CRAFT knowledge

CSS Fundamental

What is CSS

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- CSS saves a lot of work. It can control the layout of multiple web pages all at once.
- External stylesheets are stored in CSS files.

CSS Syntax

A CSS rule consists of a selector and a declaration block.

- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example

In this example all <p> elements will be center-aligned, with a red text color:

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

- p is a selector in CSS (it points to the HTML element you want to style: <p>).
- color is a property, and red is the property value.
- text-align is a property, and center is the property value.

CSS Selectors

A CSS selector selects the HTML element(s) you want to style.

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

1. The CSS element Selector

- The element selector selects HTML elements based on the element name.

Example

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

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

2. The CSS id Selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example

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

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

Note: An id name cannot start with a number!

3. The CSS class Selector

- The class selector selects HTML elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the class name.

Example

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

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

· You can also specify that only specific HTML elements should be affected by a class.

Example

In this example only <p> elements with class="center" will be red and center-aligned:

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

```
  color: red;
}
```

Note: A class name cannot start with a number!

4. The CSS Universal Selector

- The universal selector (*) selects all HTML elements on the page.

Example

The CSS rule below will affect every HTML element on the page:

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

5. The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

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

h2 {
  text-align: center;
```

```
  color: red;
}

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

· It will be better to group the selectors, to minimize the code.

· To group selectors, separate each selector with a comma.

Example

In this example we have grouped the selectors from the code above:

All CSS Simple Selectors

How To Add CSS

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

· External CSS
· Internal CSS
· Inline CSS

1. External CSS

- With an external style sheet, you can change the look of an entire website by changing just one file!

- Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Example

External styles are defined within the <link> element, inside the <head> section of an HTML page:

```
<!DOCTYPE html>

<html>



<head>

  <link rel="stylesheet" href="mystyle.css">

</head>



<body>



  <h1>This is a heading</h1>

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



</body>



</html>
```

- An external style sheet can be written in any text editor, and must be saved with a .css extension.
- The external .css file should not contain any HTML tags.
- Here is how the "mystyle.css" file looks:

"mystyle.css"

```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

Note: Do not add a space between the property value (20) and the unit (px):
Incorrect (space): margin-left: 20 px;
Correct (no space): margin-left: 20px;

2. Internal CSS

- An internal style sheet may be used if one single HTML page has a unique style.
- The internal style is defined inside the <style> element, inside the head section.

Example

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

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

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

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

<body>

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

</body>

</html>
```

3. Inline CSS

- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Example

Inline styles are defined within the "style" attribute of the relevant element:

```
<!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>
```

Cascading Order

What style will be used when there is more than one style specified for an HTML element?

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)

2. External and internal style sheets (in the head section)
3. Browser default

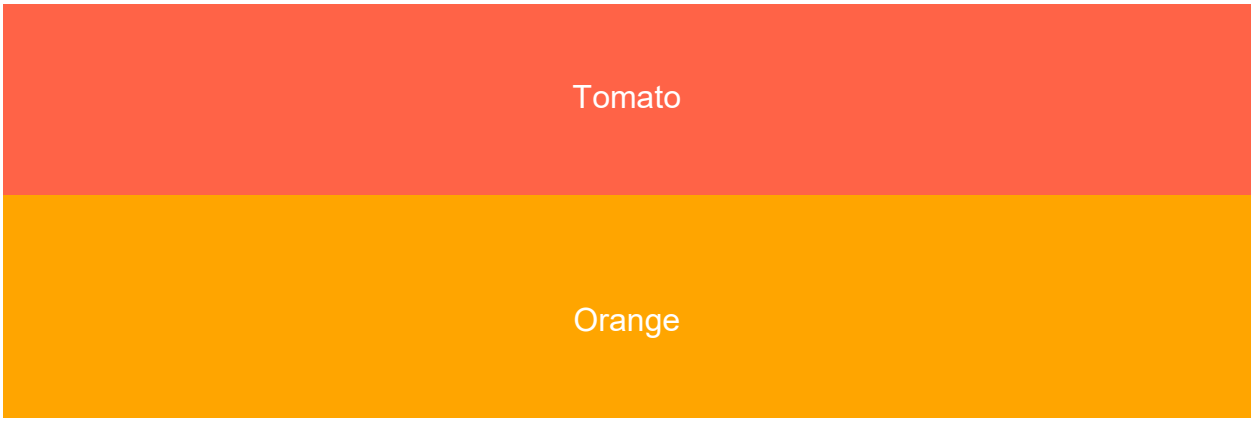
So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

CSS Colors

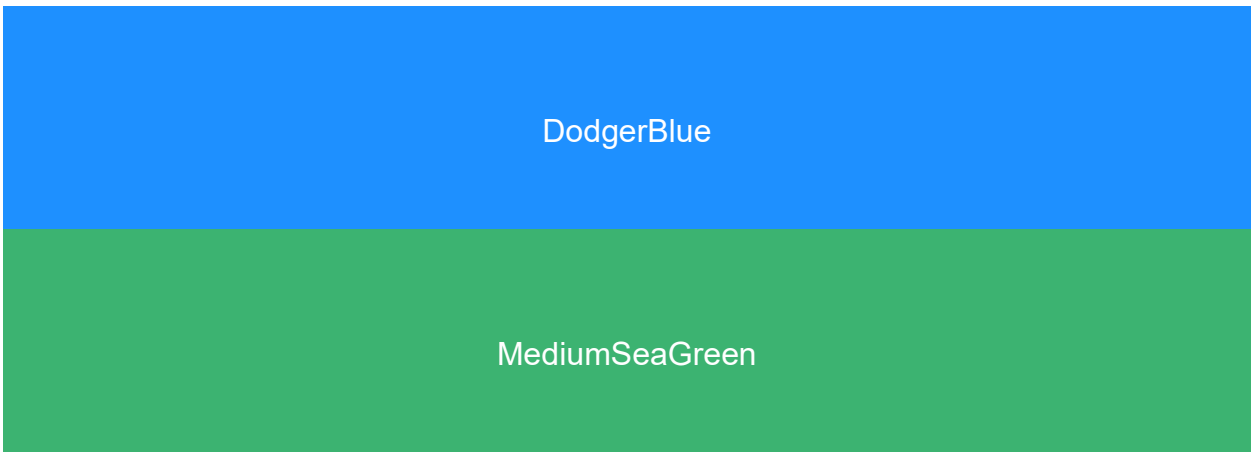
Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

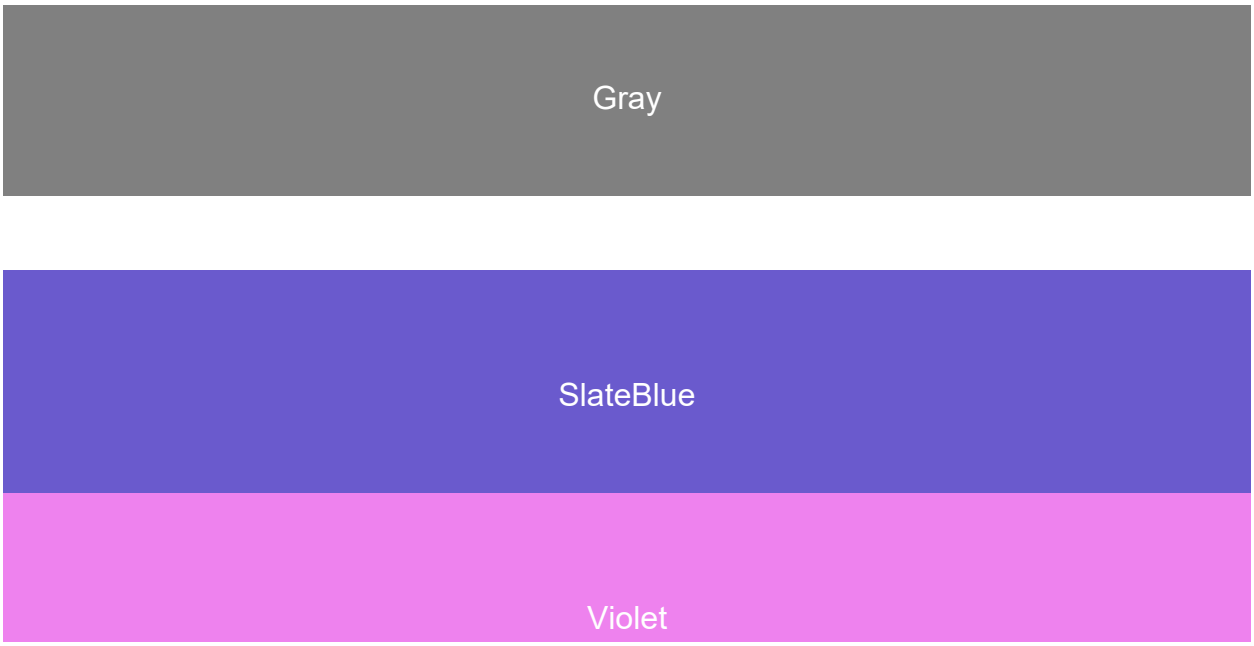
1. CSS Color Names

In CSS, a color can be specified by using a predefined color name:



LightGray





2. CSS Background Color

You can set the background color for HTML elements:

Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>

<p style="background-color:Tomato;">Lorem ipsum...</p>
```

3. CSS Text Color

You can set the color of text:

Hello World

Hello World

Hello World

Example

```
<h1 style="color:Tomato;">Hello World</h1>
```

```
<p style="color:DodgerBlue;">Lorem ipsum...</p>

<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

4. CSS Border Color

You can set the color of borders:

Hello World

Hello World

Hello World

Example

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

2. CSS RGB Colors

An RGB color value represents RED, GREEN, and BLUE light sources.

RGB Value

In CSS, a color can be specified as an RGB value, using this formula:

rgb(*red, green, blue*)

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

To display black, set all color parameters to 0, like this: rgb(0, 0, 0).

To display white, set all color parameters to 255, like this: rgb(255, 255, 255).

Experiment by mixing the RGB values below:

CSS background

The CSS background properties are used to add background effects for elements.

- background-color
- background-image

- background-repeat
- background-attachment
- background-position
- background (shorthand property)

CSS background-color

The background-color property specifies the background color of an element.

Example

The background color of a page is set like this:

```
body {
  background-color: lightblue;
}
```

Other Elements

You can set the background color for any HTML elements:

Example

Here, the <h1>, <p>, and elements will have different background colors:

```
h1 {
  background-color: green;
}

div {
  background-color: lightblue;
}
```

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

Opacity / Transparency

The opacity property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent:

Example

```
div {
  background-color: green;
  opacity: 0.3;
}
```

CSS background-image

The background-image property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

Example

Set the background image for a page:

```
body {
  background-image: url("paper.gif");
```

```
}
```

CSS background-repeat

By default, the background-image property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Example

```
body {
  background-image: url("gradient_bg.png");
}
```

If the image above is repeated only horizontally (background-repeat: repeat-x;), the background will look better:

Example

```
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
```

Tip: To repeat an image vertically, set background-repeat: repeat-y;

CSS background-repeat: no-repeat

Showing the background image only once is also specified by the background-repeat property:

Example

Show the background image only once:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
}
```

CSS background-position

The background-position property is used to specify the position of the background image.

Example

Position the background image in the top-right corner:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

CSS Background Attachment

The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):

Example

Specify that the background image should be **fixed**:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

Specify that the background image should **scroll** with the rest of the page:

```
body {
  background-color: #ffffff;
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: scroll;
}
```

CSS Borders

The CSS border properties allow you to specify the style, width, and color of an element's border.

The border-style property specifies what kind of border to display.

The following values are allowed:

dotted - Defines a dotted border

dashed - Defines a dashed border

solid - Defines a solid border

```
p.dotted {
  border-style: dotted;
}

p.dashed {
  border-style: dashed;
}

p.solid {
  border-style: solid;
}
```

CSS Border Width

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick:

Example

Demonstration of the different border widths:

```
p.one {
  border-style: solid;
  border-width: 5px;
}
```

```
p.two {
    border-style: solid;
    border-width: medium;
}

p.three {
    border-style: dotted;
    border-width: 2px;
}

p.four {
    border-style: dotted;
    border-width: thick;
}
```

CSS Border Color

The border-color property is used to set the color of the four borders.

The color can be set by:

- name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"

Note: If border-color is not set, it inherits the color of the element.

```
p.one {
    border-style: solid;
    border-color: red;
```

```
}

p.two {
    border-style: solid;
    border-color: green;
}

p.three {
    border-style: dotted;
    border-color: blue;
}
```

CSS Margins

Margins are used to create space around elements, outside of any defined borders.

The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

- margin-top
- margin-right

- margin-bottom
- margin-left

All the margin properties can have the following values:

- auto - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- *%* - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

- **Tip:** Negative values are allowed.

Example

Set different margins for all four sides of a <p> element:

```
p {
  margin-top: 100px;
        margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

- **Tip:** Negative values are allowed.

If the margin property has three values:

margin: 25px 50px 75px;

- top margin is 25px
- right and left margins are 50px
- bottom margin is 75px

Example

Use the margin shorthand property with three values:

```
p {
  margin: 25px 50px 75px;
}
```

If the margin property has two values:

margin: 25px 50px;

- top and bottom margins are 25px
- right and left margins are 50px

Example

Use the margin shorthand property with two values:

```
p {
  margin: 25px 50px;
}
```

If the margin property has one value:

margin: 25px;

- all four margins are 25px

Example

Use the margin shorthand property with one value:

```
p {
  margin: 25px;
}
```

The auto Value

You can set the margin property to auto to horizontally center the element within its container.

The element will then take up the specified width, and the remaining space will be split equally between the left and right margins.

Example

Use margin: auto:

```
div {
  width: 300px;
  margin: auto;
  border: 1px solid red;
}
```

The inherit Value

This example lets the left margin of the <p class="ex1"> element be inherited from the parent element ( ):

Example

Use of the inherit value:

```
div {
  border: 1px solid red;
  margin-left: 100px;
}

p.ex1 {
  margin-left: inherit;
```

```
}
```

CSS Padding

Padding is used to create space around an element's content, inside of any defined borders.

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- padding-left

All the padding properties can have the following values:

- *length* - specifies a padding in px, pt, cm, etc.
- *%* - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

Note: Negative values are not allowed.

1. Set different padding for all four sides of a element:

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

Padding - Shorthand Property

To shorten the code, it is possible to specify all the padding properties in one property.

The padding property is a shorthand property for the following individual padding properties:

- padding-top
- padding-right
- padding-bottom
- padding-left

So, here is how it works:

If the padding property has four values:

padding: 25px 50px 75px 100px;

- top padding is 25px
- right padding is 50px
- bottom padding is 75px
- left padding is 100px

Example

Use the padding shorthand property with four values:

```
div {
  padding: 25px 50px 75px 100px;
}
```

If the padding property has three values:

padding: 25px 50px 75px;

- top padding is 25px
- right and left paddings are 50px
- bottom padding is 75px

Example

Use the padding shorthand property with three values:

```
div {
  padding: 25px 50px 75px;
}
```

If the padding property has two values:

padding: 25px 50px;

- top and bottom paddings are 25px
- right and left paddings are 50px

Example

Use the padding shorthand property with two values:

```
div {
  padding: 25px 50px;
}
```

If the padding property has one value:

padding: 25px;

- all four paddings are 25px

Example

Use the padding shorthand property with one value:

```
div {
  padding: 25px;
```

```
}
```

CSS Height, Width and Max-width

The CSS height and width properties are used to set the height and width of an element.

The CSS max-width property is used to set the maximum width of an element.

CSS Setting height and width

The height and width properties are used to set the height and width of an element.

The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

CSS height and width Values

The height and width properties may have the following values:

- auto - This is default. The browser calculates the height and width
- length - Defines the height/width in px, cm, etc.
- % - Defines the height/width in percent of the containing block
- initial - Sets the height/width to its default value
- inherit - The height/width will be inherited from its parent value

CSS height and width Examples

Example

Set the height and width of a element:

```
div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
```

```
}
```

Note: Remember that the height and width properties do not include padding, borders, or margins! They set the height/width of the area inside the padding, border, and margin of the element!

Setting max-width

The max-width property is used to set the maximum width of an element.

The max-width can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

The problem with the above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

Using max-width instead, in this situation, will improve the browser's handling of small windows.

Tip: Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

This element has a height of 100 pixels and a max-width of 500 pixels.

Note: If you for some reason use both the width property and the max-width property on the same element, and the value of the width property is larger than the max-width property; the max-width property will be used (and the width property will be ignored).

Example

This element has a height of 100 pixels and a max-width of 500 pixels:

```
div {
  max-width: 500px;
  height: 100px;
  background-color: powderblue;
}
```

CSS Text

CSS has a lot of properties for formatting text.

Text Color

The color property is used to set the color of the text. The color is specified by:

- a color name - like "red"

```
h1 {
  color: green;
}
```

Text Color and Background Color

In this example, we define both the background-color property and the color property:

```
h1 {
  background-color: black;
  color: white;
}
```

Text Alignment

The text-align property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

```
h1 {
```

```
  text-align: center;
}

h2 {
  text-align: left;
}

h3 {
  text-align: right;
}
```

When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

Example

```
div {
  text-align: justify;
}
```

Text Align Last

The text-align-last property specifies how to align the last line of a text.

Example

Align the last line of text in three <p> elements:

```
p.a {
  text-align-last: right;
```

```
}


p.b {
  text-align-last: center;
}


p.c {
  text-align-last: justify;
}
```

CSS Text Shadow

The text-shadow property adds shadow to text.

Example

```
h1 {
  text-shadow: 2px 2px;
}
```

Next, add a color (red) to the shadow:

Example

```
h1 {
  text-shadow: 2px 2px red;
}
```

Text Font

Font Selection is Important

Generic Font Families

In CSS there are five generic font families:

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.

All the different font names belong to one of the generic font families.

```
.p1 {
  font-family: "Times New Roman", Times, serif;
}

.p2 {
  font-family: Arial, Helvetica, sans-serif;
}

.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
```

```
}
```

Font Style

The font-style property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example

```
p.normal {
  font-style: normal;
}

p.italic {
  font-style: italic;
}

p.oblique {
  font-style: oblique;
}
```

Font Weight

The font-weight property specifies the weight of a font:

Example

```
p.normal {
  font-weight: normal;
}

p.thick {
  font-weight: bold;
}
```

CSS Font Size

The font-size property sets the size of the text.

The font-size value can be an absolute, or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {
  font-size: 40px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
```

CSS Icons

Icons can easily be added to your HTML page, by using an icon library.

How To Add Icons

The simplest way to add an icon to your HTML page, is with an icon library, such as Font Awesome.

Add the name of the specified icon class to any inline HTML element (like <i> or).

All the icons in the icon libraries below, are scalable vectors that can be customized with CSS (size, color, shadow, etc.)

Font Awesome Icons

To use the Font Awesome icons, go to fontawesome.com, sign in, and get a code to add in the <head> section of your HTML page:

<script src="https://kit.fontawesome.com/*yourcode*.js"crossorigin="anonymous"></script>

Note: No downloading or installation is required! <!DOCTYPE html>

```
<html>


<head>
   <script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous"></script>
</head>


<body>
   <i class="fas fa-cloud"></i>
   <i class="fas fa-heart"></i>
   <i class="fas fa-car"></i>
   <i class="fas fa-file"></i>
   <i class="fas fa-bars"></i>
</body>


</html>
```

Bootstrap Icons

To use the Bootstrap glyphicons, add the following line inside the <head> section of your HTML page:

<linkrel="stylesheet"href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

Note: No downloading or installation is required!

```
<!DOCTYPE html>

<html>



<head>

  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

</head>



<body>



  <i class="glyphicon glyphicon-cloud"></i>

  <i class="glyphicon glyphicon-remove"></i>

  <i class="glyphicon glyphicon-user"></i>

  <i class="glyphicon glyphicon-envelope"></i>

  <i class="glyphicon glyphicon-thumbs-up"></i>



</body>



</html>
```

Google Icons

To use the Google icons, add the following line inside the <head> section of your HTML page:

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

Note: No downloading or installation is required!

```
<!DOCTYPE html>
<html>

<head>
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>

<body>

    <i class="material-icons">cloud</i>
    <i class="material-icons">favorite</i>
    <i class="material-icons">attachment</i>
    <i class="material-icons">computer</i>
    <i class="material-icons">traffic</i>

</body>

</html>
```

CSS Links

With CSS, links can be styled in many different ways.

Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

Example

```
a { color: hotpink; }
```

n addition, links can be styled differently depending on what **state** they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

```
/* 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;
}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Text Decoration

The text-decoration property is mostly used to remove underlines from links:

Example

```
a:link {
  text-decoration: none;
}
```

```
a:visited {
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

a:active {
    text-decoration: underline;
}
```

Background Color

The background-color property can be used to specify a background color for links:

Example

```
a:link {
    background-color: yellow;
 }

 a:visited {
    background-color: cyan;
 }
```

```
a:hover {
  background-color: lightgreen;
}

a:active {
  background-color: hotpink;
}
```

CSS Tables

1. Table Borders

To specify table borders in CSS, use the border property.

The example below specifies a solid border for <table>, <th>, and <td> elements:

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Example

```
table,

th,
```

```
td {
  border: 1px solid;
}
```

Full-Width Table

The table above might seem small in some cases. If you need a table that should span the entire screen (full-width), add width: 100% to the <table> element:

Example

```
table {
  width: 100%;
}
```

Collapse Table Borders

The border-collapse property sets whether the table borders should be collapsed into a single border:

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Example

```
table {
```

```
  border-collapse: collapse;
}
```

CSS Table Size

Table Width and Height

The width and height of a table are defined by the width and height properties.

The example below sets the width of the table to 100%, and the height of the <th> elements to 70px:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
table {
  width: 100%;
}

th {
  height: 70px;
}
```

To create a table that should only span half the page, use width: 50%:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
table {
  width: 50%;
}
```

CSS Table Alignment

1. Horizontal Alignment

The text-align property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.

By default, the content of <th> elements are center-aligned and the content of <td> elements are left-aligned.

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

To center-align the content of <td> elements as well, use text-align: center:

Example

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

To left-align the content, force the alignment of <th> elements to be left-aligned, with the text-align: left property:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
th {
  text-align: left;
```

```
}
```

Vertical Alignment

The vertical-align property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.

By default, the vertical alignment of the content in a table is middle (for both <th> and <td> elements).

The following example sets the vertical text alignment to bottom for <td> elements:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
td {
  height: 50px;
  vertical-align: bottom;
}
```

CSS Table Style

1. Table Padding

To control the space between the border and the content in a table, use the padding property on <td> and <th> elements:

First Name	Last Name	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
th,

td {

  padding: 15px;

  text-align: left;

}
```

Hoverable Table

Use the :hover selector on <tr> to highlight table rows on mouse over:

Example

```
tr:hover {background-color: coral;}
```

Striped Tables

First Name	Last Name	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

For zebra-striped tables, use the nth-child() selector and add a background-color to all even (or odd) table rows:

Example

```
tr:nth-child(even) {background-color: #f2f2f2;}
```

Table Color

The example below specifies the background color and text color of <th> elements:

First Name	Last Name	Savings

Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
th {
  background-color: #04AA6D;
  color: white;
}
```

CSS Layout - The display Property

The display property is the most important CSS property for controlling layout.

The display Property

The display property specifies if/how an element is displayed.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The element is a block-level element.

Examples of block-level elements:

-
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline element inside a paragraph.

Examples of inline elements:

-
- <a>
- <img>

Override The Default Display Value

As mentioned, every element has a default display value. However, you can override this.

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

A common example is making inline <li> elements for horizontal menus:

Example

```
li {
  display: inline;
}
```

Note: Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with display: block; is not allowed to have other block elements inside it.

The following example displays elements as block elements:

Example

```
span {
  display: block;
}
```

The position Property

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

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

position: static;

HTML elements are positioned static by default.

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

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This element has position: static;

Here is the CSS that is used:

Example

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

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.

This element has position: relative;

Here is the CSS that is used:

Example

```
div.relative {
  position: relative;
```

```
  left: 30px;

  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. 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.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example

```
div.fixed {

  position: fixed;

  bottom: 0;

  right: 0;

  width: 300px;

  border: 3px solid #73AD21;

}
```

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.

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

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).

Note: Internet Explorer does not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of top, right, bottom or left for sticky positioning to work.

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.**Example**

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

CSS Overflow

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

- visible - Default. The overflow is not clipped. The content renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
- auto - Similar to scroll, but it adds scrollbars only when necessary

overflow: visible

By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:

```
div {
  width: 200px;
  height: 65px;
  background-color: coral;
  overflow: visible;
}
```

overflow: hidden

With the hidden value, the overflow is clipped, and the rest of the content is hidden:

```
div {
  overflow: hidden;
 }
```

overflow: scroll

Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

```
div {
  overflow: scroll;
}
```

overflow: auto

The auto value is similar to scroll, but it adds scrollbars only when necessary:

```
div {
  overflow: auto;
}
```

The float Property

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

- left - The element floats to the left of its container
- right - The element floats to the right of its container
- none - The element does not float (will be displayed just where it occurs in the text). This is default
- inherit - The element inherits the float value of its parent

In its simplest use, the float property can be used to wrap text around images.

Example - float: right;

The following example specifies that an image should float to the **right** in a text:

```
img {
  float: right;
}
```

Example - float: left;

The following example specifies that an image should float to the **left** in a text:

```
img {
  float: left;
 }
```

Example - No float

In the following example the image will be displayed just where it occurs in the text (float: none;):

```
img {
  float: none;
 }
```

CSS Layout - Horizontal & Vertical Align

Center Align Elements

To horizontally center a block element (like), use margin: auto;

Setting the width of the element will prevent it from stretching out to the edges of its container.

The element will then take up the specified width, and the remaining space will be split equally between the two margins:

```
.center {
  margin: auto;
  width: 50%;
  border: 3px solid green;
  padding: 10px;
}
```

Center Align Text

To just center the text inside an element, use text-align: center;

```
.center {
  text-align: center;
  border: 3px solid green;
}
```

Center an Image

To center an image, set left and right margin to auto and make it into a block element:

```
img {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 40%;
}
```

Left and Right Align - Using position

One method for aligning elements is to use position: absolute;:

```
.right {
  position: absolute;
  right: 0px;
  width: 300px;
  border: 3px solid #73AD21;
  padding: 10px;
}
```

CSS Navigation Bar

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

In our examples we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense:

Example

```
<ul>
  <li><a href="default.asp">Home</a></li>
  <li><a href="news.asp">News</a></li>
  <li><a href="contact.asp">Contact</a></li>
  <li><a href="about.asp">About</a></li>
</ul>
```

Now let's remove the bullets and the margins and padding from the list:

Example

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
  }
```

Example explained:

- list-style-type: none; - Removes the bullets. A navigation bar does not need list markers
- Set margin: 0; and padding: 0; to remove browser default settings

Vertical Navigation Bar

To build a vertical navigation bar, you can style the <a> elements inside the list, in addition to the code from the previous page:

Example

```
li a {
  display: block;
  width: 60px;
}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 60px;
}

li a {
  display: block;
}
```

Vertical Navigation Bar Examples

Create a basic vertical navigation bar with a gray background color and change the background color of the links when the user moves the mouse over them:

Example

```
ul {
```

```
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 200px;
    background-color: #f1f1f1;
}

li a {
    display: block;
    color: #000;
    padding: 8px 16px;
    text-decoration: none;
}

/* Change the link color on hover */

li a:hover {
    background-color: #555;
    color: white;
}
```

Active/Current Navigation Link

Add an "active" class to the current link to let the user know which page he/she is on:

- Home
- News
- Contact
- About

Example

```
.active {
  background-color: #04AA6D;
  color: white;
}
```

Center Links & Add Borders

Add text-align:center to <li> or <a> to center the links.

Add the border property to <ul> add a border around the navbar. If you also want borders inside the navbar, add a border-bottom to all <li> elements, except for the last one:

```
ul {
  border: 1px solid #555;
}

li {
  text-align: center;
  border-bottom: 1px solid #555;
}
```

```
li:last-child {
  border-bottom: none;
}
```

Full-height Fixed Vertical Navbar

Create a full-height, "sticky" side navigation:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  height: 100%;
  /* Full height */
  position: fixed;
  /* Make it stick, even on scroll */
  overflow: auto;
  /* Enable scrolling if the sidenav has too much content */
}
```

Horizontal Navigation Bar

here are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.

Inline List Items

One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard" code from the previous page:

Example

```
li {
  display: inline;
}
```

Example explained:

- display: inline; - By default, <li> elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

Floating List Items

Another way of creating a horizontal navigation bar is to float the <li> elements, and specify a layout for the navigation links:

Example

```
li {
  float: left;
}

a {
  display: block;
  padding: 8px;
  background-color: #dddddd;
}
```

Horizontal Navigation Bar Examples

Create a basic horizontal navigation bar with a dark background color and change the background color of the links when the user moves the mouse over them:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

/* Change the link color to #111 (black) on hover */
```

```
li a:hover {
  background-color: #111;
}
```

Active/Current Navigation Link

Add an "active" class to the current link to let the user know which page he/she is on:

Example

```
.active {
  background-color: #04AA6D;
 }
```

Right-Align Links

Right-align links by floating the list items to the right (float:right;):

```
<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li style="float:right"><a class="active" href="#about">About</a></li>
</ul>
```

CSS Image Gallery

CSS can be used to create an image gallery.

Image Gallery

The following image gallery is created with CSS:

```
<html>


<head>
  <style>
    div.gallery {
      margin: 5px;
      border: 1px solid #ccc;
      float: left;
      width: 180px;
    }

    div.gallery:hover {
      border: 1px solid #777;
    }

    div.gallery img {
      width: 100%;
      height: auto;
    }
```

```
    div.desc {
        padding: 15px;
        text-align: center;
    }
  </style>
</head>

<body>

  <div class="gallery">
    <a target="_blank" href="img_5terre.jpg">
      <img src="img_5terre.jpg" alt="Cinque Terre" width="600" height="400">
    </a>
    <div class="desc">Add a description of the image here</div>
  </div>

  <div class="gallery">
    <a target="_blank" href="img_forest.jpg">
      <img src="img_forest.jpg" alt="Forest" width="600" height="400">
    </a>
    <div class="desc">Add a description of the image here</div>
  </div>

  <div class="gallery">
```

```
    <a target="_blank" href="img_lights.jpg">
      <img src="img_lights.jpg" alt="Northern Lights" width="600" height="400">
    </a>
    <div class="desc">Add a description of the image here</div>
  </div>

  <div class="gallery">
    <a target="_blank" href="img_mountains.jpg">
      <img src="img_mountains.jpg" alt="Mountains" width="600" height="400">
    </a>
    <div class="desc">Add a description of the image here</div>
  </div>

</body>

</html>
```

CSS Forms

Styling Input Fields

Use the width property to determine the width of the input field:

```
input {
  width: 100%;
}
```

Padded Inputs

Use the padding property to add space inside the text field.

Tip: When you have many inputs after each other, you might also want to add some margin, to add more space outside of them:

```
input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
}
```

Bordered Inputs

Use the border property to change the border size and color, and use the border-radius property to add rounded corners:

```
input[type=text] {
  border: 2px solid red;
  border-radius: 4px;
}
```

Colored Inputs

Use the background-color property to add a background color to the input, and the color property to change the text color:

```
input[type=text] {
```

```
  background-color: #3CBC8D;

  color: white;

}
```

Input with icon/image

If you want an icon inside the input, use the background-image property and position it with the background-position property. Also notice that we add a large left padding to reserve the space of the icon:

```
input[type=text] {

  background-color: white;

  background-image: url('searchicon.png');

  background-position: 10px 10px;

  background-repeat: no-repeat;

  padding-left: 40px;

}
```

Styling Textareas

Tip: Use the resize property to prevent textareas from being resized (disable the "grabber" in the bottom right corner):

```
textarea {

  width: 100%;

  height: 150px;

  padding: 12px 20px;

  box-sizing: border-box;
```

```
  border: 2px solid #ccc;
  border-radius: 4px;
  background-color: #f8f8f8;
  resize: none;
}
```

Styling Select Menus

Example

```
select {
  width: 100%;
  padding: 16px 20px;
  border: none;
  border-radius: 4px;
  background-color: #f1f1f1;
}
```

Styling Input Buttons

```
input[type=button],
input[type=submit],
input[type=reset] {
  background-color: #04AA6D;
  border: none;
  color: white;
  padding: 16px 32px;
  text-decoration: none;
```

```
  margin: 4px 2px;

  cursor: pointer;

}
```