph.

"styles.css":

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

CSS COLORS, FONTS AND SIZES

Here, we will demonstrate some commonly used CSS properties. You will learn more about them later.

The CSS `color` property defines the text color to be used.

The CSS `font-family` property defines the font to be used.

The CSS `font-size` property defines the text size to be used.


```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
p {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

This is a heading

This is a paragraph.

CSS Border

The CSS `border` property defines a border around an HTML element.

Tip: You can define a border for nearly all HTML elements.

Example

Use of CSS border property:

```
p {  
  border: 2px solid powderblue;  
}
```

CSS Padding

The CSS `padding` property defines a padding (space) between the text and the border.

Example

Use of CSS border and padding properties:

```
p {  
  border: 2px solid powderblue;  
  padding: 30px;  
}
```

CSS Margin

The CSS `margin` property defines a margin (space) outside the border.

Example

Use of CSS border and margin properties:

```
p {  
  border: 2px solid powderblue;  
  margin: 50px;  
}
```

CSS *.CLASS* SELECTOR

```
<!DOCTYPE html>
<html>
<head>
<style>
p.hometown {
  background: yellow;
}
</style>
</head>
<body>

<h1>Demo of the .class selector</h1>

<p>My name is Donald.</p>
<p class="hometown">I live in Ducksburg.</p>

<p>My name is Dolly.</p>
<p class="hometown">I also live in Ducksburg.</p>

</body>
</html>
```

Select and style all elements with class="intro":

```
.intro {
  background-color: yellow;
}
```

Demo of the .class selector

My name is Donald.

I live in Ducksburg.

My name is Dolly.

I also live in Ducksburg.

LAB WORKS — NO 14

Consider a HTML page contains two divisions and one paragraph tags. The divisions have id div1 and div2 respectively. The color of text in both divisions should be red and background color of the divisions should be green. The font style in paragraph should be Arial and the size of font should be 14. Write necessary CSS for the given scenario.

CSS EXAMPLES

This is heading 1

This is an ordinary paragraph. Notice that this text is blue. The default text color for a page is defined in the body selector.

Another paragraph.

Heading 1 (center)

Heading 2 (left)

Heading 3 (right)

The three headings above are aligned center, left and right.

```
<style>
body {
  color: blue;
}

h1 {
  color: green;
}
</style>
```

```
<style>
h1 {
  text-align: center;
}

h2 {
  text-align: left;
}

h3 {
  text-align: right;
}
</style>
```

Using text-decoration: none

A link with no underline: [W3Schools.com](https://www.w3schools.com)

```
<style>
a {
  text-decoration: none;
}
```

Heading 1

Heading 2

Heading 3

A paragraph.



```
<style>
h1 {
  text-decoration: underline;
}

h2 {
  text-decoration: underline red;
}

h3 {
  text-decoration: underline red double;
}

p {
  text-decoration: underline red double 5px;
}
```

Using letter-spacing

T h i s i s h e a d i n g 1

This is heading2

```
<style>
h2 {
  letter-spacing: 5px;
}

h3 {
  letter-spacing: -2px;
}
```

Using line-height

This is a paragraph with a standard line-height.
The default line height in most browsers is about 110% to 120%.

This is a paragraph with a smaller line-height.
This is a paragraph with a smaller line-height.

This is a paragraph with a bigger line-height.

This is a paragraph with a bigger line-height.

```
<style>
p.small {
  line-height: 0.7;
}

p.big {
  line-height: 1.8;
}
</style>
```


CSS BOX MODEL

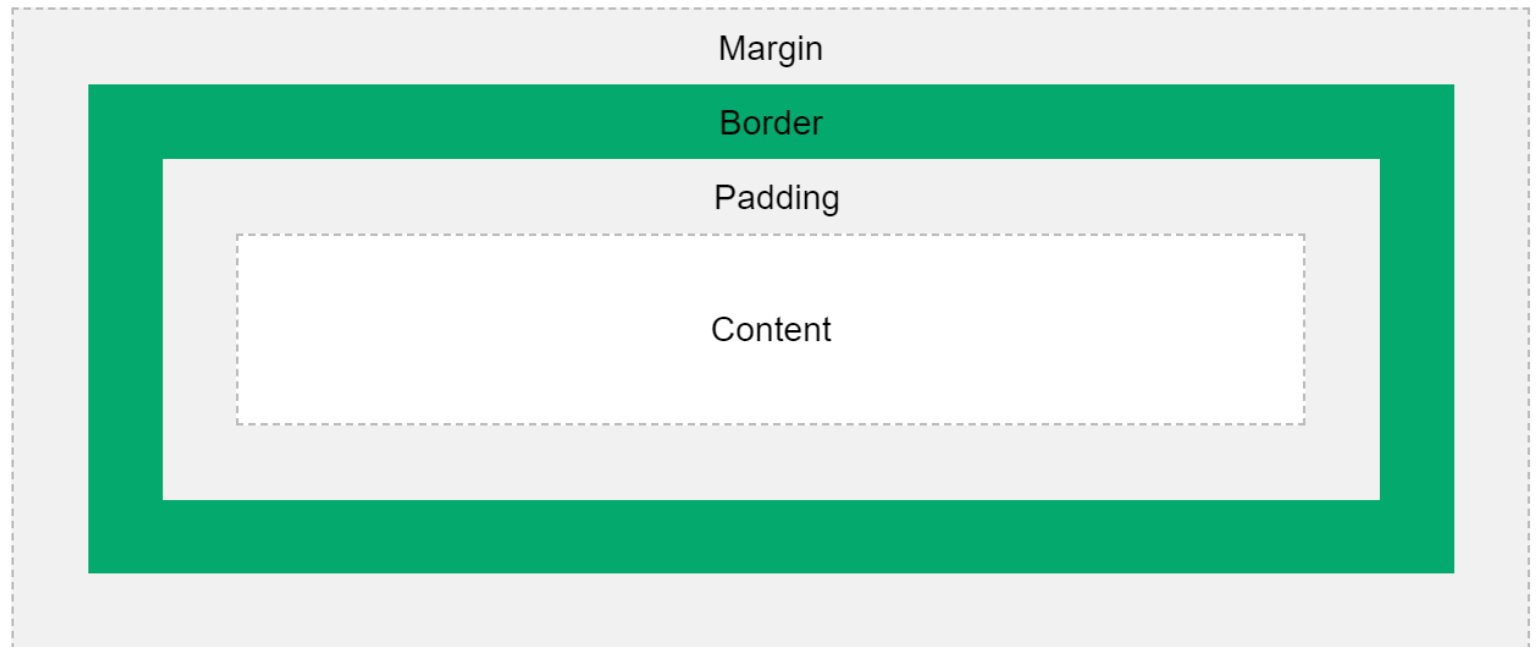
In CSS, the term "box model" is used when talking about design and layout.

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

The box model allows us to add a border around elements, and to define space between elements.

The image below illustrates the box model:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent



```
<!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>
```

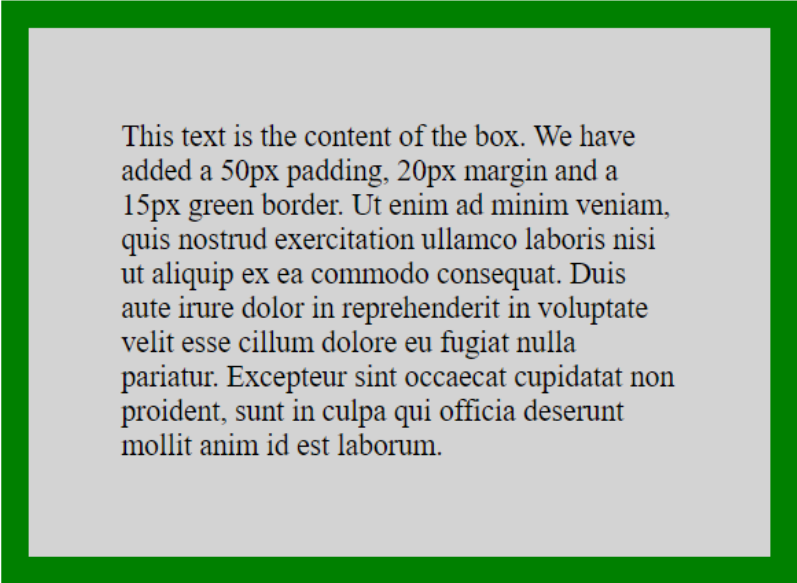
```
<p>The CSS box model is essentially a box that wraps around
every HTML element. It consists of: borders, padding, margins,
and the actual content.</p>
```

```
<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.

Create a box model that wraps around <div> tag with margins, borders, padding and actual content

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    /* Box Model Styling */
    .box-container {
      width: 300px; /* Set the width of the content area */
      margin: 20px; /* Set the margin outside the border */
      padding: 15px; /* Set the padding inside the border */
      border: 2px solid #333; /* Set the border around the padding */
      box-sizing: border-box; /* Include padding and border in the total width */
    }
  </style>
</html>
```

```
/* Additional Styling for Visualization */
```

```
body {  
  font-family: Arial, sans-serif;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  height: 100vh;  
  margin: 0;  
}
```

```
.content {  
  background-color: #f0f0f0;  
  text-align: center;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
  <!-- Box Model Container -->
```

```
  <div class="box-container">
```

```
    <!-- Content Area -->
```

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

```
      <p>This is the actual content.</p>
```

```
    </div>
```

```
  </div>
```

```
</body>
```

```
</html>
```

CSS LAYOUT - THE POSITION PROPERTY

The `position` property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

```
position: static|absolute|fixed|relative|sticky|initial|inherit;
```

Value	Description
static	Default value. Elements render in order, as they appear in the document flow
absolute	The element is positioned relative to its first positioned (not static) ancestor element
fixed	The element is positioned relative to the browser window
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position
sticky	<p>The element is positioned based on the user's scroll position</p> <p>A sticky element toggles between <code>relative</code> and <code>fixed</code>, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like <code>position:fixed</code>).</p>

position: static;

HTML elements are positioned static by default. Static positioned elements are not affected by the top, bottom, left, and right properties.

```
div.static {  
    position: static;  
    border: 3px solid #73AD21;  
}
```

This <div> element has position: static;

position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

```
<style>  
div.relative {  
    position: relative;  
    left: 30px;  
    border: 3px solid #73AD21;  
}  
</style>
```

position: relative;

An element with position: relative; is positioned relative to its normal position:

This div element has position: relative;

position: fixed;

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

```
<style>
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```

position: fixed;

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:

position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

This <div> element has position: relative;

This <div> element has
position: absolute;

```
div.relative {  
  position: relative;  
  width: 400px;  
  height: 200px;  
  border: 3px solid #73AD21;  
}
```

```
div.absolute {  
  position: absolute;  
  top: 80px;  
  right: 0;  
  width: 200px;  
  height: 100px;  
  border: 3px solid #73AD21;  
}
```


position: sticky;

An element with `position: sticky;` is positioned based on the user's scroll position.

A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like `position: fixed`).

Try to scroll inside this frame to understand how sticky positioning works.

I am sticky!

Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

```
div.sticky {  
  position: -webkit-sticky; /* Safari */  
  position: sticky;  
  top: 0;  
  background-color: green;  
  border: 2px solid #4CAF50;  
}
```

CSS SELECTORS

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

1. Simple selectors (select elements based on name, id, class)
2. Combinator selectors (select elements based on a specific relationship between them)
3. Pseudo-class selectors (select elements based on a certain state)
4. Pseudo-elements selectors (select and style a part of an element)
5. Attribute selectors (select elements based on an attribute or attribute value)

The CSS element Selector

Here, all <p> elements on the page will be center-aligned, with a red text color:

The CSS rule below will be applied to the HTML element with id="para1":

An id name cannot start with a number!

In this example all HTML elements with class="center" will be red and center-aligned:

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

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

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

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
p.center {
```

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

```
  color: red;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1 class="center">This heading will not be affected</h1>
```

```
<p class="center">This paragraph will be red and center-aligned.</p>
```

```
</body>
```

```
</html>
```

This heading will not be affected

This paragraph will be red and center-aligned.

CSS Combinators

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator. There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Descendant Selector

The following example selects all <p> elements inside <div> elements:

```
div p {  
    background-color: yellow;  
}
```

Child Selector (>)

The child selector selects all elements that are the children of a specified element.

The following example selects all <p> elements that are children of a <div> element:

```
div > p {  
    background-color: yellow;  
}
```

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
div > p {  
    background-color: yellow;  
}  
</style>  
</head>  
<body>  
  
<h2>Child Selector</h2>  
  
<p>The child selector (>) selects all elements that are the children of a specif  
  
<div>  
    <p>Paragraph 1 in the div.</p>  
    <p>Paragraph 2 in the div.</p>  
    <section>  
        <!-- not Child but Descendant -->  
        <p>Paragraph 3 in the div (inside a section element).</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>  
</html>
```

Child Selector

The child selector (>) selects all elements that are the children of a specified element.

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div (inside a section element).

Paragraph 4 in the div.

Paragraph 5. Not in a div.

Paragraph 6. Not in a div.

Adjacent Sibling Selector (+)

The adjacent sibling selector is used to select an element that is directly after another specific element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects the first <p> element that are placed immediately after <div> elements:

```
div + p {  
    background-color: yellow;  
}
```

General Sibling Selector (~)

The general sibling selector selects all elements that are next siblings of a specified element.

The following example selects all <p> elements that are next siblings of <div> elements:

```
div ~ p {  
    background-color: yellow;  
}
```

CSS Pseudo-classes

A pseudo-class is used to define a special state of an element. For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Anchor Pseudo-classes: Links can be displayed in different ways:

```
/* unvisited link */
a:link {
    color: #FF0000;
}

/* visited link */
a:visited {
    color: #00FF00;
}
```

```
/* mouse over link */
a:hover {
    color: #FF00FF;
}

/* selected link */
a:active {
    color: #0000FF;
}
```

Note: `a:hover` MUST come after `a:link` and `a:visited` in the CSS definition in order to be effective! `a:active` MUST come after `a:hover` in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.

Match all `<i>` elements in all first child `<p>` elements

In the following example, the selector matches all `<i>` elements in `<p>` elements that are the first child of another element:

```
<!DOCTYPE html>
<html>
<head>
<style>
p:first-child i {
  color: blue;
}
</style>
</head>
<body>

<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>

<div>
  <p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
  <p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
</div>

</body>
</html>
```

I am a *strong* person. I am a *strong* person.

I am a *strong* person. I am a *strong* person.

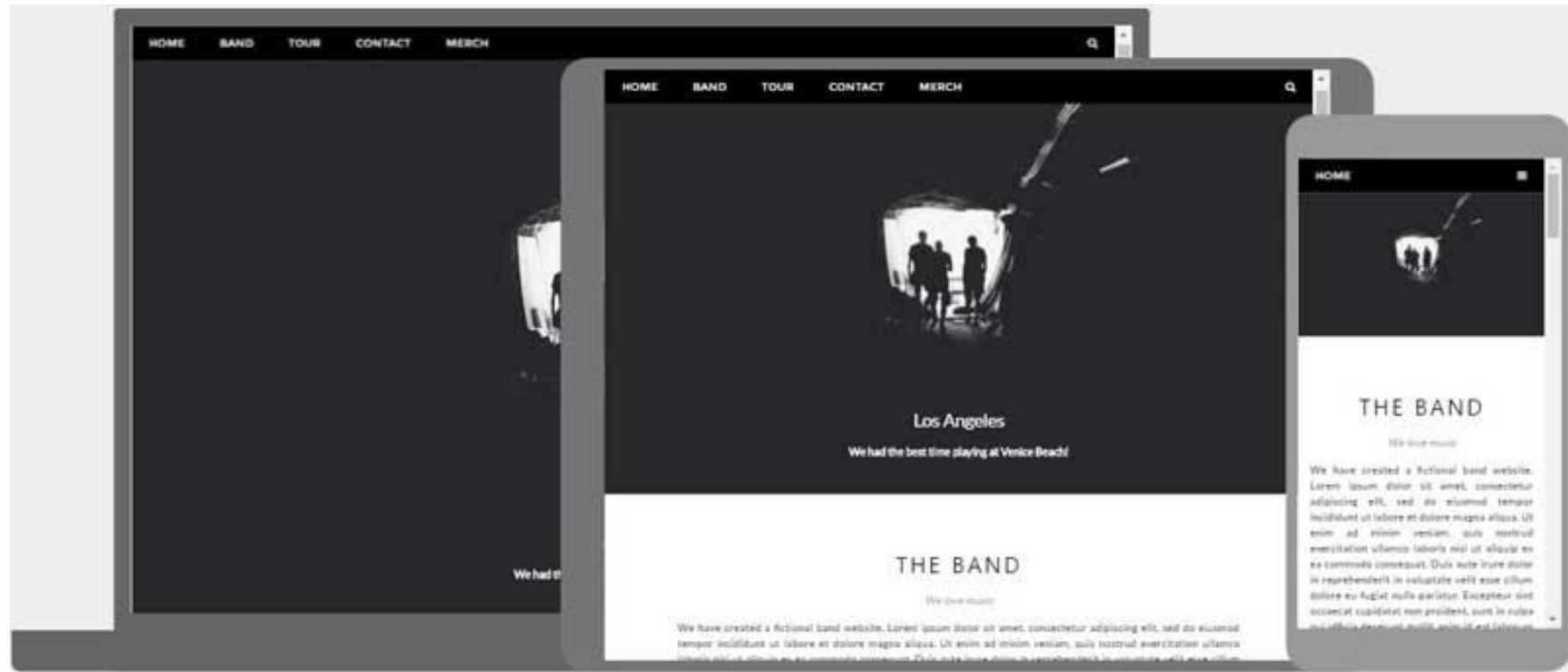
I am a *strong* person. I am a *strong* person.

I am a *strong* person. I am a *strong* person.

RESPONSIVE WEB DESIGN

Responsive web design is about creating web pages that look good on all devices!
A responsive web design will automatically adjust for different screen sizes and viewports.

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones):



SETTING THE VIEWPORT

To create a responsive website, add the following `<meta>` tag to all your web pages:

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

- This will set the viewport of your page, which will give the browser instructions on how to control the page's dimensions and scaling.
- Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:



Lorem ipsum dolor sit amet, consectetur 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 feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet domine...

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
<body>

<h2>Setting the Viewport</h2>
<p>This example does not really do anything, other than showing
you how to add the viewport meta element.</p>

</body>
</html>
```

Setting the Viewport

This example does not really do anything, other than showing you how to add the viewport meta element.

RESPONSIVE IMAGES

Responsive images are images that scale nicely to fit any browser size.

Using the width Property: If the CSS width property is set to 100%, the image will be responsive and scale up and down:

```

```

Notice that in the example above, the image can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the max-width property instead.

Using max-width property: if the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

```

```

RESPONSIVE TEXT SIZE

The text size can be set with a "vw" unit, which means the "viewport width".
That way the text size will follow the size of the browser window:

Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.

Hello World

Resize the browser window to see how the text size scales.

```
<h1 style="font-size:10vw">Hello World</h1>
```

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
<body>
```

```
<h1 style="font-size:10vw;">Responsive Text</h1>
```

```
<p style="font-size:5vw;">Resize the browser window to see
how the text size scales.</p>
```

```
<p style="font-size:5vw;">Use the "vw" unit when sizing the text.
10vw will set the size to 10% of the viewport width.</p>
```

```
<p>Viewport is the browser window size. 1vw = 1% of
viewport width. If the viewport is 50cm wide, 1vw is
0.5cm.</p>
```

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

Responsive Text

Resize the browser window to see how the text size scales.

Use the "vw" unit when sizing the text. 10vw will set the size to 10% of the viewport width.

Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.

MEDIA QUERIES

In addition to resize text and images, it is also common to use media queries in responsive web pages.

With media queries you can define completely different styles for different browser sizes.

Example: resize the browser window to see that the three div elements below will display horizontally on large screens and stack vertically on small screens:

Left Menu

Main Content

Right Content


```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport"
content="width=device-width, initial-
scale=1.0">
<style>
* {
  box-sizing: border-box;
}

.left {
  background-color: #2196F3;
  padding: 20px;
  float: left;
  width: 20%; /* The width is 20%, by default */
}

.main {
  background-color: #f1f1f1;
  padding: 20px;
  float: left;
  width: 60%; /* The width is 60%, by default */
}
```

```
.right {
  background-color: #04AA6D;
  padding: 20px;
  float: left;
  width: 20%; /* The width is
20%, by default */
}

/* Use a media query to add a
break point at 800px: */
@media screen and (max-width:
800px) {
  .left, .main, .right {
    width: 100%; /* The width is
100%, when the viewport is
800px or smaller */
  }
}
</style>
</head>
```

```
<body>

<h2>Media Queries</h2>
<p>Resize the browser window.</p>

<p>Make sure you reach the breakpoint
at 800px when resizing this frame.</p>

<div class="left">
  <p>Left Menu</p>
</div>

<div class="main">
  <p>Main Content</p>
</div>

<div class="right">
  <p>Right Content</p>
</div>

</body>
</html>
```

Media Queries

Resize the browser window.

Make sure you reach the breakpoint at 800px when resizing this frame.

Left Menu

Main Content

Right Content

BOOTSTRAP

Another popular CSS framework is Bootstrap. Bootstrap uses HTML, CSS and jQuery to make responsive web pages.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
```

```
<body>

<div class="container">
  <div class="jumbotron">
    <h1>My First Bootstrap Page</h1>
    <p>Resize this responsive page to see the effect!</p>
  </div>
  <div class="row">
    <div class="col-sm-4">
      <h3>Column 1</h3>
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit...</p>
      <p>Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris...</p>
    </div>
    <div class="col-sm-4">
      <h3>Column 2</h3>
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit...</p>
      <p>Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris...</p>
    </div>
  </div>
</div>
```

```
<div class="col-sm-4">
  <h3>Column 2</h3>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit...</p>
  <p>Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris...</p>
</div>
<div class="col-sm-4">
  <h3>Column 3</h3>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit...</p>
  <p>Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris...</p>
</div>
</div>

</body>
</html>
```

My First Bootstrap Page

Resize this responsive page to see the effect!

Column 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit...

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...

Column 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit...

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...

Column 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit...

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...

LAB EXERCISES

15. Give an Example of Responsive web design. Include text and images.
16. Give an example of External CSS.

5. Why is the use of CSS?
6. Define responsive web design.

13. How do you insert external style sheet in an HTML page?

18. What is CSS box model? Create a box model that wraps around `<div>` tag with margins, borders, padding, and actual content.

21. Explain different ways of inserting CSS in an HTML document? What is pseudo-class selector?