Оглавление

[Intro to CSS 2](#_Toc507573532)

[Inline Styles 10](#_Toc507573533)

[The <style> Tag 14](#_Toc507573534)

[The .css file 18](#_Toc507573535)

[Linking the CSS File 18](#_Toc507573536)

[**Tag Name** 19](#_Toc507573537)

[**Class Name** 20](#_Toc507573538)

[**Multiple Classes** 21](#_Toc507573539)

[**ID Name** 22](#_Toc507573540)

[**Classes and IDs** 22](#_Toc507573541)

[**Specificity** 23](#_Toc507573542)

[Chaining Selectors 24](#_Toc507573543)

[Nested Elements 25](#_Toc507573544)

[Chaining and Specificity 26](#_Toc507573545)

[Important 26](#_Toc507573546)

[Multiple Selectors 27](#_Toc507573547)

[**Review CSS Selectors** 28](#_Toc507573548)

[Font Family 28](#_Toc507573549)

[Font Size 29](#_Toc507573550)

[Font Weight 30](#_Toc507573551)

[**Text Align** 30](#_Toc507573552)

[**Color** 31](#_Toc507573553)

[Opacity 31](#_Toc507573554)

[**Background Image** 32](#_Toc507573555)

[**Review Visual Rules** 32](#_Toc507573556)

[**Introduction to the Box Model** 33](#_Toc507573557)

[The Box Model 34](#_Toc507573558)

[Height and Width 34](#_Toc507573559)

[Borders 35](#_Toc507573560)

[Border Radius 35](#_Toc507573561)

[**Padding I** 36](#_Toc507573562)

[Padding II 37](#_Toc507573563)

[**Margins I** 38](#_Toc507573564)

[Margins II 39](#_Toc507573565)

[Auto 40](#_Toc507573566)

[Margin Collapse 40](#_Toc507573567)

[**Minimum and Maximum Height and Width** 41](#_Toc507573568)

[**Overflow** 43](#_Toc507573569)

[Resetting Defaults 44](#_Toc507573570)

[**Visibility** 44](#_Toc507573571)

[**Review** 45](#_Toc507573572)

[Why Change the Box Model? 46](#_Toc507573573)

[Box Model: Content-Box 47](#_Toc507573574)

[**Box Model: Border-Box** 47](#_Toc507573575)

[**The New Box Model** 48](#_Toc507573576)

[Review: Changing the Box Model 49](#_Toc507573577)

# Intro to CSS

Введение в CSS

Базовая структура каждой веб-страницы, HTML, очень проста сама по себе. Красивые веб-сайты, которые вы видите в Интернете, оформлены с использованием различных инструментов, включая CSS.

CSS, или каскадные таблицы стилей, - это язык, который веб-разработчики используют для стилизации содержимого HTML на веб-странице. Если вы заинтересованы в изменении цветов, типов шрифтов, размеров шрифтов, теней, изображений, позиционирования элементов и т. Д., CSS - это инструмент для работы!

На этом уроке вы узнаете, как выбрать элементы HTML, которые вы хотите стилизовать, и настроить структуру файлов CSS.

Взгляните на код и браузер справа. Код представляет собой простой HTML без каких-либо стилей. Давайте кратко рассмотрим возможности CSS.

Скопируйте следующую строку кода, вставьте ее в строку 5 и запустите свой код

<link href="style.css" type="text/css" rel="stylesheet">

Что произошло? Потратьте некоторое время, чтобы изучить и поэкспериментировать с кодом в style.css.

Index.html

<!DOCTYPE html>

<html>

<head>

<link href='https://fonts.googleapis.com/css?family=Roboto:400,300,500,100' rel='stylesheet' type='text/css'>

</head>

<body>

<div class="header">

<div class="container">

<h1>Innovation Cloud</h1>

<p>Connect your ideas globally</p>

<a class="btn" href="#">Learn More</a>

</div>

</div>

<div class="nav">

<div class="container">

<ul>

<li>Register</li>

<li>Schedule</li>

<li>Sponsors</li>

<li>About</li>

<li>Contact</li>

</ul>

</div>

</div>

<div class="main">

<div class="container">

<img src="" height="128" width="196">

<h2>The Innovation Cloud Conference</h2>

<p>Connect with the best minds across a wide range of industries to share ideas and brainstorm new solutions to challenging problems.</p>

<p>Hear industry leaders talk about what worked (and what didn't) so that you can save time on your most challenging projects.</p>

<p>Learn about the latest research and technologies that you can use immediately to invent the future.</p>

</div>

</div>

<div class="jumbotron">

<div class="container">

<h2>Stay Connected</h2>

<p>Receive weekly insights from industry insiders.</p>

<a class="btn" href="#">Join</a>

</div>

</div>

<div class="footer">

<div class="container">

<p>&copy; Innovation Cloud Conference</p>

</div>

</div>

</body>

</html>

Style.css

html, body {

margin: 0;

padding: 0;

}

body {

font-family: 'Roboto', sans-serif;

font-weight: 100;

}

.container {

margin: 0 auto;

max-width: 940px;

padding: 0 10px;

}

.header {

background: url(http://s3.amazonaws.com/codecademy-content/projects/innovation-cloud/bg.jpg) no-repeat center center;

background-size: cover;

height: 800px;

text-align: center;

}

.header .container {

position: relative;

top: 200px;

}

.header h1 {

color: #fff;

line-height: 100px;

font-size: 80px;

margin-top: 0;

margin-bottom: 80px;

text-transform: uppercase;

}

@media (min-width:850px) {

.header h1 {

font-size: 120px;

}

}

.header p {

color: #fff;

font-weight: 500;

letter-spacing: 8px;

margin-bottom: 40px;

margin-top: 0;

text-transform: uppercase;

}

.btn {

color: #fff;

background: #000;

padding: 10px 40px;

text-decoration: none;

transition: background .5s;

}

.nav {

background: #000;

height: 80px;

width: 100%;

}

.nav ul {

height: 80px;

list-style: none;

margin: 0 auto;

padding: 0;

}

.nav ul li {

color: #fff;

display: inline-block;

height: 80px;

line-height: 80px;

list-style: none;

padding: 0 10px;

transition: background .5s;

}

.btn:hover, .nav ul li:hover {

background: #117bff;

cursor: pointer;

transition: background .5s;

}

.main .container {

margin: 80px auto;

}

.main img {

float: left;

margin: 50px 80px 50px 0;

}

.jumbotron {

background: url(http://s3.amazonaws.com/codecademy-content/projects/innovation-cloud/jumbotron\_bg.jpg) center center;

background-size: cover;

height: 600px;

}

.jumbotron .container {

position: relative;

top: 220px;

}

.jumbotron h2 {

color: #fff;

text-align: right;

}

.jumbotron p {

color: #fff;

text-align: right;

}

.jumbotron .btn {

margin: 10px 0 0;

float: right;

}

.footer {

background: #000;

height: 80px;

padding-bottom: 50px;

}

.footer p {

color: #fff;

font-size: 14px;

height: 80px;

line-height: 80px;

margin: 0;

}

@media (max-width: 500px) {

.header h1 {

font-size: 50px;

line-height: 64px;

}

.main, .jumbotron {

padding: 0 30px;

}

.main img {

width: 100%;

}

}

# Inline Styles (Встроенные стили)

Хотя CSS отличается от языка HTML, можно писать код CSS непосредственно в коде HTML, используя встроенные стили.

Чтобы стилизовать HTML-элемент, вы можете добавить атрибут style непосредственно к открывающему тегу. После добавления атрибута вы можете установить его в соответствии со стилем (стилями) CSS, который вы хотите применить к этому элементу

.<p style="color: red;">I'm learning to code!</p>

Код в приведенном выше примере демонстрирует, как использовать встроенные стили. Элемент абзаца имеет атрибут style в своем открывающем теге. Затем атрибут style устанавливается равным color: red ;, который устанавливает цвет текста абзаца на красный в браузере.

Вы можете быть удивлены синтаксисом следующего фрагмента кода: color: red ;. На данный момент детали синтаксиса не важны; вы узнаете больше о синтаксисе CSS в других упражнениях. На данный момент важно знать, что встроенные стили - это быстрый способ напрямую стилизовать HTML-элемент.

Если вы хотите добавить более одного стиля со встроенными стилями, просто продолжайте добавлять к атрибуту стиля. Обязательно заканчивайте стили точкой с запятой (;).

<p style="color: red; font-size: 20px;">I'm learning to code!</p>

Что сделать:

**1.**

В index.html используйте встроенные стили, чтобы установить для семейства шрифтов первого абзаца значение Arial.

Index.html<!DOCTYPE html>

<html>

<head>

<title>Vacation World</title>

<link href="./style.css" type="text/css" rel="stylesheet">

</head>

<body>

<img src="https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/explorer.jpeg" />

<h1 id="article-title" class="title uppercase">Top Vacation Spots</h1>

<h5>By: Stacy Gray</h5>

<h6 class="publish-time">Published: 2 Days Ago</h6>

<p>The world is full of fascinating places. Planning the perfect vacation involves packing up, leaving home, and experiencing something new.</p>

<h2 class="destination">1. Florence, Italy</h2>

<div class="description">A city-size shrine to the Renaissance, Florence offers frescoes, sculptures, churches, palaces, and other monuments from the richest cultural flowering the world has known. Names from its dazzling historical pastDante, Michelangelo, Galileo, Machiavelliare some of the most resonant of the medieval age. <a href="http://travel.nationalgeographic.com/travel/city-guides/florence-italy/" target="\_blank">Learn More</a>.

<h5>Top Attractions</h5>

<ul>

<li>Museums</li>

<li>Bike Tours</li>

<li>Historical Monuments</li>

</ul>

</div>

<h2 class="destination">2. Beijing, China</h2>

<div class="description">A city in the midst of reinventing itself and continuing to build on the success of the 2008 Summer Olympics, Beijing is a place of frenzied construction. New housing, new roads, and new sports venues seem to spring up overnight. At the same time, the capital of the Peoples Republic of China remains an epicenter of tradition, with the treasures of nearly 2,000 years as the imperial capital still on viewin the famed Forbidden City and in the luxuriant pavilions and gardens of the Summer Palace.

<a href="http://travel.nationalgeographic.com/travel/city-guides/beijing-china/" target="\_blank">Learn More</a>.

<h5>Top Attractions</h5>

<ul>

<li>Biking</li>

<li>Historical Sites</li>

<li>Restaurants and Dining</li>

</ul>

</div>

<h2 class="destination">3. Seoul, South Korea</h2>

<div class="description">The Korean capital is a city of contrasts. Fourteenth-century city gates squat in the shadow of 21st-century skyscrapers, while the broad Han River is back-dropped by granite mountains rising in the city centercomplete with alpine highways speeding around their contours and temples nestling among their crags. Fashionable, gadget-laden youths battle for sidewalk space with fortune-tellers and peddlers, while tiny neighborhoods of traditional cottages contrast with endless ranks of identical apartments.

<a href="http://travel.nationalgeographic.com/travel/city-guides/seoul-south-korea/" target="\_blank">Learn More</a>.

<h5>Top Attractions</h5>

<ul>

<li>Parasailing</li>

<li>Segway Tours</li>

<li>Spas and Resorts</li>

</ul>

</div>

<h2> More Desinations </h2>

<ul>

<li><h4 class="destination">Jackson Hole, Wyoming</h4></li>

<li><h4 class="destination">Cape Town, South Africa</h4></li>

<li><h4 class="destination">La Paz, Bolivia</h4></li>

</ul>

<p>&mdash;Best of luck with your travels, and be sure to send pictures and stories. We"d love to hear them!</p>

</body>

</html>

# The <style> Tag(Тег <style>)

Встроенные стили - это быстрый способ стилизации HTML, но они также имеют ограничения. Если вы хотите стилизовать, например, несколько элементов <h1>, вам придется добавить встроенный стиль для каждого элемента вручную. Кроме того, вам также придется поддерживать HTML-код при добавлении дополнительных элементов <h1>.

К счастью, HTML позволяет вам писать код CSS в своем собственном выделенном разделе с элементом <style>. CSS можно записывать между открывающими и закрывающими тегами <style>. Чтобы использовать элемент <style>, он должен быть размещен внутри элемента <head>.<head>

<style>

</style>

</head>

After adding a <style> tag in the head section, you can begin writing CSS code.

<head>

<style>

p {

color: red;

font-size: 20px;

}

</style>

</head>

The CSS code in the example above changes the color of all paragraph text to red and also changes the size of the text to 20 pixels. Note how the syntax of the CSS code matches (for the most part) the syntax you used for inline styling. The main difference is that you can specify which elements to apply the styling to.

Again, the details of the CSS syntax in the example above aren't important at the moment. You will learn more about the details of CSS syntax in later lessons.

Instructions

**1.**

First, add a <style> element in the head of **index.html**. Then, make sure to delete the inline styles that you added to th

e paragraph.

**2.**

Add the inline styles that you removed from the <p> element to the <style> element in the head.

Index.html

<!DOCTYPE html>

<html>

<head>

<title>Vacation World</title>

<link href="./style.css" type="text/css" rel="stylesheet">

</head>

<body>

<img src="https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/explorer.jpeg" />

<h1 id="article-title" class="title uppercase">Top Vacation Spots</h1>

<h5>By: Stacy Gray</h5>

<h6 class="publish-time">Published: 2 Days Ago</h6>

<p>The world is full of fascinating places. Planning the perfect vacation involves packing up, leaving home, and experiencing something new.</p>

<h2 class="destination">1. Florence, Italy</h2>

<div class="description">A city-size shrine to the Renaissance, Florence offers frescoes, sculptures, churches, palaces, and other monuments from the richest cultural flowering the world has known. Names from its dazzling historical pastDante, Michelangelo, Galileo, Machiavelliare some of the most resonant of the medieval age. <a href="http://travel.nationalgeographic.com/travel/city-guides/florence-italy/" target="\_blank">Learn More</a>.

<h5>Top Attractions</h5>

<ul>

<li>Museums</li>

<li>Bike Tours</li>

<li>Historical Monuments</li>

</ul>

</div>

<h2 class="destination">2. Beijing, China</h2>

<div class="description">A city in the midst of reinventing itself and continuing to build on the success of the 2008 Summer Olympics, Beijing is a place of frenzied construction. New housing, new roads, and new sports venues seem to spring up overnight. At the same time, the capital of the Peoples Republic of China remains an epicenter of tradition, with the treasures of nearly 2,000 years as the imperial capital still on viewin the famed Forbidden City and in the luxuriant pavilions and gardens of the Summer Palace.

<a href="http://travel.nationalgeographic.com/travel/city-guides/beijing-china/" target="\_blank">Learn More</a>.

<h5>Top Attractions</h5>

<ul>

<li>Biking</li>

<li>Historical Sites</li>

<li>Restaurants and Dining</li>

</ul>

</div>

<h2 class="destination">3. Seoul, South Korea</h2>

<div class="description">The Korean capital is a city of contrasts. Fourteenth-century city gates squat in the shadow of 21st-century skyscrapers, while the broad Han River is back-dropped by granite mountains rising in the city centercomplete with alpine highways speeding around their contours and temples nestling among their crags. Fashionable, gadget-laden youths battle for sidewalk space with fortune-tellers and peddlers, while tiny neighborhoods of traditional cottages contrast with endless ranks of identical apartments.

<a href="http://travel.nationalgeographic.com/travel/city-guides/seoul-south-korea/" target="\_blank">Learn More</a>.

<h5>Top Attractions</h5>

<ul>

<li>Parasailing</li>

<li>Segway Tours</li>

<li>Spas and Resorts</li>

</ul>

</div>

<h2> More Desinations </h2>

<ul>

<li><h4 class="destination">Jackson Hole, Wyoming</h4></li>

<li><h4 class="destination">Cape Town, South Africa</h4></li>

<li><h4 class="destination">La Paz, Bolivia</h4></li>

</ul>

<p>&mdash;Best of luck with your travels, and be sure to send pictures and stories. We"d love to hear them!</p>

</body>

</html>

# The .css file

Developers avoid mixing code by storing HTML and CSS code in separate files (HTML files contain only HTML code, and CSS files contain only CSS code).

You can create a CSS file by using the **.css** file name extension, like so: **style.css**

With a CSS file, you can write all the CSS code needed to style a page without sacrificing the readability and maintainability of your HTML file.

Instructions

**1.**

Take a look at **index.html**. Cut the CSS code in between the opening and closing <style> tags and paste it directly in the new file called **style.css**.

Make sure to delete the remaining <style> element (now empty) from **index.html**.

# Linking the CSS File

Perfect! We successfully separated structure (HTML) from styling (CSS), but the web page still looks bland. Why?

When HTML and CSS code are in separate files, the files must be linked. Otherwise, the HTML file won't be able to locate the CSS code, and the styling will not be applied.

You can use the <link> element to link HTML and CSS files together. The <link> element must be placed within the head of the HTML file. It is a self-closing tag and requires the following three attributes:

1. href — like the anchor element, the value of this attribute must be the address, or path, to the CSS file.
2. type — this attribute describes the type of document that you are linking to (in this case, a CSS file). The value of this attribute should be set to text/css.
3. rel — this attribute describes the relationship between the HTML file and the CSS file. Because you are linking to a stylesheet, the value should be set to stylesheet.

When linking an HTML file and a CSS file together, the <link> element will look like the following:

<link href="https://www.codecademy.com/stylesheets/style.css" type="text/css" rel="stylesheet">

Note that in the example above the path to the stylesheet is a URL:

https://www.codecademy.com/stylesheets/style.css

Specifying the path to the stylesheet using a URL is one way of linking a stylesheet.

If the CSS file is stored in the same [directory](https://en.wikipedia.org/wiki/Directory_(computing)) as your HTML file, then you can specify a [relative path](https://en.wikipedia.org/wiki/Path_(computing)#Absolute_and_relative_paths) instead of a URL, like so:

<link href="./style.css" type="text/css" rel="stylesheet">

Using a relative path is very common way of linking a stylesheet.

**1.**

Let's link the stylesheet **style.css** to the HTML file **index.html**.

First, add a <link> element within the <head> section.

**2.**

Next, add the href attribute to the <link> element and set it equal to style.css.

Take a look at the web page in the browser to the right. Do you notice any changes?

**3.**

Next, add the type attribute and set it to the correct value.

**4.**

Finally, add the rel attribute and set it to the correct value.

**Tag Name**

CSS can select HTML elements by using an element's tag name. A tag name is the word (or character) between HTML angle brackets.

For example, in HTML, the tag for a paragraph element is <p>. The CSS syntax for selecting <p> elements is:

p {

}

In the example above, all paragraph elements will be selected using a CSS *selector*. The selector in the example above is p. Note that the CSS selector matches the HTML tag for that element, but without the angle brackets.

In addition, two curly braces follow immediately after the selector (an opening and closing brace, respectively). Any CSS properties will go inside of the curly braces to style the selected elements.

**1.**

In **style.css**, add a selector for <h1> elements.

**Note:** The content of the web page will update because we've already linked **index.html** and **style.css** for you.

**2.**

Inside the curly braces of the h1 selector you just declared, write:

color: maroon;

This code will make the text color of all <h1> tags maroon.

**Class Name**

CSS is not limited to selecting elements by tag name. HTML elements can have more than just a tag name; they can also have *attributes*. One common attribute is the class attribute. It's also possible to select an element by its class attribute.

For example, consider the following HTML:

<p class="brand">Sole Shoe Company</p>

The paragraph element in the example above has a class attribute within the <p> tag. The class attribute is set to "brand". To select this element using CSS, we could use the following CSS selector:

.brand {

}

To select an HTML element by its class using CSS, a period (.) must be prepended to the class's name. In the example above case, the class is brand, so the CSS selector for it is .brand.

**1.**

In **style.css**, add a CSS selector for the HTML element with a class of title.

**2.**

Inside the curly braces of the .title selector you just declared, write:

color: teal;

This code will change the color of the title to teal, since the title h1 element has a class of title in the HTML. You can see the HTML element by navigating to **index.html** on line 11.

We'll see in a later exercise why using .title overrides the h1 selector.

**Multiple Classes**

We can use CSS to select an HTML element's class attribute by name.

So far, we've selected elements using only one class name per element. If every HTML element had a single class, all the style information for each element would require a new class.

Luckily, it's possible to add more than one class name to an HTML element's class attribute.

For instance, perhaps there's a heading element that needs to be green and bold. You could write two CSS rules like so:

.green {

color: green;

}

.bold {

font-weight: bold;

}

Then, you could include both of these classes on one HTML element like this:

<h1 class="green bold"> ... </h1>

We can add multiple classes to an HTML element's class attribute by separating them with a space. This enables us to mix and match CSS classes to create many unique styles without writing a custom class for every style combination needed.

**1.**

In **style.css**, add a class selector that will make the title of the page stand out more by making all of its letters uppercased. Write a class named .uppercase. Then, write this inside of its curly braces:

text-transform: uppercase;

**2.**

Now you can add the class to the title element. Navigate to **index.html**. On line 11, there is a <h1> element that has a class of title. Add the uppercase class to this element.

**ID Name**

If an HTML element needs to be styled uniquely (no matter what classes are applied to the element), we can add an ID to the element. To add an ID to an element, the element needs an id attribute:

<h1 id="large-title"> ... </h1>

Then, CSS can select HTML elements by their id attribute. To select an id element, CSS prepends the id name with a hashtag (#). For instance, if we wanted to select the HTML element in the example above, it would look like this:

#large-title {

}

The id name is large-title, therefore the CSS selector for it is #large-title.

**1.**

In **style.css**, add a CSS selector for an element with an id of article-title. Inside of its curly braces, write:

font-family: cursive;

text-transform: capitalize;

These two CSS attributes will make the font cursive and will capitalize the first letter of each word, while lowercasing the rest.

**2.**

Navigate to **index.html**. On line 11, add an id attribute to the h1 element, and include article-title as its id. You'll see the title change to a cursive font that is not all uppercased.

**Classes and IDs**

CSS can select HTML elements by their tag, class, and ID. CSS classes and IDs have different purposes, which can affect which one you use to style HTML elements.

CSS classes are meant to be reused over many elements. By writing CSS classes, you can style elements in a variety of ways by mixing classes on HTML elements.

For instance, imagine a page with two headlines. One headline needs to be bold and blue, and the other needs to be bold and green. Instead of writing separate CSS rules for each headline that repeat each other's code, it's better to write a .bold CSS rule, a .green CSS rule, and a .blue CSS rule. Then you can give one headline the bold green classes, and the other the bold blue classes.

While classes are meant to be used many times, an ID is meant to style only one element. As we'll learn in the next exercise, IDs override the styles of tags and classes. Since IDs override class and tag styles, they should be used sparingly and only on elements that need to always appear the same.

**1.**

On line 13 of **index.html**, there’s an element that displays the time the article on the page was published.

Add a class attribute, with a class of publish-time.

**2.**

Add a publish-time class selector in **style.css** and make its text color gray by writing this within the CSS rule’s body:

color: gray;

**Specificity**

Specificity is the order by which the browser decides which CSS styles will be displayed. A best practice in CSS is to style elements while using the lowest degree of specificity, so that if an element needs a new style, it is easy to override.

IDs are the most specific selector in CSS, followed by classes, and finally, tags. For example, consider the following HTML and CSS:

<h1 class="headline">Breaking News</h1>

h1 {

color: red;

}

.headline {

color: firebrick;

}

In the example code above, the color of the heading would be set to firebrick, as the class selector is more specific than the tag selector. If an ID attribute (and selector) were added to the code above, the styles within the ID selector's body would override all other styles for the heading. The only way to override an ID is to add *another* ID with additional styling.

Over time, as files grow with code, many elements may have IDs, which can make CSS difficult to edit, since a new, more specific style must be created to change the style of an element.

To make styles easy to edit, it's best to style with a tag selector, if possible. If not, add a class selector. If that is not specific enough, then consider using an ID selector.

**1.**

In **index.html**, the element on line 11 has an h1 tag, two classes, and an ID. Since the ID is more specific than both, its styles will be applied to the element. Let's re-write the ID of this element to be less specific by creating classes.

In **index.html**, delete the id attribute on the h1 element on line 11.

**2.**

Now delete the #article-title ID in the CSS.

Navigate to **style.css** delete the #article-id ID selector and its contents.

**3.**

Navigate to **style.css**. Add a class selector named .cursive. Inside its body, write:

font-family: cursive;

**4.**

Add another class selector named .capitalize. In its curly braces, write:

text-transform: capitalize;

**5.**

Now, navigate back to **index.html**, and replace the uppercase class with the cursive and capitalize classes on the h1 element on line 11.

# Chaining Selectors

When writing CSS rules, it's possible to require an HTML element to have two or more CSS selectors at the same time.

This is done by combining multiple selectors, which we will refer to as chaining. For instance, if there was a .special class for h1 elements, the CSS would look like:

h1.special {

}

The code above would select only the h1 elements that have a class of special. If a p element also had a class of special, the rule in the example would not style the paragraph.

**1.**

Let's use chaining to select the destinations to add a style to them.

In **style.css**, write a CSS selector for h2 elements with a class of .destination. Inside the selector's curly braces, write this:

font-family: cursive;

This will make the destinations cursive, like the title of the article.

# Nested Elements

In addition to chaining selectors to select elements, CSS also supports selecting elements that are nested within other HTML elements. For instance, consider the following HTML:

<ul class='main-list'>

<li> ... </li>

<li> ... </li>

<li> ... </li>

</ul>

The nested <li> elements are selected with the following CSS:

.main-list li {

}

In the example above, .main-list selects the .main-list element (the unordered list element). The nested <li> are selected by adding li to the selector, separated by a space, resulting in .main-list li as the final selector (note the space in the selector).

Selecting elements in this way can make our selectors even more specific by making sure they appear in the context we expect.

**1.**

In **index.html**, each destination has a description paragraph below it. Inside each description, there's a list of attractions. Let’s select the Top Attractions element and make it stand out more by making it teal.

Navigate to **style.css**. Add a selector that targets all of the h5 elements nested inside elements with class .description.

**2.**

Inside the curly braces of the selector, write:

color: teal;

# Chaining and Specificity

In the last exercise, instead of selecting all h5 elements, you selected only the h5 elements nested inside the .description elements. This CSS selector was more specific than writing only h5. Adding more than one tag, class, or ID to a CSS selector increases the specificity of the CSS selector.

For instance, consider the following CSS:

p {

color: blue;

}

.main p {

color: red;

}

Both of these CSS rules define what a p element should look like. Since .main p has a class and a p tag as its selector, only the p elements inside the .main element will appear red. This occurs despite there being another more general rule that states p elements should be blue.

**1.**

In **style.css**, write a selector for h5 elements. Inside of the curly braces write:

color: rebeccapurple;

Notice that the h5 elements in the descriptions will not change color. They will continue to be teal.

This is due to there being a more specific selector for h5 elements that you wrote in the last exercise. Because of the more specific CSS selector (.description h5), the more general selector of h5 will not take hold.

# Important

There is one thing that is even more specific than IDs: !important. !important can be applied to specific attributes instead of full rules. It will override any style no matter how specific it is. As a result, it should almost never be used. Once !important is used, it is very hard to override.

The syntax of !important in CSS looks like this:

p {

color: blue !important;

}

.main p {

color: red;

}

Since !important is used on the p selector’s color attribute, all p elements will appear blue, even though there is a more specific .main p selector that sets the color attribute to red.

The !important flag is only useful when an element appears the same way 100% of the time. Since it's almost impossible to guarantee that this will be true throughout a project and over time, it's best to avoid !important altogether. If you ever see !important used (or are ever tempted to use it yourself) we strongly recommend reorganizing your CSS. Making your CSS more flexible will typically fix the immediate problem and make your code more maintainable in the long run.

**1.**

Add !important to the h5 selector's color attribute that you defined in the last exercise. !important should go after rebeccapurple, and before the semicolon.

Notice that the h5 elements will now be rebeccapurple instead of teal. That's because !important will override any other style no matter what.

# Multiple Selectors

In order to make CSS more concise, it's possible to add CSS styles to multiple CSS selectors all at once. This prevents writing repetitive code.

For instance, the following code has repetitive style attributes:

h1 {

font-family: Georgia;

}

.menu {

font-family: Georgia;

}

Instead of writing font-family: Georgia twice for two selectors, we can separate the selectors by a comma to apply the same style to both, like this:

h1,

.menu {

font-family: Georgia;

}

By separating the CSS selectors with a comma, both the h1 and the .menu elements will receive the font-family: Georgia styling.

**1.**

Write selectors for the h5 and p elements so they both will be styled with the same CSS rule. Apply this style to both elements:

font-family: Georgia;

Notice that the font across the page will change to Georgia without writing the same CSS rule twice.

**Review CSS Selectors**

Throughout this lesson, you learned how to select HTML elements with CSS and apply styles to them. Let's review what you learned:

* CSS can change the look of HTML elements. In order to do this, CSS must select HTML elements, then apply styles to them.
* CSS can select HTML elements by tag, class, or ID.
* Multiple CSS classes can be applied to one HTML element.
* Classes can be reusable, while IDs can only be used once.
* IDs are more specific than classes, and classes are more specific than tags. That means IDs will override any styles from a class, and classes will override any styles from a tag selector.
* Multiple selectors can be chained together to select an element. This raises the specificity, but can be necessary.
* Nested elements can be selected by separating selectors with a space.
* The !important flag will override any style, however it should almost never be used, as it is extremely difficult to override.
* Multiple unrelated selectors can receive the same styles by separating the selector names with commas.

Great work this lesson. With this knowledge, you'll be able to use CSS to change the look and feel of websites to make them look great.

# Font Family

If you've ever used a formatted word processor, chances are that you probably also used a feature that allowed you change the font you were typing in. Font refers to the technical term [typeface](https://en.wikipedia.org/wiki/Typeface), or font family.

To change the typeface of text on your web page, you can use the font-family property.

h1 {

font-family: Garamond;

}

In the example above, the font family for all main heading elements has been set to Garamond.

When setting typefaces on a web page, keep the following points in mind:

1. The font specified in a stylesheet must be installed on a user's computer in order for that font to display when a user visits the web page.
2. The default typeface for all HTML elements is Times New Roman. You may be familiar with this typeface if you have ever used a formatted word processor. If no font-family attribute is defined, the page will appear in Times New Roman.
3. It's a good practice to limit the number of typefaces used on a web page to 2 or 3. This helps the page load faster in some cases and is usually a good design decision.
4. When the name of a typeface consists of more than one word, it's a best practice to enclose the typeface's name in quotes, like so:

h1 {

font-family: "Courier New";

}

You can find a reference of web safe fonts [here](http://www.cssfontstack.com/).

Instructions

**1.**

Inside **style.css**, add the font family of the main heading (h1) and subheading (h2) to Georgia.

**2.**

Next, change the font family of the paragraph to Helvetica.

# Font Size

Changing the typeface isn't the only way to customize text. Often times, different sections of a web page are highlighted by modifying the font size.

To change the size of text on your web page, you can use the font-size property.

p {

font-size: 18px;

}

In the example above, the font-size of all paragraphs was set to 18px. px means pixels and is a way to measure font size.

Instructions

**1.**

In **style.css**, set the font-size of paragraph elements to 18 pixels.

# Font Weight

In CSS, the font-weight property controls how bold or thin text appears.

p {

font-weight: bold;

}

In the example above, all paragraphs on the web page would appear bolded.

The font-weight property has a another value: normal. Why does it exist?

If we wanted all text on a web page to appear bolded, we could select all text elements and change their font weight to bold. If a certain section of text was required to appear normal, however, we could set the font weight of that particular element to normal, essentially shutting off bold for that element.

Instructions

**1.**

In **style.css**, set the font weight of paragraph elements to bold.

**Text Align**

No matter how much styling is applied to text (typeface, size, weight, etc.), text always appears on the left side of the browser.

To align text we can use the text-align property. The text-align property will align text to the element that holds it, otherwise known as its *parent*.

h1 {

text-align: right;

}

The text-align property can be set to one of the following three values:

1. left — aligns text to the left hand side of its parent element, which in this case is the browser.
2. center — centers text inside of its parent element.
3. right — aligns text to the right hand side of its parent element.

Instructions

**1.**

In **style.css**, set the text-align property of the main heading so that it appears in the center.

**Color**

Before discussing the specifics of color, it's important to make two distinctions about color. Color can affect the following design aspects:

* Foreground color
* Background color

Foreground color is the color that an element appears in. For example, when a heading is styled to appear green, the *foreground color* of the heading has been styled.

Conversely, when a heading is styled so that its background appears yellow, the *background color* of the heading has been styled.

In CSS, these two design aspects can be styled with the following two properties:

* color: this property styles an element's foreground color
* background-color: this property styles an element's background color

h1 {

color: red;

background-color: blue;

}

In the example above, the text of the heading will appear in red, and the background of the heading will appear blue.

Instructions

**1.**

In **style.css**, set the background color in the .caption selector to white.

**2.**

Then, in the same class selector, set the color of the text to black. Observe the result in the caption on the picture at the bottom of the page.

# Opacity

Opacity is the measure of how transparent an element is. It's measured from 0 to 1, with 1 representing 100%, or fully visible and opaque, and 0 representing 0%, or fully invisible.

Opacity can be used to make elements fade into others for a nice overlay effect. To adjust the opacity of an element, the syntax looks like this:

.overlay {

opacity: 0.5;

}

In the example above, the .overlay element would be 50% visible, letting whatever is positioned behind it show through.

Instructions

**1.**

Make the .caption class transparent by adding an opacity attribute with a value of 0.75.

**Background Image**

CSS has the ability to change the background of an element. One option is to make the background of an element an image. This is done through the CSS property background-image. Its syntax looks like this:

.main-banner {

background-image: url("https://www.example.com/image.jpg");

}

1. The background-image property will set the element's background to display an image.
2. The value provided to background-image is a url. The url should be a url to an image. The url can be a file within your project, or it can be a link to an external site. To link to an image inside an existing project, you must provide a relative file path. If there was an image folder in the project, with an image named mountains.jpg, the relative file path would look like:

.main-banner {

background-image: url("images/mountains.jpg");

}

Instructions

**1.**

In **style.css**, change the background image of the .image class. Use the following URL:

https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/soccer.jpeg

**Review Visual Rules**

Incredible work! You used CSS to alter text and images throughout a website. Throughout this lesson, you learned concepts including:

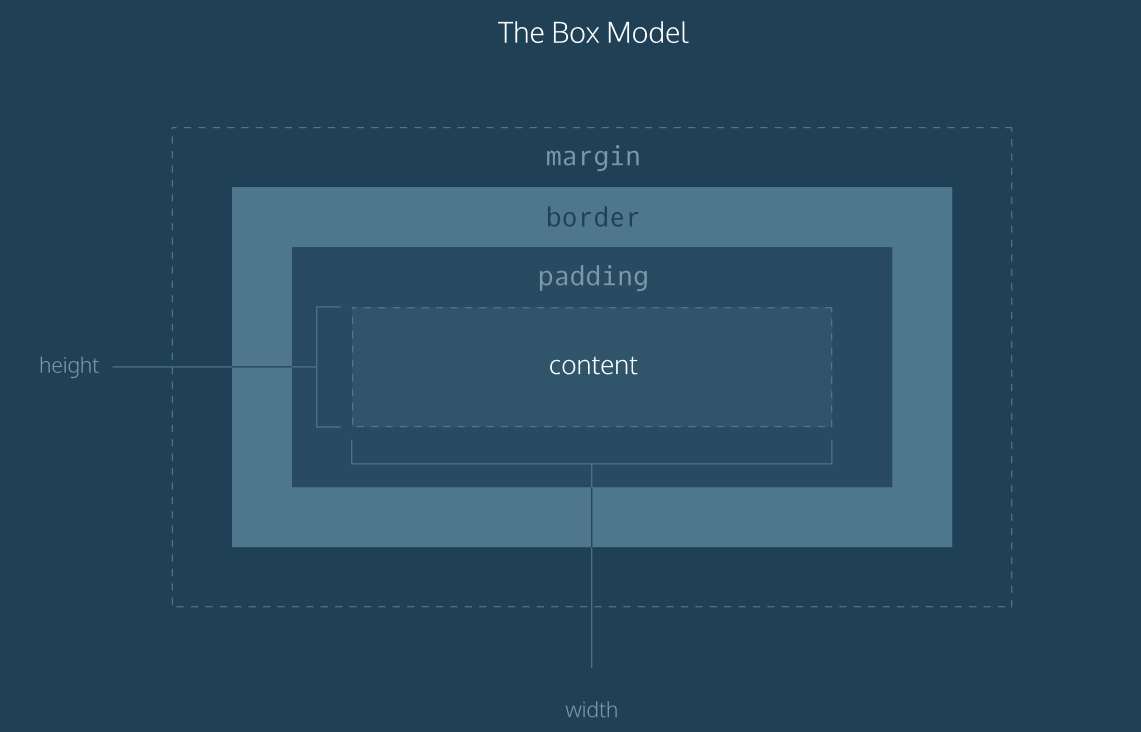
* CSS declarations are structured into property and value pairs.
* The font-family property defines the typeface of an element.
* font-size controls the size of text displayed.
* font-weight defines how thin or thick text is displayed.
* The text-align property places text in the left, right, or center of its parent container.
* Text can have two different color attributes: color and background-color. color defines the color of the text, while background-color defines the color behind the text.
* CSS can make an element transparent with the opacity property.
* CSS can also set the background of an element to an image with the background-image property.

**Introduction to the Box Model**

Browsers load HTML elements with default position values. This often leads to an unexpected and unwanted user experience, while limiting the views you can create. In this lesson you will learn about the *box model*, an important concept to understand how elements are positioned and displayed on a website.

If you have used HTML and CSS, you have unknowingly seen aspects of the box model. For example, if you have set the background color of an element, you may have noticed that the color was applied not only to the area directly behind the element, but also to the area to the right of the element. Also, if you have aligned text, you know it is aligned relative to something. What is that something?

All elements on a web page are interpreted by the browser as "living" inside of a box. This is what is meant by the box model.



For example, when you change the background color of an element, you change the background color of its entire box.

In this lesson, you'll learn about the following aspects of the box model:

1. The dimensions of an element's box.
2. The borders of an element's box.
3. The paddings of an element's box.
4. The margins of an element's box.

Let's begin!

Instructions

Take some time to edit the code to the right. See if you can figure out how these following properties impact an element's display:

1. height
2. width
3. padding
4. border
5. margin
6. overflow

# The Box Model

The box model comprises the set of properties which define parts of an element that take up space on a web page. The model includes the content area's size (width and height) and the element's padding, border, and margin. The properties include:

1. Width and height — specifies the width and height of the content area.
2. Padding — specifies the amount of space between the content area and the border.
3. Border — specifies the thickness and style of the border surrounding the content area and padding.
4. Margin — specifies the amount of space between the border and the outside edge of the element.

The image to the right is a visual representation of the box model.

Open [this](https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-4/diagram-boxmodel.svg) image in a new tab so you can reference the box model as you move through the lesson.

# Height and Width

An element's content has two dimensions: a height and a width. By default, the dimensions of an HTML box are set to hold the raw contents of the box.

The CSS height and width properties can be used to modify these default dimensions.

p {

height: 80px;

width: 240px;

}

In this example, the height and width of paragraph elements are set to 80 pixels and 240 pixels, respectively — the px in the code above stands for pixels.

Pixels allow you to set the exact size of an element's box (width and height). When the width and height of an element are set in pixels, it will be the same size on all devices — an element that fills a laptop screen will overflow a mobile screen.

**1.**

Add a height of 700 pixels to #banner.

**2.**

Set .pull-quote width to 350 pixels.

**3.**

Set the #banner .content h1 width to 400 pixels.

# Borders

A border is a line that surrounds an element, like a frame around a painting. Borders can be set with a specific width, style, and color.

1. width — The thickness of the border. A border's thickness can be set in pixels or with one of the following keywords: thin, medium, or thick.
2. style — The design of the border. Web browsers can render any of [10 different styles](https://developer.mozilla.org/en-US/docs/Web/CSS/border-style#Values). Some of these styles include: none, dotted, and solid.
3. color — The color of the border. Web browsers can render colors using a few different formats, including [140 built-in color keywords](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value).

p {

border: 3px solid coral;

}

In the example above, the border has a width of 3 pixels, a style of solid and a color of coral. All three properties are set in one line of code.

The default border is medium none color, where color is the current color of the element. If width, style, or color are not set in the CSS file, the web browser assigns the default value for that property.

p.content-header {

height: 80px;

width: 240px;

border: solid coral;

}

In this example, the border style is set to solid and the color is set to coral. The width is not set, so it defaults to medium.

Instructions

**1.**

Add a dotted 1 pixel red border to all h2 headings.

**2.**

Add a border to the #banner .content h1 rule so it looks like [this](https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-4/img-header_border.png).

The border width is 3 pixels.

# Border Radius

Ever since we revealed the borders of boxes, you may have noticed that the borders highlight the true shape of an element's box: square. Thanks to CSS, a border doesn't have to be square.

You can modify the corners of an element's border box with the border-radius property.

div.container {

border: 3px solid rgb(22, 77, 100);

border-radius: 5px;

}

The code in the example above will set all four corners of the border to a radius of 5 pixels (i.e. the same curvature that a circle with radius 5 pixels would have).

You can create a border that is a perfect circle by setting the radius equal to the height of the box, or to 100%.

div.container {

height: 60px;

width: 60px;

border: 3px solid rgb(22, 77, 100);

border-radius: 100%;

}

The code in the example above creates a div that is a perfect circle.

Instructions

**1.**

In **style.css**, set the border radius of #banner .content h1 to 15 pixels.

**Padding I**

The space between the contents of a box and the borders of a box is known as *padding*. Padding is like the space between a picture and the frame surrounding it. In CSS, you can modify this space with the padding property.

p.content-header {

border: 3px solid coral;

padding: 10px;

}

The code in this example puts 10 pixels of space between the content of the paragraph (the text) and the borders, on all four sides.

The padding property is often used to expand the background color and make content look less cramped.

If you want to be more specific about the amount of padding on each side of a box's content, you can use the following properties:

1. padding-top
2. padding-right
3. padding-bottom
4. padding-left

Each property affects the padding on only one side of the box's content, giving you more flexibility in customization.

p.content-header {

border: 3px solid fuschia;

padding-bottom: 10px;

}

In the example above, only the bottom side of the paragraph's content will have a padding of 10 pixels.

**1.**

In one line, set the .navigation li elements to have 20 pixels of padding. Click Run and observe their change.

**2.**

Look at the red boxes at the bottom of the web page. Set the .share a elements to have 14 pixels of padding. Observe how the red boxes at the bottom of the page changed.

**3.**

Set the top and bottom padding of h2 elements to 20 pixels and set the left and right padding of h2 elements to 30 pixels.

# Padding II

Another implementation of the padding property lets you specify exactly how much padding there should be on each side of the content in a single declaration.

p.content-header {

border: 3px solid grey;

padding: 6px 11px 4px 9px;

}

In the example above, the four values 6px 11px 4px 9px correspond to the amount of padding in a clockwise rotation. In order, it specifies the amount of padding on the top (6 pixels), right (11 pixels), bottom (4 pixels), and left (9 pixels) sides of the content.

When using this implementation of the padding property, we must specify a padding value for all four sides of the element.

However, if the top and bottom values for padding will equal each other, and the left and right values for padding will also equal each other, you can use the following shortcut:

p.content-header {

padding: 5px 10px;

}

The first value, 5px, sets the padding value for the top and bottom sides of the content. The second value, 10px, sets the padding value for the left and right sides of the content.

Instructions

**1.**

Change the h2 paddings so they are set in one line of CSS, using two values.

**2.**

Using two values for the padding property, set the paragraph padding to 10 pixels on the top and bottom and 20 pixels on the left and right.

**Margins I**

So far you've learned about the following components of the box model: content, borders, and padding. The fourth and final component of the box model is *margin*.

Margin refers to the space directly outside of the box. The margin property is used to specify the size of this space.

p {

border: 1px solid aquamarine;

margin: 20px;

}

The code in the example above will place 20 pixels of space on the outside of the paragraph's box on all four sides. This means that other HTML elements on the page cannot come within 20 pixels of the paragraph's border.

If you want to be even more specific about the amount of margin on each side of a box, you can use the following properties:

1. margin-top
2. margin-right
3. margin-bottom
4. margin-left

Each property affects the margin on only one side of the box, providing more flexibility in customization.

p {

border: 3px solid DarkSlateGrey;

margin-right: 15px;

}

In the example above, only the right side of the paragraph's box will have a margin of 15 pixels. It's common to see margin values used for a specific side of an element.

Instructions

**1.**

Set the top margin of p elements to 60 pixels.

**2.**

Look at the three red boxes at the bottom of the web page. These elements are anchor elements of class .share. Set these .share a elements to have a margin of 10 pixels

# Margins II

What if you don't want equal margins on all four sides of the box?

A similar implementation of the margin property is used to specify exactly how much margin there should be on each side of the box in a single declaration.

p {

margin: 6px 10px 5px 12px;

}

In the example above, the four values 6px 10px 5px 12px refer to the amount of margin around the box in a clockwise rotation. In order, it specifies the amount of margin on the top (6 pixels), right (10 pixels), bottom (5 pixels), and left (12 pixels) sides of the box.

When using this implementation of the margin property, the margin value must be specified for all four sides of the box.

Just like the padding shortcut, when you're certain that the top and bottom values for margin will equal each other, and that the left and right values for margin will also equal each other, you can use the following shortcut:

p {

margin: 6px 12px;

}

The first value, 6px, sets a margin value for the top and bottom of the box. The second value, 12px, sets a margin value for the left and right sides of the box.

Instructions

**1.**

Using two values, set the h2 top and bottom margins to 30 pixels and the left and right margins to 20 pixels.

# Auto

The margin property also lets you center content. However, you must follow a few syntax requirements. Take a look at the following example:

div {

margin: 0 auto;

}

In the example above, margin: 0 auto; will center the divs in their containing elements. The 0 sets the top and bottom margins to 0 pixels. The auto value instructs the browser to adjust the left and right margins until the element is centered within its containing element.

The div elements in the example above should center within an element that fills the page, but this doesn't occur. Why?

In order to center an element, a width must be set for that element. Otherwise, the width of the div will be automatically set to the full width of its containing element, like the <body>, for example. It's not possible to center an element that takes up the full width of the page.

div.headline {

width: 400px;

margin: 0 auto;

}

In the example above, the width of the div is set to 400 pixels, which is less than the width of most screens. This will cause the div to center within a containing element that is greater than 400 pixels wide.

**1.**

Set the width of the .pull-quote class elements to 350 pixels.

**2.**

In one line, set the vertical margins of the .pull-quote class to 0 and the horizontal margins to auto.

**3.**

Set the vertical margins of the #main element to 0, and the horizontal margins to auto.

# Margin Collapse

As you have seen, padding is space added inside an element's border, while margin is space added outside an element's border. One additional difference is that top and bottom margins, also called vertical margins, collapse, while top and bottom padding does not.

Horizontal margins (left and right), like padding, are always displayed and added together. For example, if two divs with ids #div-one and #div-two, are next to each other, they will be as far apart as the sum of their adjacent margins.

#img-one {

margin-right: 20px;

}

#img-two {

margin-left: 20px;

}

In this example, the space between the #img-one and #img-two borders is 40 pixels. The right margin of #img-one (20px) and the left margin of #img-two (20px) add to make a total margin of 40 pixels.

Unlike horizontal margins, vertical margins do not add. Instead, the larger of the two vertical margins sets the distance between adjacent elements.

#img-one {

margin-bottom: 30px;

}

#img-two {

margin-top: 20px;

}

In this example, the vertical margin between the #img-one and #img-two elements is 30 pixels. Although the sum of the margins is 50 pixels, the margin collapses so the spacing is only dependent on the #img-one bottom margin.

It may be helpful to think of collapsing vertical margins as a short person trying to push a taller person. The tall person has longer arms and can easily push the short person, while the person with short arms cannot reach the person with long arms.

Instructions

Study the graphic display to the right. Elements A and B have 20 pixels of horizontal margin, the sum of each element's margin. Elements A and C have 30 pixels of vertical margin — the top margin of element C.

**Minimum and Maximum Height and Width**

Because a web page can be viewed through displays of differing screen size, the content on the web page can suffer from those changes in size. To avoid this problem, CSS offers two properties that can limit how narrow or how wide an element's box can be sized to.

1. min-width — this property ensures a minimum width of an element's box.
2. max-width — this property ensures a maximum width of an element's box.

p {

min-width: 300px;

max-width: 600px;

}

In the example above, the width of all paragraphs will not shrink below 300 pixels, nor will the width exceed 600 pixels.

Content, like text, can become difficult to read when a browser window is narrowed or expanded. These two properties ensure that content is legible by limiting the minimum and maximum widths of an element.

You can also limit the minimum and maximum *height* of an element.

1. min-height — this property ensures a minimum height for an element's box.
2. max-height — this property ensures a maximum height of an element's box.

p {

min-height: 150px;

max-height: 300px;

}

In the example above, the height of all paragraphs will not shrink below 150 pixels and the height will not exceed 300 pixels.

What will happen to the contents of an element's box if the max-height property is set too low? It's possible for the content to spill outside of the box, resulting in content that is not legible. You'll learn how to work around this issue in the next exercise.

**1.**

In **style.css**, set the minimum width of the paragraph to 200 pixels.

After you've done this successfully, resize the browser and notice how the paragraph's box will no longer shrink below 200 pixels.

**2.**

Next, set the maximum width of the paragraph to 800 pixels.

After you've done this successfully, resize the browser and notice how the paragraph's box will no longer expand beyond 800 pixels.

**3.**

In **style.css**, set the minimum height of the paragraph to 200 pixels.

After you've done this successfully, resize the browser and notice how the height of paragraph's box will no longer shrink below 200 pixels.

**4.**

In **style.css**, set the maximum height of the paragraph to 300 pixels.

After you've done this successfully, resize the browser and notice how the height of paragraph's box will no longer expand beyond 300 pixels. You should see your text overflowing. In the next exercise, we will fix that!

**Overflow**

All of the components of the box model comprise an element’s size. For example, an image that has the following dimensions is 364 pixels wide and 244 pixels tall.

* 300 pixels wide
* 200 pixels tall
* 10 pixels padding on the left and right
* 10 pixels padding on the top and bottom
* 2 pixels border on the left and right
* 2 pixels border on the top and bottom
* 20 pixels margin on the left and right
* 10 pixels margin on the top and bottom

The total dimensions (364px by 244px) are calculated by adding all of the vertical dimensions together and all of the horizontal dimensions together. Sometimes, these components result in an element that is larger than the parent's containing area.

How can we ensure that we can view all of an element that is larger than its parent's containing area?

The overflow property controls what happens to content that spills, or overflows, outside its box. It can be set to one of the following values:

* hidden - when set to this value, any content that overflows will be hidden from view.
* scroll - when set to this value, a scrollbar will be added to the element's box so that the rest of the content can be viewed by scrolling.
* visible - when set to this value, the overflow content will be displayed outside of the containing element. Note, this is the default value.

p {

overflow: scroll;

}

In the example above, if any of the paragraph content overflows (perhaps a user resizes their browser window), a scrollbar will appear so that users can view the rest of the content.

The overflow property is set on a parent element to instruct a web browser how to render child elements. For example, if a div’s overflow property is set to scroll, all children of this div will display overflowing content with a scroll bar.

**1.**

In order to see the impact of overflow: scroll, first change the height of the #main element to 1000 pixels.

**2.**

Set the overflow of the #main element to scroll.

When you scroll down, a second scroll bar should appear over the paragraph section.

# Resetting Defaults

All major web browsers have a default stylesheet they use in the absence of an external stylesheet. These default stylesheets are known as user agent stylesheets. In this case, the term "[user agent](https://en.wikipedia.org/wiki/User_agent)" is a technical term for the browser.

User agent stylesheets often have default CSS rules that set default values for padding and margin. This affects how the browser displays HTML elements, which can make it difficult for a developer to design or style a web page.

Many developers choose to reset these default values so that they can truly work with a clean slate.

\* {

margin: 0;

padding: 0;

}

The code in the example above resets the default margin and padding values of all HTML elements. It is often the first CSS rule in an external stylesheet.

Note that both properties are both set to 0. When these properties are set to 0, they do not require a unit of measurement.

Instructions

**1.**

In **style.css**, reset the default margin and padding values for the body. What happens to the web page in the browser?

**Visibility**

Elements can be hidden from view with the visibility property.

The visibility property can be set to one of the following values:

1. hidden — hides an element.
2. visible — displays an element.

<ul>

<li>Explore</li>

<li>Connect</li>

<li class="future">Donate</li>

<ul>

.future {

visibility: hidden;

}

In the example above, the list item with a class of future will be hidden from view in the browser.

Keep in mind, however, that users can still view the contents of the list item (e.g., Donate) by viewing the source code in their browser. Furthermore, the web page will *only* hide the contents of the element. It will still leave an empty space where the element is intended to display.

**Note:** What's the difference between display: none and visibility: hidden? An element with display: none will be completely removed from the web page. An element with visibility: hidden, however, will not be visible on the web page, but the space reserved for it will.

Instructions

**1.**

Take a look at the list items in **index.html**. Notice that the list item Donate has a class of donate.

In **style.css**:

1. Add a class selector for donate
2. Set the visibility to hidden

**Review**

In this lesson, we covered the four properties of the box model: height and width, padding, borders, and margins. Understanding the box model is an important step towards learning more advanced HTML and CSS topics. Let's take a minute to review what you learned.

1. The box model comprises a set of properties used to create space around and between HTML elements.
2. The height and width of a content area can be set in pixels or percentage.
3. Borders surround the content area and padding of an element. The color, style, and thickness of a border can be set with CSS properties.
4. Padding is the space between the content area and the border. It can be set in pixels or percent.
5. Margin is the amount of spacing outside of an element's border.
6. Horizontal margins add, so the total space between the borders of adjacent elements is equal to the sum of the right margin of one element and the left margin of the adjacent element.
7. Vertical margins collapse, so the space between vertically adjacent elements is equal to the larger margin.
8. margin: 0 auto horizontally centers an element inside of its parent content area, if it has a width.
9. The overflow property can be set to display, hide, or scroll, and dictates how HTML will render content that overflows its parent's content area.
10. The visibility property can hide or show elements.

Make some adjustments to the code in the code editor. See if you can improve the appearance of the page by changing the following properties:

1. width
2. height
3. padding
4. border
5. margin
6. overflow

# Why Change the Box Model?

The last lesson focused on the most important aspects of the box model: box dimensions, borders, padding, and margin.

The box model, however, has an awkward limitation regarding box dimensions. This limitation is best illustrated with an example.

<h1>Hello World</h1>

h1 {

border: 1px solid black;

height: 200px;

width: 300px;

padding: 10px;

}

In the example above, a heading element's box has solid, black, 1 pixel thick borders. The height of the box is 200 pixels, while the width of the box is 300 pixels. A padding of 10 pixels has also been set on all four sides of the box's content.

Unfortunately, under the current box model, the border thickness and the padding will affect the dimensions of the box.

The 10 pixels of padding increases the height of the box to 220 pixels and the width to 320 pixels. Next, the 1-pixel thick border increases the height to 222 pixels and the width to 322 pixels.

Under this box model, the border thickness and padding are added to the overall dimensions of the box. This makes it difficult to accurately size a box. Over time, this can also make all of a web page's content difficult to position and manage.

In this brief lesson, you'll learn how to use a different technique that avoids this problem altogether.

# Box Model: Content-Box

Many properties in CSS have a default value and don't have to be explicitly set in the stylesheet.

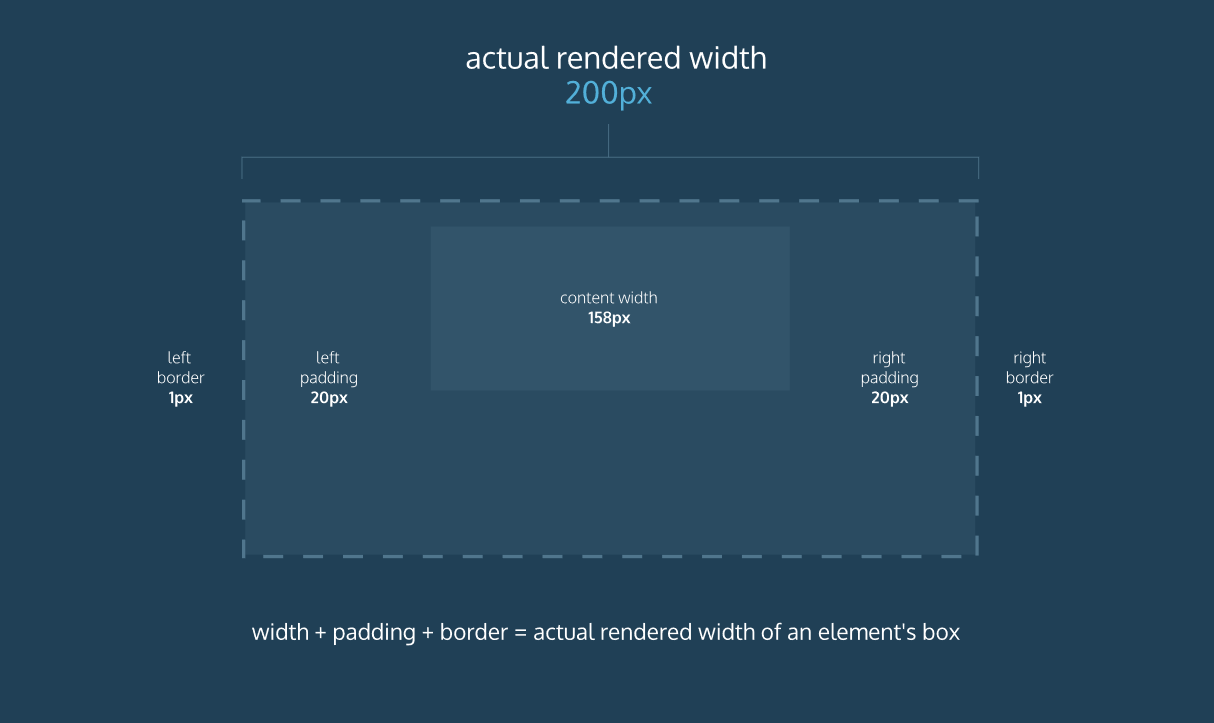
For example, the default font-weight of text is normal, but this property-value pair is not typically specified in a stylesheet.

The same can be said about the box model that browsers assume. In CSS, the box-sizing property controls the type of box model the browser should use when interpreting a web page.

The default value of this property is content-box. This is the same box model that is affected by border thickness and padding.

Instructions

Study the diagram to the right. It illustrates the default box model used by the browser, content-box. When you're done, continue to the next exercise.



**Box Model: Border-Box**

Fortunately, we can reset the entire box model and specify a new one: border-box.

\* {

box-sizing: border-box;

}

The code in the example above resets the box model to border-box for all HTML elements. This new box model avoids the dimensional issues that exist in the former box model you learned about.

In this box model, the height and width of the box will remain fixed. The border thickness and padding will be included inside of the box, which means the overall dimensions of the box do not change.

<h1>Hello World</h1>

\* {

box-sizing: border-box;

}

h1 {

border: 1px solid black;

height: 200px;

width: 300px;

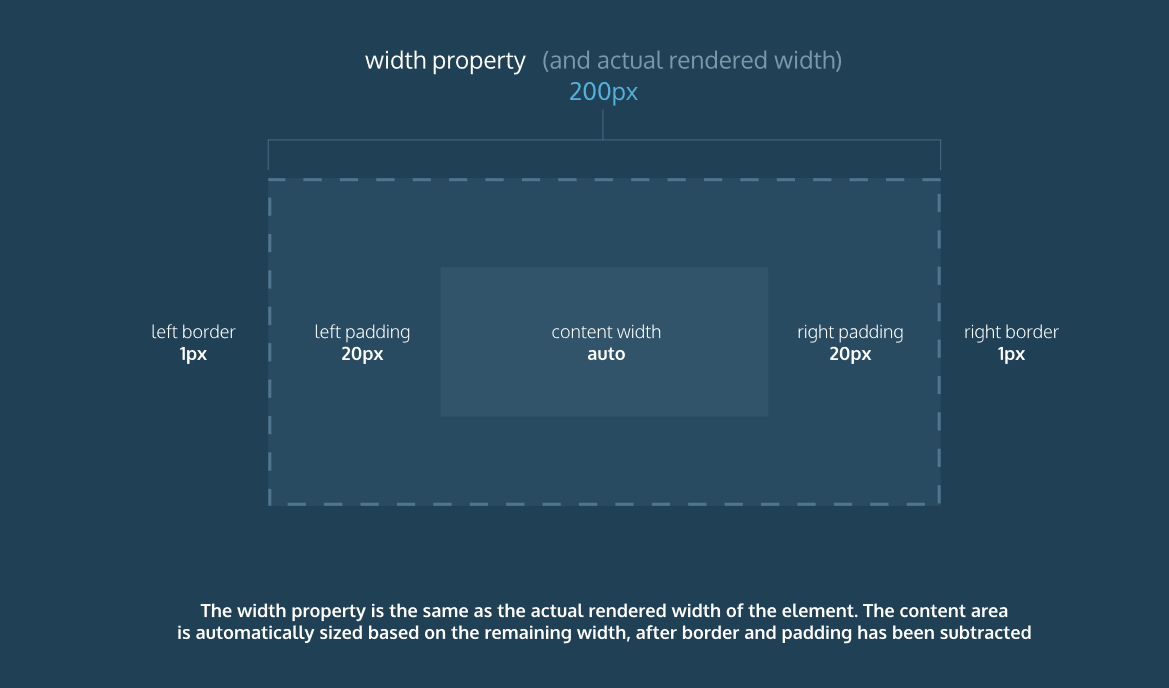
padding: 10px;

}

In the example above, the height of the box would remain at 200 pixels and the width would remain at 300 pixels. The border thickness and padding would remain entirely *inside* of the box.

Instructions

Study the diagram to the right. It illustrates the new box model, border-box.



**The New Box Model**

Now that you know about the new box model, let's actually implement it in the browser.

\* {

box-sizing: border-box;

}

It's that simple! In the example above, the universal selector (\*) targets all elements on the web page and sets their box model to the border-box model.

**1.**

In **style.css**, change the box model for all elements on the web page to the new box model.

You probably didn't see a difference in the web page to the right - that's ok! The new box model simply makes sure that the dimensions of elements remains the same regardless of border width and padding.

# Review: Changing the Box Model

In this lesson, you learned about an important limitation of the default box model: box dimensions are affected by border thickness and padding.

Let's review what you learned:

1. In the default box model, box dimensions are affected by border thickness and padding.
2. The box-sizing property controls the box model used by the browser.
3. The default value of the box-sizing property is content-box.
4. The value for the new box model is border-box.
5. The border-box model is not affected by border thickness or padding.

**Flow of HTML**

A browser will render the elements of an HTML document that has no CSS from left to right, top to bottom, in the same order as they exist in the document. This is called the *flow* of elements in HTML.

In addition to the properties that it provides to style HTML elements, CSS includes properties that change how a browser *positions* elements. These properties specify where an element is located on a page, if the element can share lines with other elements, and other related attributes.

In this lesson, you will learn five properties for adjusting the position of HTML elements in the browser:

* position
* display
* z-index
* float
* clear

Each of these properties will allow us to position and view elements on a web page. They can be used in conjunction with any other styling properties you may know.

Index.html

<!DOCTYPE html>

<html>

<head>

<title>Please Participate in Our Survey!</title>

<link href="https://fonts.googleapis.com/css?family=Oswald:300,700|Varela+Round" rel="stylesheet">

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<header>

<ul>

<li>Question 1</li>

<li>Question 2</li>

<li>Question 3</li>

<li>Question 4</li>

<li>Question 5</li>

<li>Question 6</li>

</ul>

</header>

<div class="welcome">

<h1>Welcome to our survey!</h1>

<p>We're looking forward to getting your answers so we can make sure our products and services are the best they can be!</p>

</div>

<div class="question">

<h4>Question 1</h4>

<h2>I like participating in physical activity such as running, swimming, or biking.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

<div class="question">

<h4>Question 2</h4>

<h2>I try to keep up to date with the latest fashion in active wear.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

<div class="question">

<h4>Question 3</h4>

<h2>I purchase clothing online regularly.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

<div class="question">

<h4>Question 4</h4>

<h2>I try to buy goods that are designed and/or manufactured in my home country.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

<div class="question">

<h4>Question 5</h4>

<h2>I look to famous athletes when trying to choose what to wear when training.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

</body>

</html>

Style.css

body {

background-color: #FFF;

margin: 0 auto;

}

header {

background-color: #466995;

border-bottom: 1px solid #466995;

}

ul {

margin: 30px auto;

padding: 0 20px;

text-align: center;

}

li {

color: #FFF;

font-family: 'Oswald', sans-serif;

font-size: 16px;

font-weight: 300;

text-transform: uppercase;

}

li:hover {

color: #DBE9EE;

}

h1 {

color: #466995;

font-family: 'Oswald', sans-serif;

font-size: 32px;

font-weight: 300;

text-transform: uppercase;

}

h2 {

color: #333;

font-family: 'Varela Round', sans-serif;

font-size: 26px;

font-weight: 100;

margin: 0 auto 20px auto;

}

h3 {

color: #466995;

font-family: 'Oswald', sans-serif;

font-size: 18px;

text-align: center;

font-weight: 700;

text-transform: uppercase;

padding: 30px;

}

h4 {

color: #466995;

font-family: 'Oswald', sans-serif;

font-size: 18px;

font-weight: 300;

letter-spacing: 2px;

text-align: center;

text-transform: uppercase

}

p {

color: #333;

font-family: 'Varela Round', sans-serif;

font-size: 18px;

}

footer {

background-color: #DBE9EE;

text-align: center;

}

.welcome {

background-color: #DBE9EE;

box-sizing: border-box;

padding: 40px;

text-align: center;

width: 100%;

}

.question {

text-align: center;

}

.answer {

border: 1px solid #466995;

margin: 20px;

}

.answer:hover {

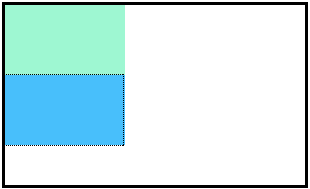
background: #C0D6DF;

color: #FFF;

}

**Position**

Take a look at the block-level elements in the image below:



The boxes in the image above were created with the following CSS:

.boxes {

width: 120px;

height: 70px;

}

and the following HTML:

<div class="boxes"></div>

<div class="boxes"></div>

Notice the block-level elements in the image above take up their own line of space and therefore don't overlap each other. In the browser to the right you can see block-level elements also consistently appear on the left side of the browser. This is the default *position* for block-level elements.

The default position of an element can be changed by setting its position property. The position property can take one of four values:

1. static - the default value (it does not need to be specified)
2. relative
3. absolute
4. fixed

In the next few exercises, you'll learn about the values in items 2, 3, and 4 above. For now, it's important to understand that if you favor the default position of an HTML element, you don't need to set its position property.

Instructions

**1.**

In **style.css**, set the position in .question to static.

Notice that setting position to static does nothing. That's because static simply refers to the default behavior.

Index.html

<!DOCTYPE html>

<html>

<head>

<title>Please Participate in Our Survey!</title>

<link href="https://fonts.googleapis.com/css?family=Oswald:300,700|Varela+Round" rel="stylesheet">

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<header>

<ul>

<li>Question 1</li>

<li>Question 2</li>

<li>Question 3</li>

<li>Question 4</li>

<li>Question 5</li>

<li>Question 6</li>

</ul>

</header>

<div class="welcome">

<h1>Welcome to our survey!</h1>

<p>We're looking forward to getting your answers so we can make sure our products and services are the best they can be!</p>

</div>

<div class="question">

<h4>Question 1</h4>

<h2>I like participating in physical activity such as running, swimming, or biking.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

<div class="question">

<h4>Question 2</h4>

<h2>I try to keep up to date with the latest fashion in active wear.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

<div class="question">

<h4>Question 3</h4>

<h2>I purchase clothing online regularly.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

<div class="question">

<h4>Question 4</h4>

<h2>I try to buy goods that are designed and/or manufactured in my home country.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

<div class="question">

<h4>Question 5</h4>

<h2>I look to famous athletes when trying to choose what to wear when training.</h2>

<div class="answer">

<h3>Disagree</h3>

</div>

<div class="answer">

<h3>Neutral</h3>

</div>

<div class="answer">

<h3>Agree</h3>

</div>

</div>

</body>

</html>

Style.css

body {

background-color: #FFF;

margin: 0 auto;

}

header {

background-color: #466995;

border-bottom: 1px solid #466995;

}

ul {

margin: 30px auto;

padding: 0 20px;

text-align: center;

}

li {

color: #FFF;

font-family: 'Oswald', sans-serif;

font-size: 16px;

font-weight: 300;

text-transform: uppercase;

}

li:hover {

color: #DBE9EE;

}

h1 {

color: #466995;

font-family: 'Oswald', sans-serif;

font-size: 32px;

font-weight: 300;

text-transform: uppercase;

}

h2 {

color: #333;

font-family: 'Varela Round', sans-serif;

font-size: 26px;

font-weight: 100;

margin: 0 auto 20px auto;

}

h3 {

color: #466995;

font-family: 'Oswald', sans-serif;

font-size: 18px;

text-align: center;

font-weight: 700;

text-transform: uppercase;

padding: 30px;

}

h4 {

color: #466995;

font-family: 'Oswald', sans-serif;

font-size: 18px;

font-weight: 300;

letter-spacing: 2px;

text-align: center;

text-transform: uppercase

}

p {

color: #333;

font-family: 'Varela Round', sans-serif;

font-size: 18px;

}

footer {

background-color: #DBE9EE;

text-align: center;

}

.welcome {

background-color: #DBE9EE;

box-sizing: border-box;

padding: 40px;

text-align: center;

width: 100%;

}

.question {

text-align: center;

}

.answer {

border: 1px solid #466995;

margin: 20px;

}

.answer:hover {

background: #C0D6DF;

color: #FFF;

}

**Position: Relative**

One way to modify the default position of an element is by setting its position property to relative.

This value allows you to position an element *relative* to its default static position on the web page.

.box-bottom {

background-color: DeepSkyBlue;

position: relative;

}

Although the code in the example above instructs the browser to expect a relative positioning of the div, it does not specify where the div should be positioned on the page.

.box-bottom {

background-color: DeepSkyBlue;

position: relative;

top: 20px;

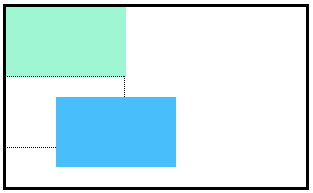
left: 50px;

}

In the example above, the <div> has been positioned using two of the four *offset properties*. The valid offset properties are:

1. top - moves the element down.
2. bottom - moves the element up.
3. left - moves the element right.
4. right - moves the element left.

In the example above, the <div> will be moved down 20 pixels and to the right 50 pixels from its default static position. The image below displays the new position of the box. The dotted line represents where the statically positioned (default) box was positioned.



Units for offset properties can be specified in pixels, ems, or percentages. Note that offset properties will not work if the value of the element's position property is the default static.

Instructions

**1.**

In **style.css**, set the position in .question to relative.

**2.**

Next, offset .question 40 pixels from the top (again, in **style.css**)

# Position: Absolute

Another way of modifying the position of an element is by setting its position to absolute.

When an element's position is set to absolute all other elements on the page will ignore the element and act like it is not present on the page. The element will be positioned relative to its closest positioned parent element.

.box-bottom {

background-color: DeepSkyBlue;

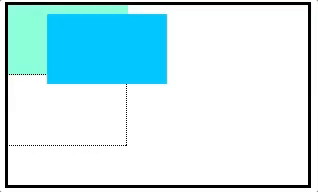
position: absolute;

top: 20px;

left: 50px;

}

In the example above, the .box-bottom <div> will be moved down and right from the top left corner of the view. If offset properties weren't specified, the top box would be entirely covered by the bottom box. Take a look at the gif below:



The bottom box in this image (colored blue) is displaced from the top left corner of its container. It is 20 pixels lower and 50 pixels to the right of the top box.

In the next exercise, we will compare the scrolling of absolute elements with fixed elements.

**1.**

In **style.css**, set the position inside of the header selector to absolute. Scroll up and down the web page. What do you notice?

**2.**

When you changed the position to absolute, you may have noticed that the header shrunk horizontally. We'll learn why in a later exercise. For now, set the width property of the header to 100%.

# Position: Fixed

When an element's position is set to absolute, as in the last exercise, the element will scroll with the rest of the document when a user scrolls.

We can fix an element to a specific position on the page (regardless of user scrolling) by setting its position to fixed.

.box-bottom {

background-color: DeepSkyBlue;

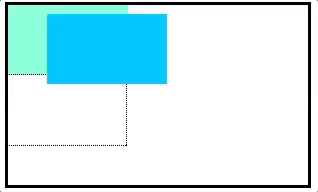
position: fixed;

top: 20px;

left: 50px;

}

In the example above, the .box-bottom <div> will remain fixed to its position no matter where the user scrolls on the page, like in the image below:



This technique is often used for navigation bars on a web page.

**1.**

In **style.css**, change the position inside of the header rule to fixed. Scroll up and down the web page. What do you notice?

**2.**

Notice that part of the "Welcome" section is now covered up by the header. That's because when we changed the position of the header to fixed, we removed it from the flow of the html document. Let's fix that. Change the position of the .welcome element to relative.

**3.**

Offset the "Welcome" section by 200 pixels from the top. Everything might not be displaying correctly just yet; we'll fix it in a later exercise.

# Z-Index

When boxes on a web page have a combination of different positions, the boxes (and therefore, their content) can overlap with each other, making the content difficult to read or consume.

.box-top {

background-color: Aquamarine;

}

.box-bottom {

background-color: DeepSkyBlue;

position: absolute;

top: 20px;

left: 50px;

}

In the example above, the .box-bottom <div> ignores the .box-top <div> and overlaps it as a user scrolls.

The z-index property controls how far "back" or how far "forward" an element should appear on the web page.

The z-index property accepts integer values. Depending on their values, the integers instruct the browser on the order in which elements should be displayed on the web page.

.box-top {

background-color: Aquamarine;

position: relative;

z-index: 2;

}

.box-bottom {

background-color: DeepSkyBlue;

position: absolute;

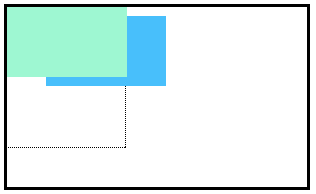
top: 20px;

left: 50px;

z-index: 1;

}

In the example above, we set the .box-top position to relative and the z-index to 2. We changed position to relative, because the z-index property does not work on static elements. The z-index of 2 moves the .box-top element forward, because it is greater than the .box-bottom z-index, 1. See the example image below:



In the image above, you can see the top box is moved in front of the bottom box.

Instructions

**1.**

In **style.css**, set the z-index of the header to 10. Notice how the header is no longer covered by other elements when you scroll!