In this course, we will learn the basics of **responsive web design** including **HTML** tags and **CSS** Rules. Using this knowledge, we will build the following **responsive company website** from scratch. In this lesson we setup our project environment.

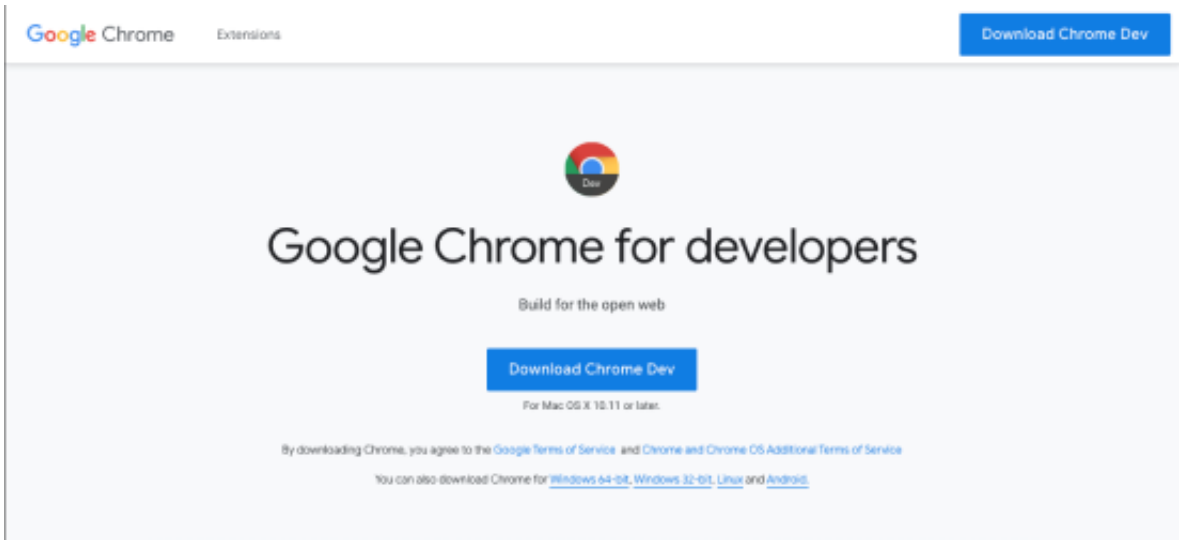
Development Environment

To begin, we need to setup our development environment. For that we require a modern **web browser** and **code editor**.

Web Browser

The first thing we need is a modern Web Browser and your best choice is one that has an integrated developer console and other tools for Web Development. Popular choices include Google Chrome and Mozilla Firefox.

Google Chrome for Developers (https://www.google.com/intl/en_ca/chrome/dev/)



Mozilla Firefox for Developers (https://www.mozilla.org/en-US/firefox/developer/)

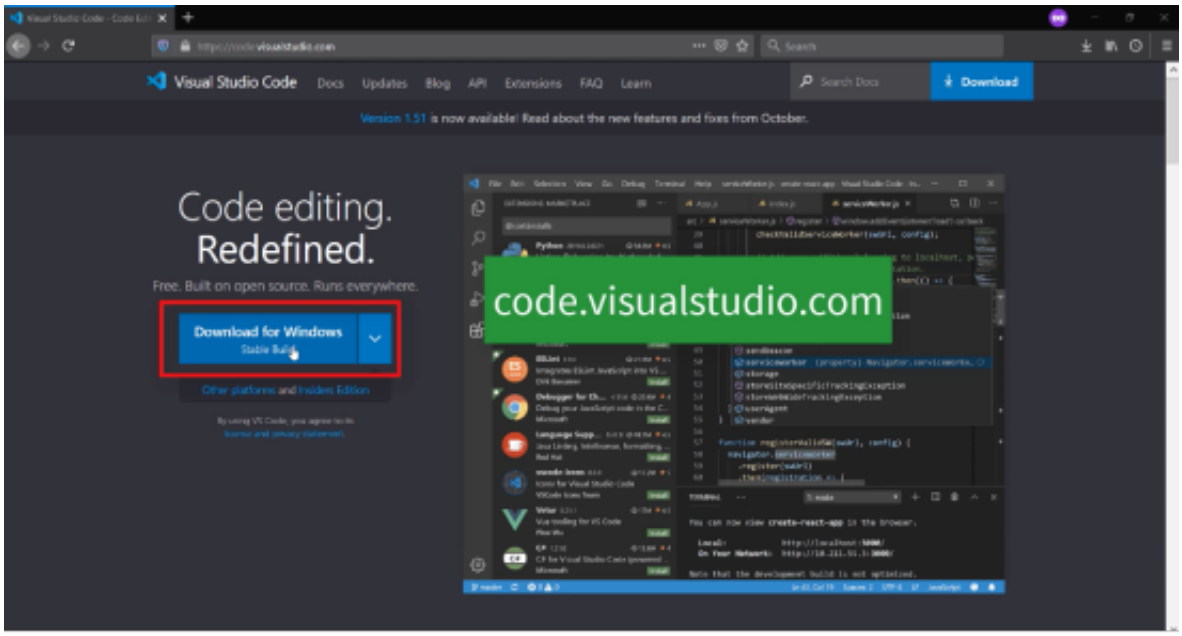
© 2021 Zenva Pty Ltd. All rights reserved.

Code Editor

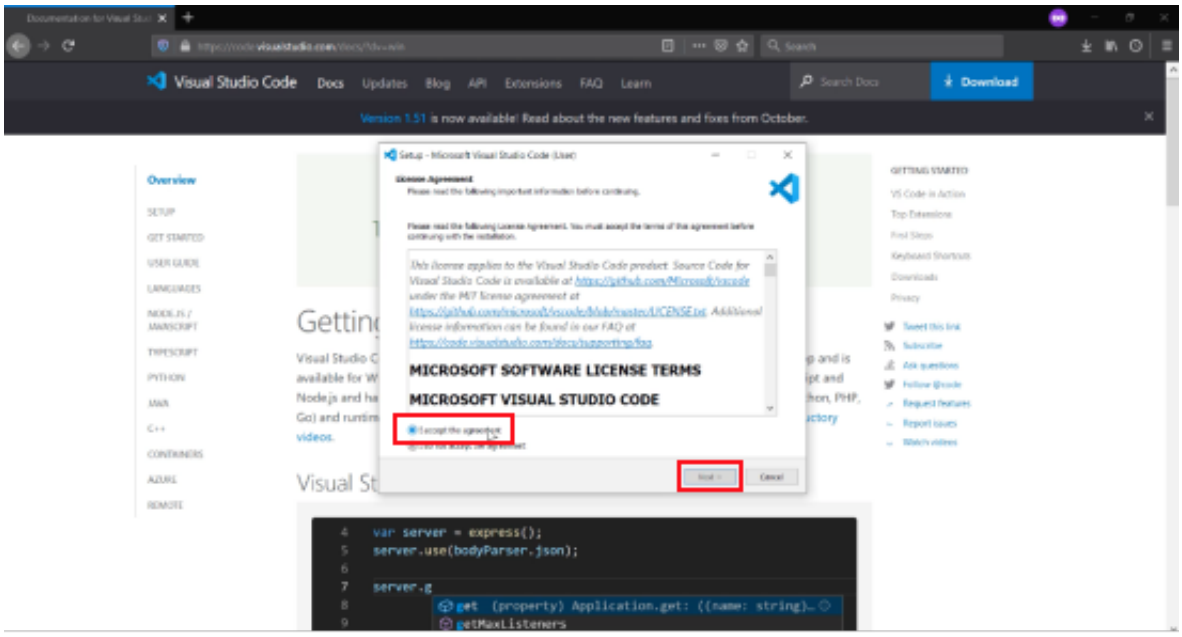
When it comes to **code editors** there are many good choices for Web Development (HTML, CSS, JavaScript). In this course we will be using **Visual Studio Code (VS Code)** which is free, widely used and available for all operating systems. For this course, you can use your favorite code editor or follow along with the relevant VS Code installation (Windows or Mac) below.

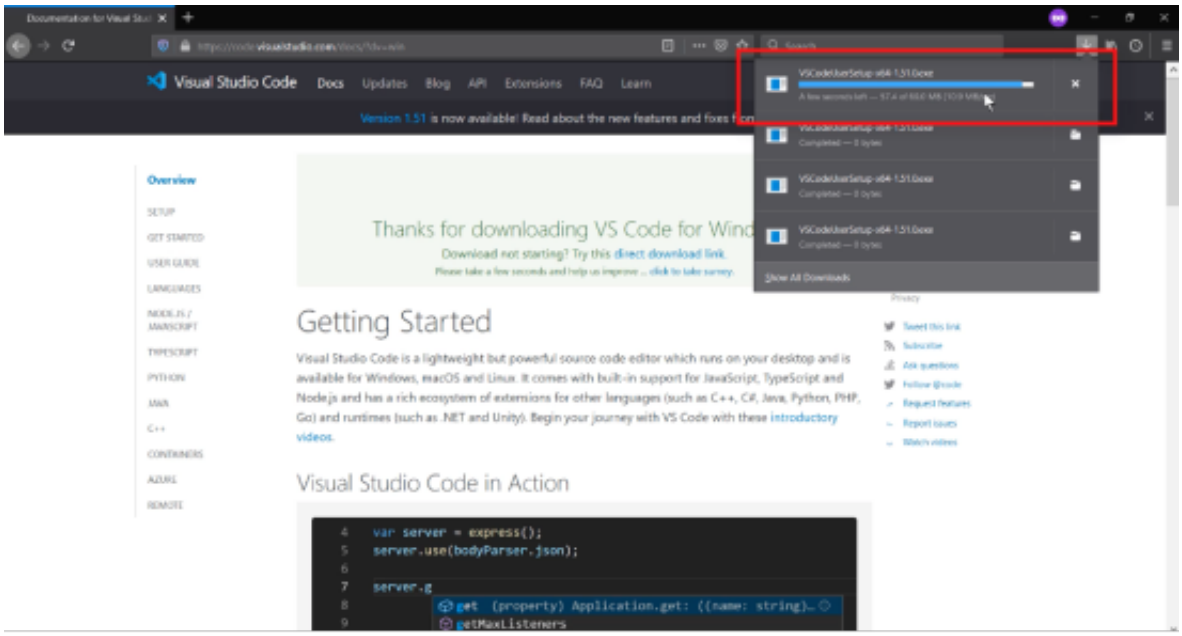
VS Code Installation (Windows)

To install VS Code on Windows, browse to the VS Code homepage (https://code.visualstudio.com/) and follow the installation instructions for your operating system. For Windows, click **Download for Windows** which will download the executable file to your computer. Once that completes, run the the installation **exe** and follow the installation prompts accepting or adjusting the default settings. When installation completes, you can run VS Code from the installation window or from the **Windows Start Menu**.

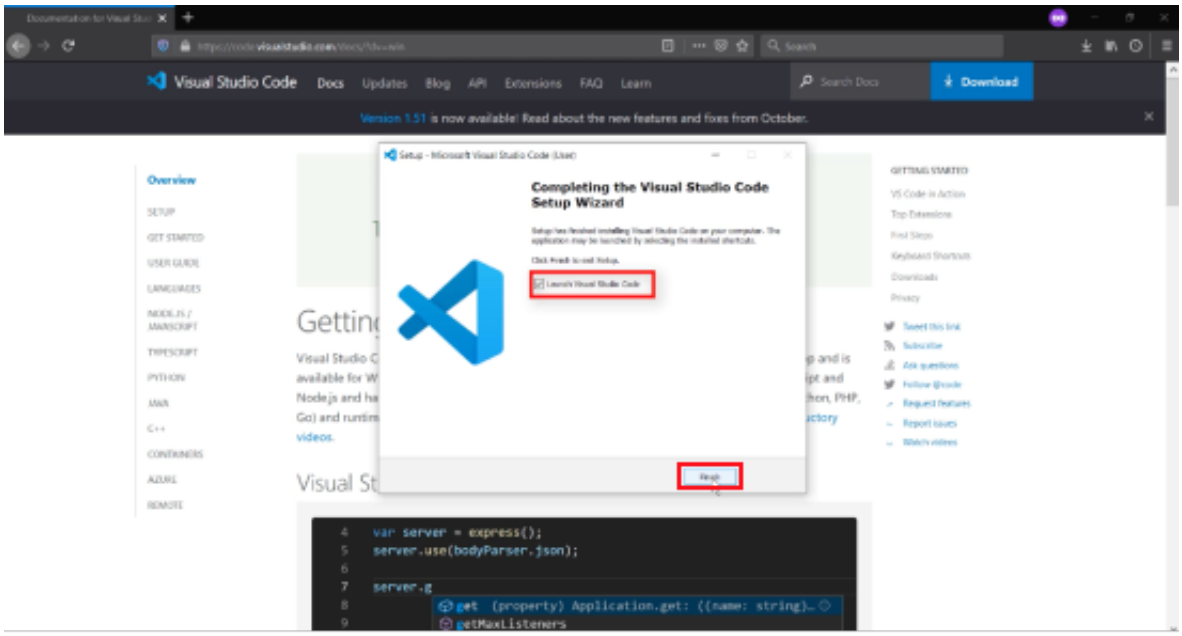


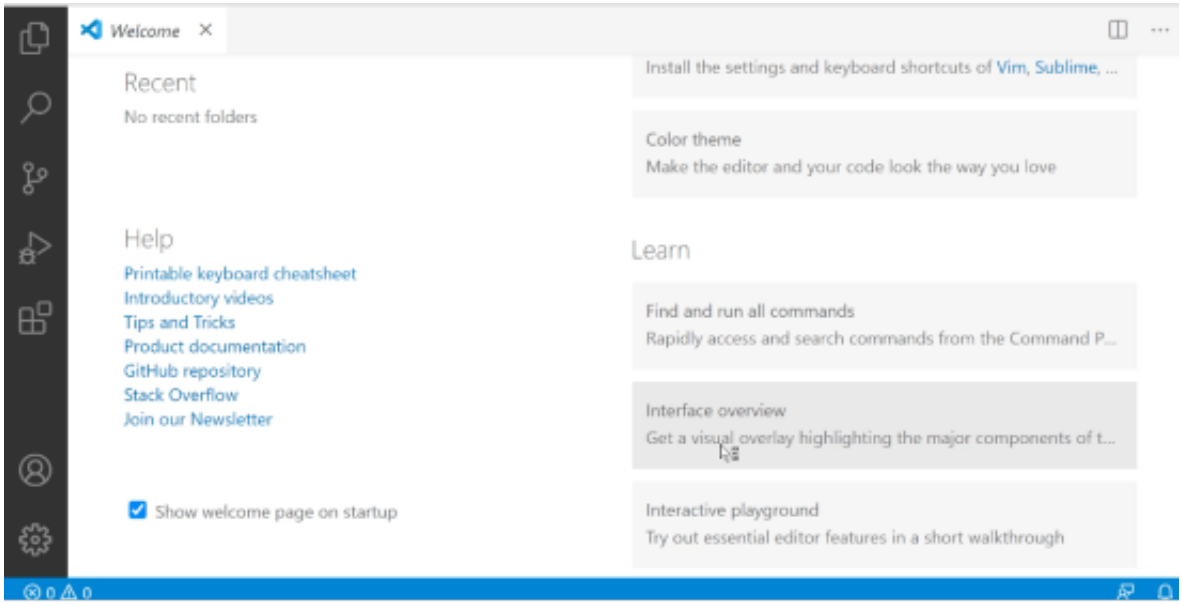
© 2021 Zenva Pty Ltd. All rights reserved.





© 2021 Zenva Pty Ltd. All rights reserved.

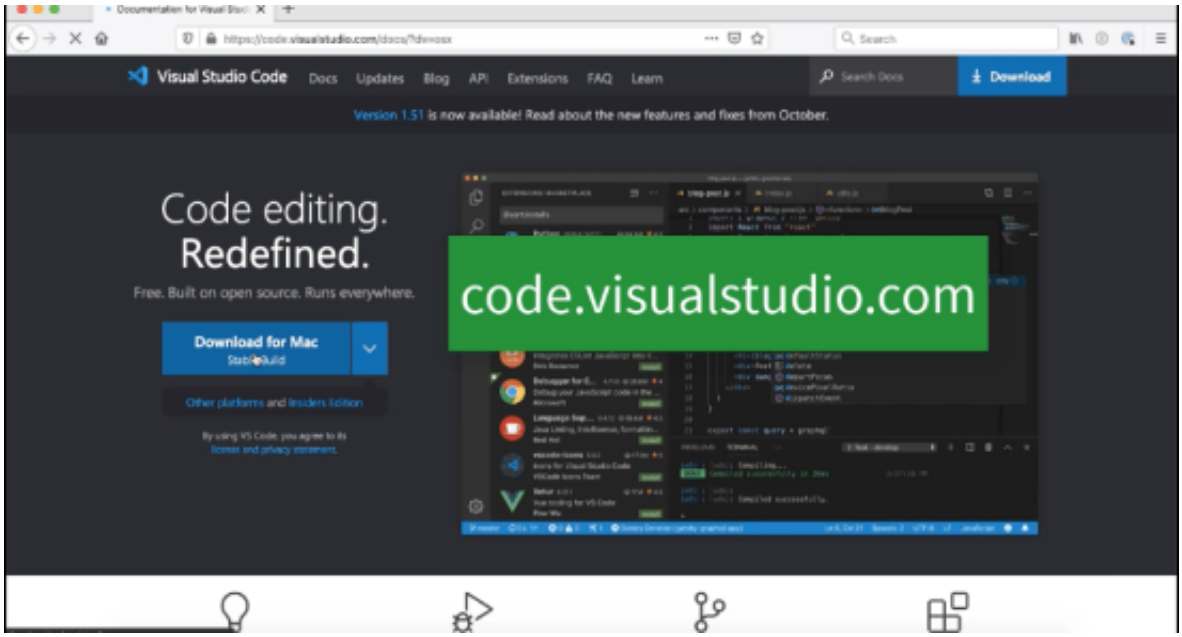


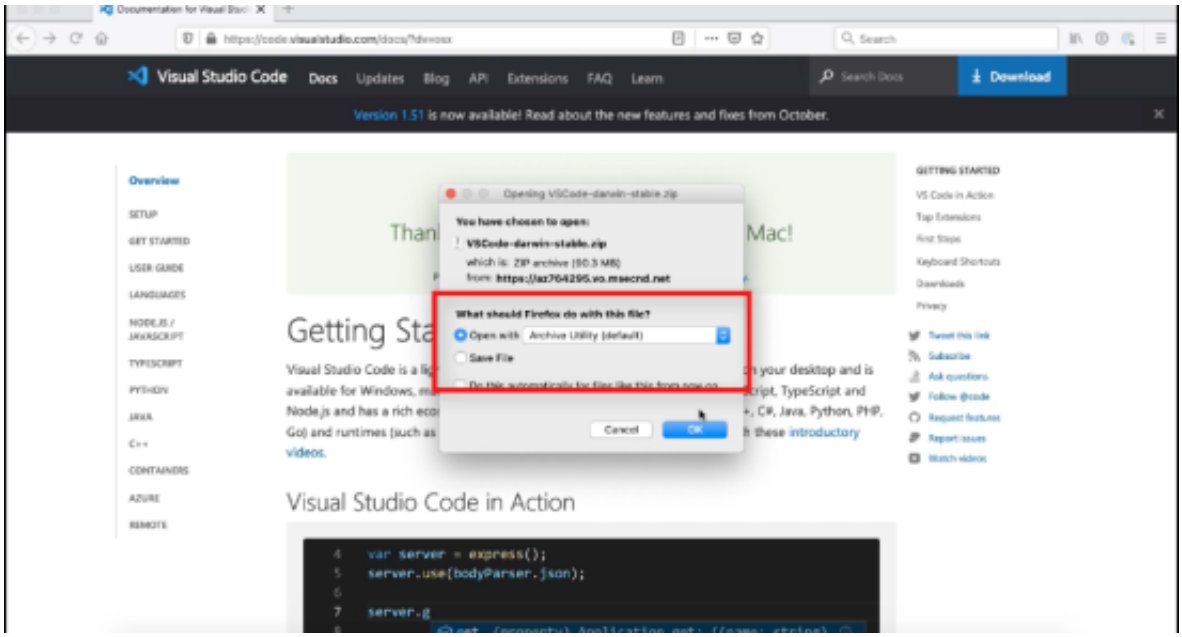


VS Code Installation (Mac)

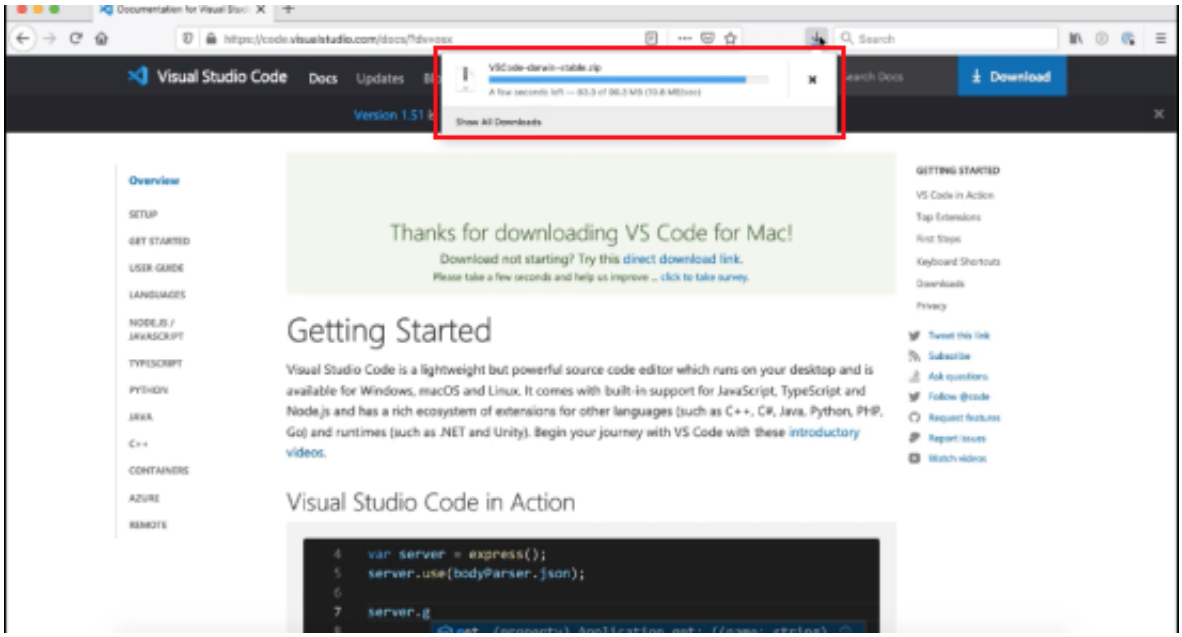
To install VS Code on Mac, browse to the VS Code homepage (https://code.visualstudio.com/) and follow the installation instructions for your operating system. For Mac, click **Download for Mac** and select the option to **Open with Archive Utility** to extract it. When the file finishes downloading you should see the extracted **VS Code** file in your **Downloads** folder. From there, drag the VS Code file to the **Applications** folder if you want it in your **task bar** otherwise just **double-click** to open code editor. You may be prompted with a security warning. If so, click **open**.

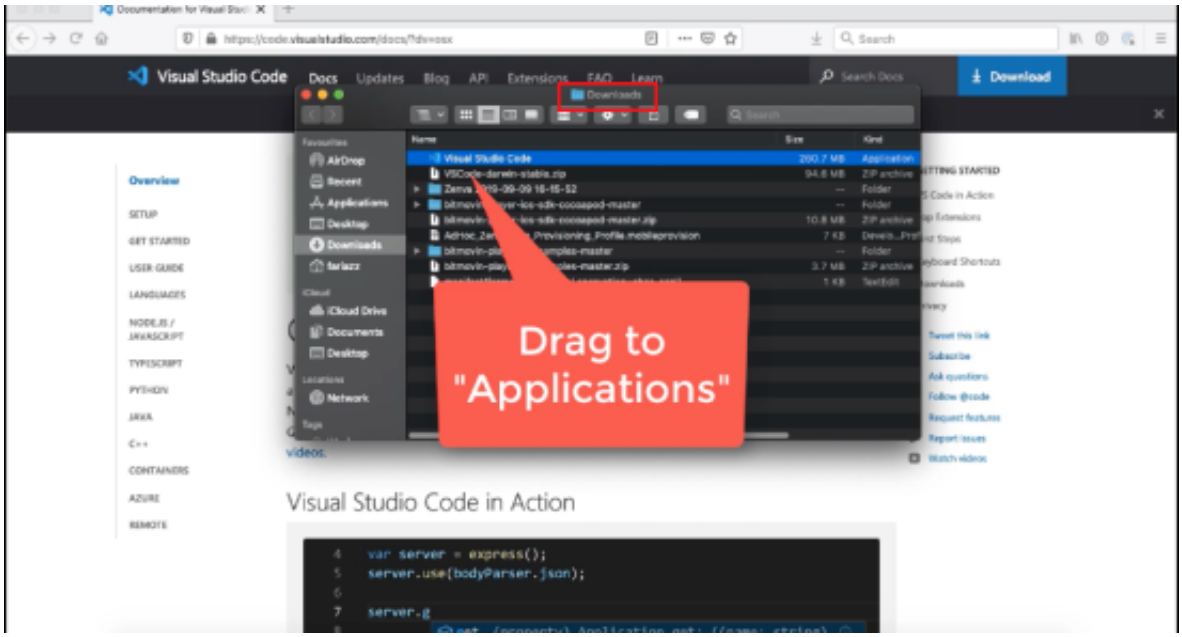
© 2021 Zenva Pty Ltd. All rights reserved.



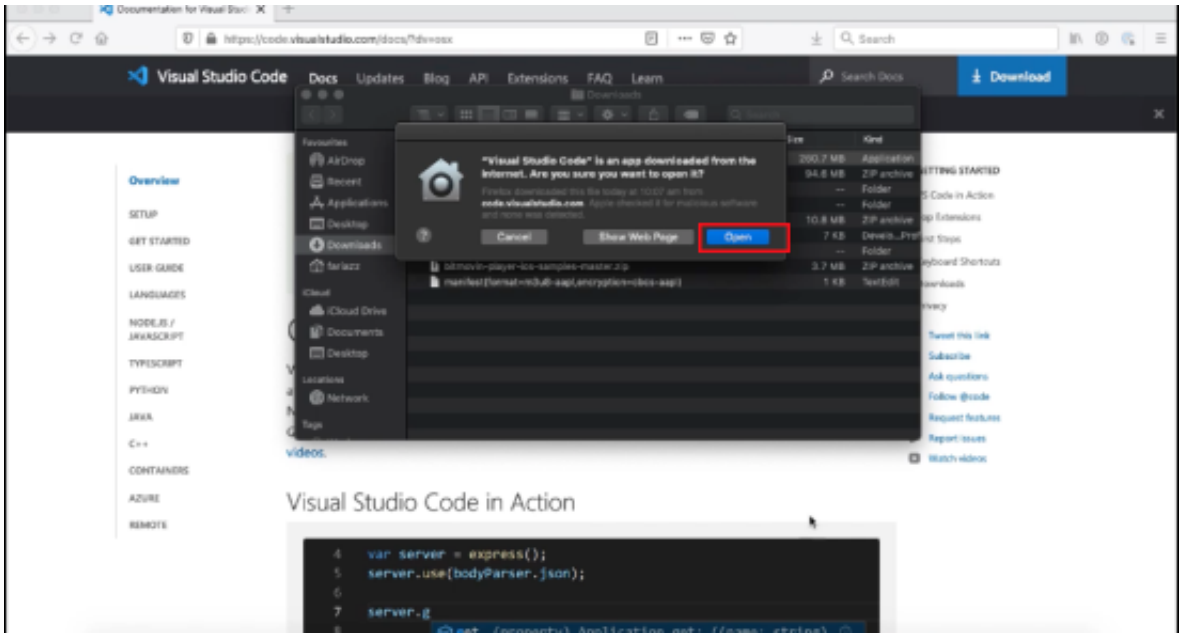


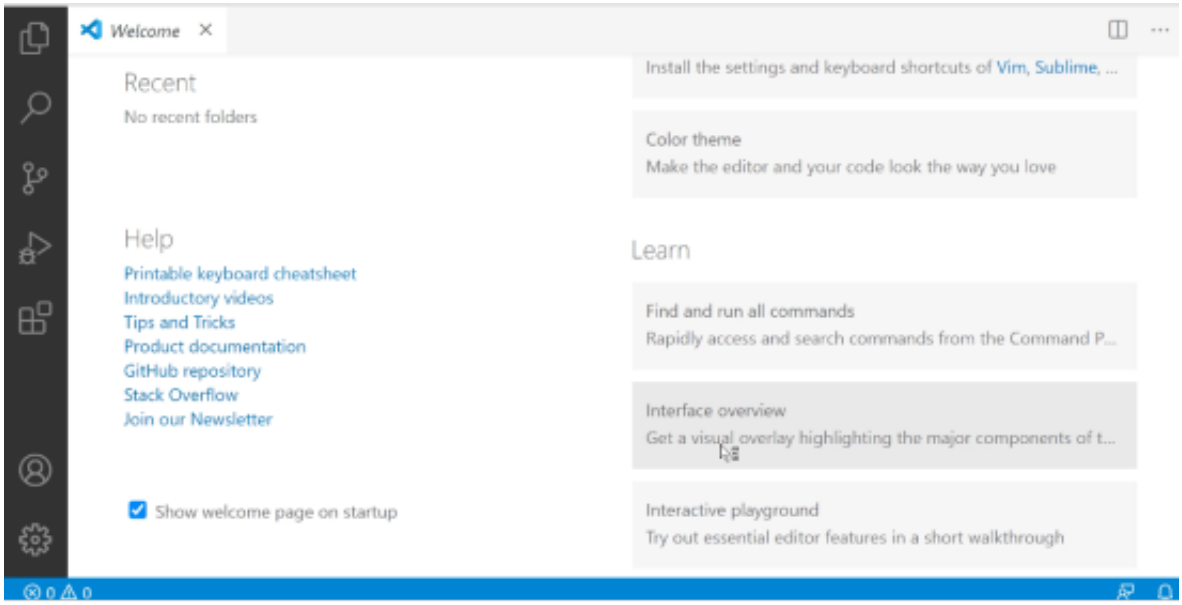
© 2021 Zenva Pty Ltd. All rights reserved.





© 2021 Zenva Pty Ltd. All rights reserved.

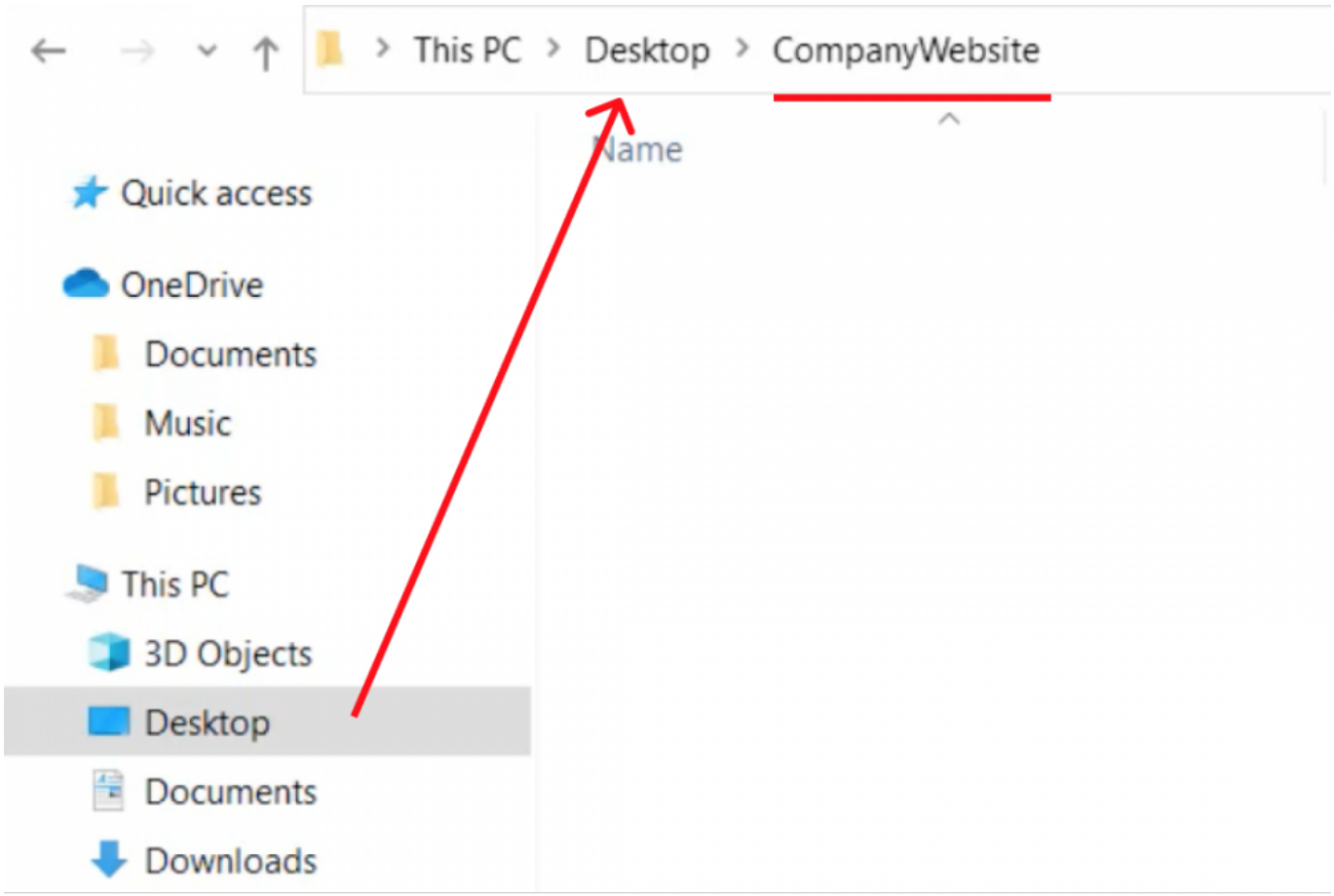




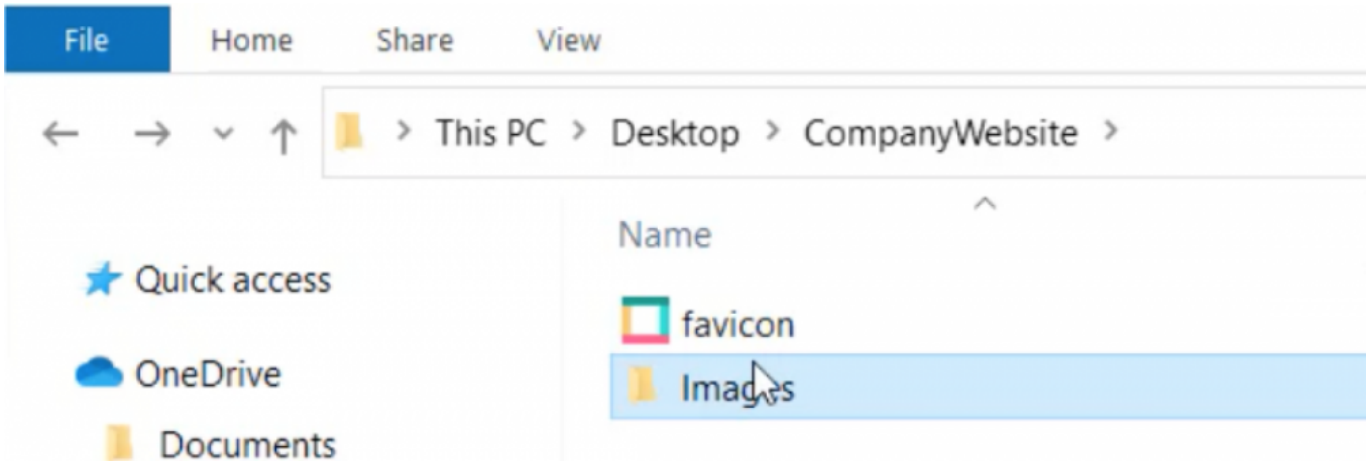
Project Setup

With your editor setup, choose a place to store your code files. We setup a Web Development project by setting up a main folder and within that we have all our web pages, code, styles and assets needed by our website. If you don't already have a place, you can simply create a folder in your Documents or on your Desktop to get started. In this example, we create a folder on our **Desktop** called **CompanyWebsite**.

© 2021 Zenva Pty Ltd. All rights reserved.

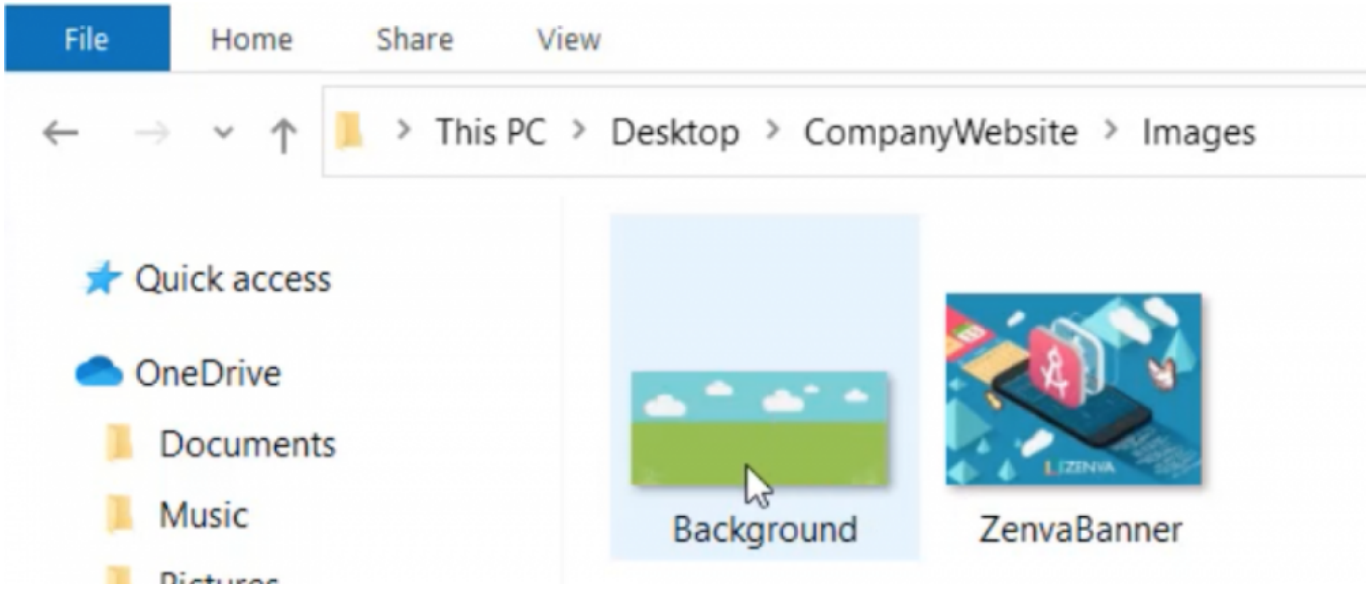


The next step is to get the **assets** we need from the **course downloads** (found under the Course Files tab). **Download** and **extract** this **zip** file into any folder (possibly a folder you setup for reference so you can check code along the way). Once extracted, we need to copy two things to the **CompanyWebsite** folder we just created. The first is an **icon** file named **favicon.ico** and the second is the **images** folder.

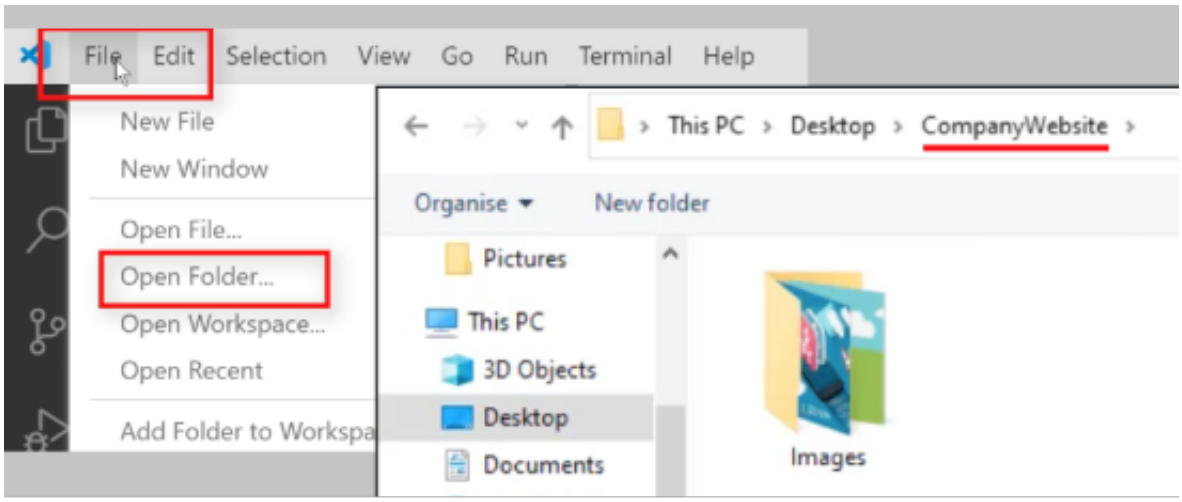


Inside the **images** folder, we have two images for our project (**background.png** and **ZenvaBanner.png**). You can uploaded any number of your own images by brining them into the project images folder.

© 2021 Zenva Pty Ltd. All rights reserved.



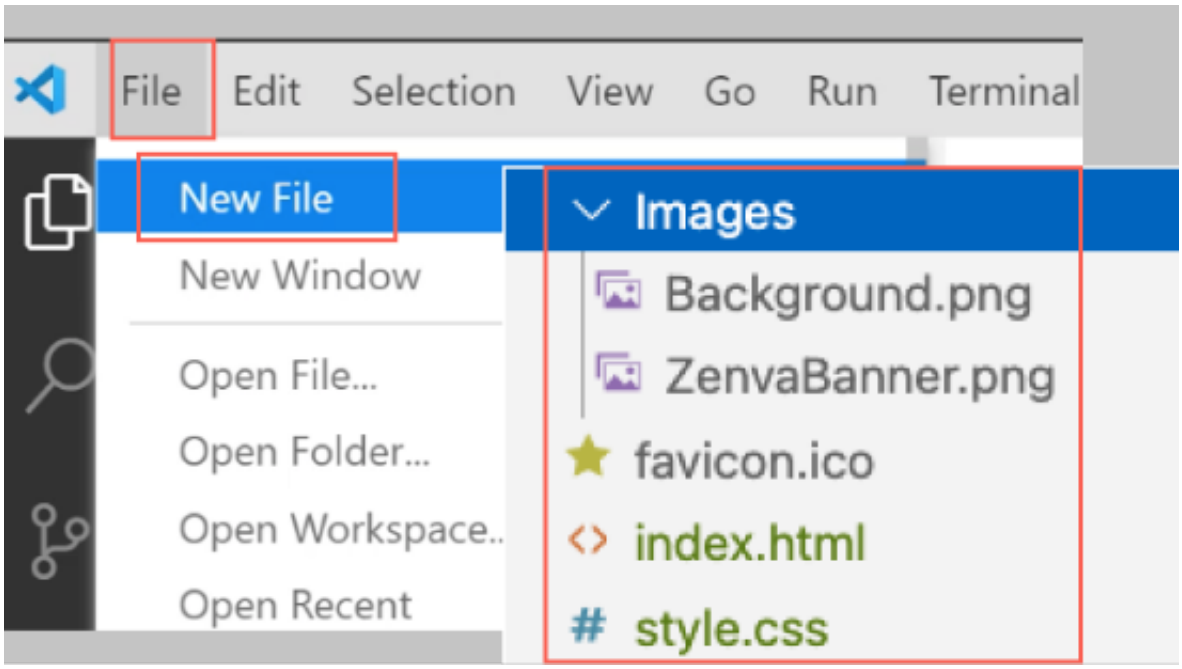
At this point we can open our project folder inside our code editor. **Open VS Code**, choose the **File** menu option and **Open Folder** menu item. Open the **project folder** by navigating to our project, **CompanyWebsite**, we created above.



Create Web Files

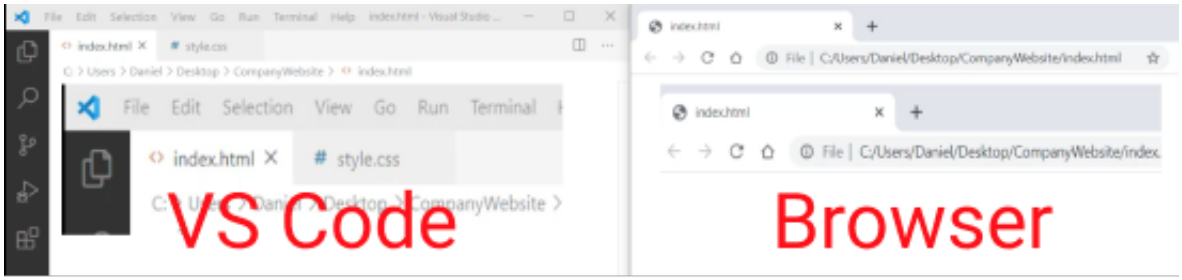
Inside VS Code, create two new files. The first file is our web page (**index.html**) and the second file is our stylesheet (**style.css**). Create the files by choosing the **File** menu option and **New File** menu item. It's important to have the dot HTML (.html) file extension for our web page and the dot CSS (.css) file extension for our stylesheet. Our project should now have the files and images folder shown on the right below.

© 2021 Zenva Pty Ltd. All rights reserved.



Code Editor and Browser Organization

There are many ways to open our website in a web **browser.** For simplicity, **right-click** on the **index.html** file in the **VS Code File Explorer** and choose **open in browser** from the **context menu**. With the code editor and the web browser both open it is helpful to size and position them so both are in view. That way we can make code changes and refresh the browser to see our updates.



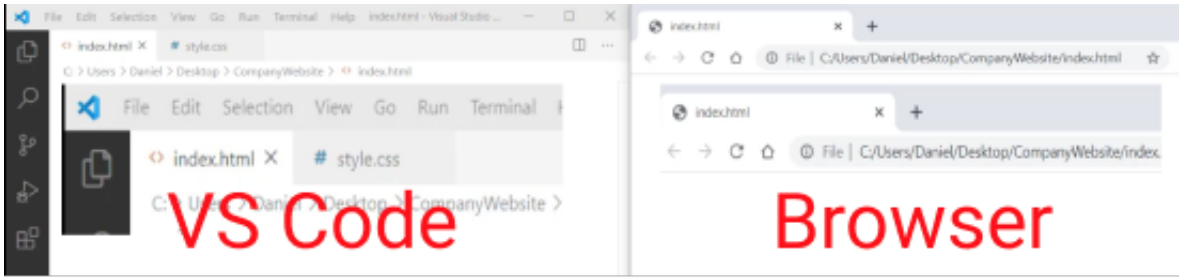
© 2021 Zenva Pty Ltd. All rights reserved.

Live Coding – Project Setup

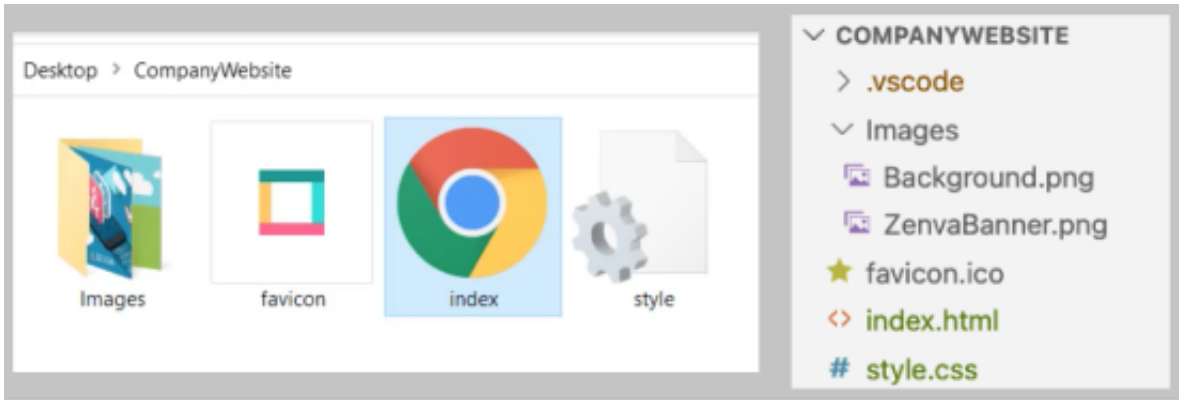
Before diving into building a responsive website, you want to setup your coding environment with a web page (index.html) and a stylesheet (style.css) first. You can use the default pages from the online code editor we use here

© 2021 Zenva Pty Ltd. All rights reserved.

In the last lesson we setup our code editor and browser and positioned them so we can see both at the same time.



We also setup our **project folder** (CompanyWebsite) which contains our **index.html** (web page) file, our **style.css** (stylesheet) file and copied from the our project course downloads the **images** (folder with two images) and our **favicon.ico** (icon file).



For the best course experience, it's important that the steps from the last lesson have all these **files** and **sub-folders** inside a **single parent folder** and that **folder** open within our code **editor**. In this lesson, we begin setting up the main structure for our website.

HTML Web Page Structure

Real world websites include **HTML Document elements** that provide **structure** and **information** to the browser about the web page (note that the **capitalization** is important). As is, this code sets up our web page correctly but currently displays nothing.

```
<!DOCTYPE html>
<html lang="en">
    <head>
    </head>
    <body>
    </body>
</html>
```

HTML Document Elements	Description
HTML Document Language (**<!DOCTYPE html**)	The Document Language or Type tells the browser this code is HTML. It is a single line at the top of the web page file
Root Element (**<html></html>**)	Opening and Closing elements that wrap the entire web page. We also include an attribute for language. Notice in the code block we set this to English (<html lang="en">)

© 2021 Zenva Pty Ltd. All rights reserved.

Head Element (**<head></head>**)	Opening and Closing elements that are used primarily to wrap informational elements for the web page. This includes such things as the **Title** and **Icon** the browser tab will display; **Metadata**; as well as links to external resource files (most commonly **CSS stylesheets** and **site icons**)
Body Element (**<body></body>**)	Opening and Closing elements that are used primarily to wrap the HTML elements of the web page to be displayed

HTML Head Elements

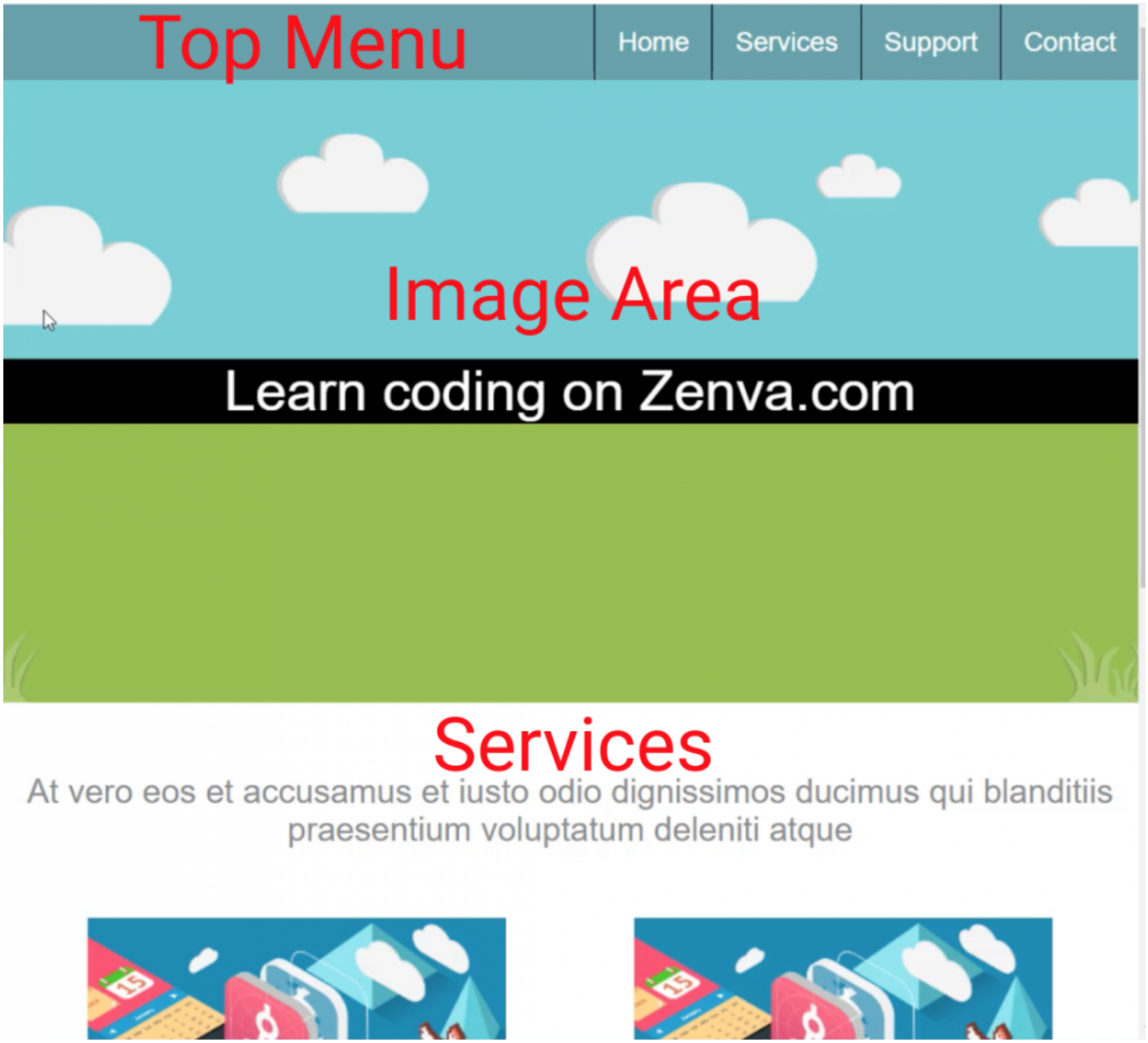
HTML **Head** elements includes such things as the **Title** and **Icon** the browser tab will display as well as **CSS Style** blocks (<style></style>) or links to external resource files (most commonly **CSS stylesheets** and **site icons** such as the **favicon.ico** shown in the code block).

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Learn web development on ZENVA.com</title>
        <meta charset="UTF-8">
        <meta name="description" content="This is what Google will show.">
        <link rel="icon" href="favicon.ico">
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
    </body>
</html>
```

HTML Head Elements	Description
Metadata (**<meta>**)	The Meta element provides information to the browser using a HTML **attributes.** Examples include **Character Set** (<meta charset=”UTF-8″>) and website **Description** (<meta name=”description” content=”This is what Google will show”>)
Web Page Title (**<title></title>**)	The text content displayed in the browser tab
Web Page Icon (**<link rel=”icon” href=”favicon.ico”>**)	Link to our icon (rel=”icon”) displayed in the browser tab. The **favicon** (favorite icon) is a tiny (16×16 pixel) icon with a **.ico** file extension. The **href** attribute points to our **favicon.ico** file
Stylesheet (**<link rel=”stylesheet” href=”style.css”>**)	Link to our CSS **stylesheet** (rel=”stylesheet”). The **href** attribute points to our **style.css** file

© 2021 Zenva Pty Ltd. All rights reserved.

In the last lesson, we began setting up our website **HTML structure** including the overall structure and **head** section. In this lesson, we continue by adding HTML display elements to our **body** section. This is what the final result of our **Responsive Company Website** will look like. We have a navigation bar (Top Menu), a Banner (Image Area), followed by our page content (Services).



Responsive Website

What does it mean to have a **responsive website**? This means that the website will adapt to whatever screen size it is being displayed on and also respond to changes in that screen size. Notice in the image how the website adapts its layout in response the screen size getting smaller. For example, the navigation bar is horizontal on the wide screen and adapts to be vertical on a small screen (picture viewing website on computer, tablet and phone).

© 2021 Zenva Pty Ltd. All rights reserved.

Navigation Bar (Top Menu)

Let's begin creating our **Top Menu** by setting up a container for our navigation elements. To do this in HTML we can use a general and versatile element called a **div** tag. This tag is one of the most used in HTML because it essentially acts as a container for other elements allowing you to group elements with all the content between an opening and closing **div** tag taking up one complete horizontal line on our web page. This means, by default, a **div** container is sized as **width** equal to **100%** of its **parent** (in this case the **body** element) and a **height** equal to the height of its **contents**. We can now attach a **class** attribute to the **div** tag allowing us to **style** the container and its children using **CSS** which we will put inside our **stylesheet** (style.css)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- existing code -->
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" />
    <meta name="description" content="This is what Google will show." />
    <link rel="icon" href="favicon.ico" />
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <!-- new code -->
    <div class="nav">
    </div>
  </body>
</html>
```

Next, we add the four Navigation Bar (Top Menu) choices inside an **Unordered List** (<ul>) as **List Items** (<li>).

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- existing code -->
  </head>
  <body>
    <div class="nav">
    <!-- new code -->
      <ul>
        <li>Home</li>
```

© 2021 Zenva Pty Ltd. All rights reserved.

```
        <li>Services</li>
        <li>Support</li>
        <li>Contact</li>
      </ul>
    </div>
  </body>
</html>
```

HTML Body Elements	Description
	Generic container for other HTML elements. In this example we provide a **CSS Class Name** (class=”nav”) allowing us to style the container and its children using CSS which we will put inside our stylesheet (style.css)
	Unordered List. Contains, as its Children, List Items ( ) with no specific sequence. Defaults to displaying the tags as a vertical list with bullet points
Home	List Item. Displays the content placed inside its tags. In this example, the text “Home”

As we can see, **without CSS styling** applied it does not resemble the image above of our completed Website **Top Menu**.

- Home
- Services
- Support
- Contact

Website Content Container

In the same way we wrapped the **Top Menu**, we will now wrap the rest of the site in a **div** container with a **CSS Class Name** (class=”wrapper”). Inside this wrapper **div** tag we will continue to use **div** tags with **CSS Class Names** to further sub-divide the sections of our web page into smaller containers with their own **CSS Class Names**. Since **CSS** styling is an essential aspect of **responsive web design**, being able to **select** specific sections or groups of sections to apply CSS styles to allow us to take full control of the look and feel of our website.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- existing code -->
  </head>
  <body>
    <!-- existing code -->
    <div class="nav">
    </div>
    <!-- new code -->
    <div class="wrapper">
    </div>
  </body>
</html>
```

© 2021 Zenva Pty Ltd. All rights reserved.

Website Content (Image Area)

Inside the **div** with a **class** of “wrapper”, add another container **div** with a **class** of “image-area” then nest another **div** inside with **two classes** “**image-text**” and “**centered-container**“. You can add as many classes as you need to a single element to achieve your desired result. In the CSS **stylesheet** (style.css) we will style both these classes separately. Not only can one element have many CSS classes but any CSS class can also be applied to many elements.

Inside this **div** we provide our **Image Area** text “Learn coding at Zenva.com”. Notice that when we refresh our web browser we see the text content but since we haven’t applied CSS styles yet it’s just default font text displayed on the screen. Keep in mind that you can name your custom classes anything that makes sense for you or the naming convention you are following.

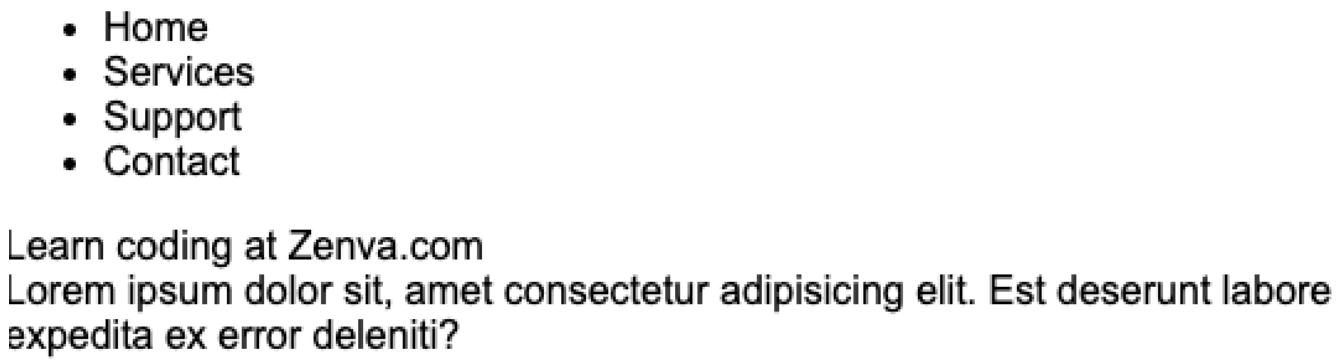
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- existing code -->
  </head>
  <body>
    <!-- existing code -->
    <div class="nav">
    </div>
    <div class="wrapper">
      <!-- new code -->
      <div class="image-area">
        <div class="image-text centered-container">Learn coding at Zenva.com</div>
      </div>
    </div>
  </body>
</html>
```

Website Content (Intro Text)

Inside the **div** with a **class** of “wrapper”, and as a **sibling** of the **div** with a **class** of “image-area”, add another **div** with **two** classes “**intro-text**” and “**centered-container**“. Inside this one, we’re going to include a **placeholder text** known as “lorem ipsum”. You can just write in any text you want here to fill up space, copy the text from the **index.html** file of the **course downloads**, search for “lorem ipsum” placeholder text on the Internet or from within **VS Code** simply place your cursor between the two **div** tags and type “lorem15” then press the **tab key** (lorem15). When you press the **tab key**, VS Code will fill in 15 words of “lorem ipsum” placeholder text for you.

© 2021 Zenva Pty Ltd. All rights reserved.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- existing code -->
  </head>
  <body>
    <!-- existing code -->
    <div class="nav">
    </div>
    <div class="wrapper">
      <div class="image-area">
        <div class="image-text centered-container">Learn coding at Zenva.com</div>
      </div>
      <!-- new code -->
      <div class="intro-text centered-container">
        Lorem ipsum dolor sit, amet consectetur adipisicing elit. Est deserunt labore
 expedita ex error deleniti?
      </div>
    </div>
  </body>
</html>
```



Website Content (Services)

Inside the **div** with a **class** of “wrapper”, and as a **sibling** of the **div** with a **class** of “intro-text”, add another container **div** with **two classes** “services” and “centered-container”. Inside this container **div** place four sibling container **div** tags each with the **class** of “single-service”. Inside each of these containers we will place an **image** (inside an **img** tag) and some **text** (inside a Paragraph or **p** tag).

For the **image** we link to the **ZenvaBanner.png** file in our **images** folder (). Note that because our image file is not in the same folder as the file that is referencing it we need to include a **path** to the file as well as the file’s **name**.

For the text, include inside each **p** tag placeholder text as we did above. Again, you can just write in any text you want here to fill up space, copy the text from the **index.html** file of the **course downloads**, search for “lorem ipsum” placeholder text on the Internet or from within **VS Code** simply place your cursor between the two **p** tags and type “lorem10” then press the **tab key** (<p>lorem10). When you press the **tab key**, VS Code will fill in 10 words of “lorem ipsum” placeholder text for you.

© 2021 Zenva Pty Ltd. All rights reserved.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- existing code -->
  </head>
  <body>
    <!-- existing code -->
    <div class="nav"></div>
    <div class="wrapper">
      <!-- existing code -->
      <div class="image-area"></div>
      <div class="intro-text centered-container"></div>
      <!-- new code -->
      <div class="services centered-container">
        <div class="single-service">
          <img src="Images/ZenvaBanner.png" />
          <p>At vero eos et accusamus et iusto odio dignissimos ducimus</p>
        </div>
        <div class="single-service">
          <img src="Images/ZenvaBanner.png" />
          <p>At vero eos et accusamus et iusto odio dignissimos ducimus</p>
        </div>
        <div class="single-service">
          <img src="Images/ZenvaBanner.png" />
          <p>At vero eos et accusamus et iusto odio dignissimos ducimus</p>
        </div>
        <div class="single-service">
          <img src="Images/ZenvaBanner.png" />
          <p>At vero eos et accusamus et iusto odio dignissimos ducimus</p>
        </div>
      </div>
    </div>
  </body>
</html>
```

© 2021 Zenva Pty Ltd. All rights reserved.

- Home
- Services
- Support
- Contact

Learn coding at Zenva.com
At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque

At vero eos et accusamus et iusto odio dignissimos ducimus

At vero eos et accusamus et iusto odio dignissimos ducimus

At vero eos et accusamus et iusto odio dignissimos ducimus

At vero eos et accusamus et iusto odio dignissimos ducimus

© 2021 Zenva Pty Ltd. All rights reserved.

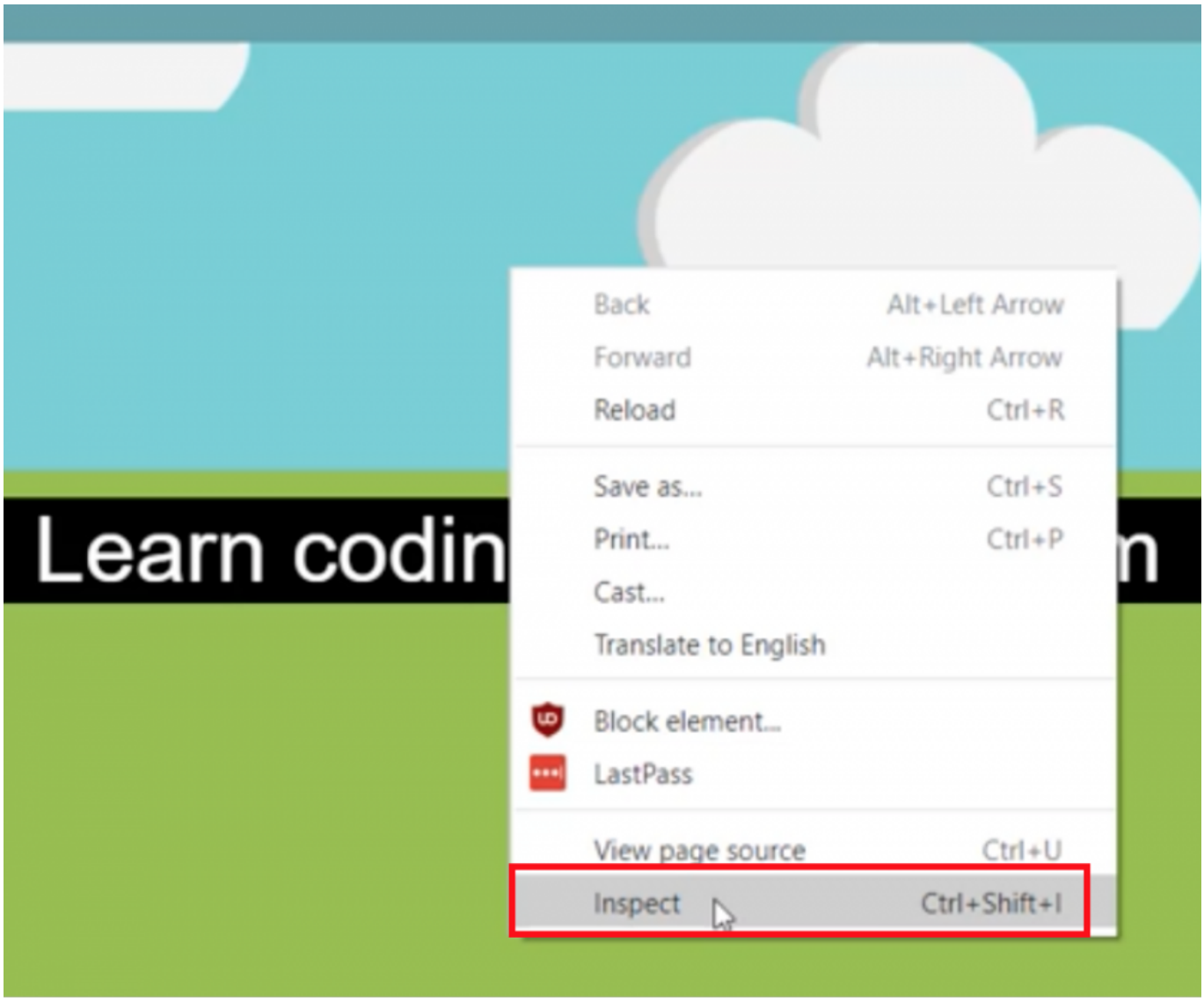
Although it may not look like it, the web page image above has the same HTML code as the website image shown at the beginning of this lesson. Why do they look so different? The answer is CSS styling which we'll work on in the upcoming lessons.

© 2021 Zenva Pty Ltd. All rights reserved.

Before we move on to **styling** our website and making it **responsive** we will cover an important tool to understanding and debugging your website. In the Google Chrome browser, this is known as **Chrome Dev Tools**. Other modern browsers have similar web development tools.

Open Chrome Dev Tools

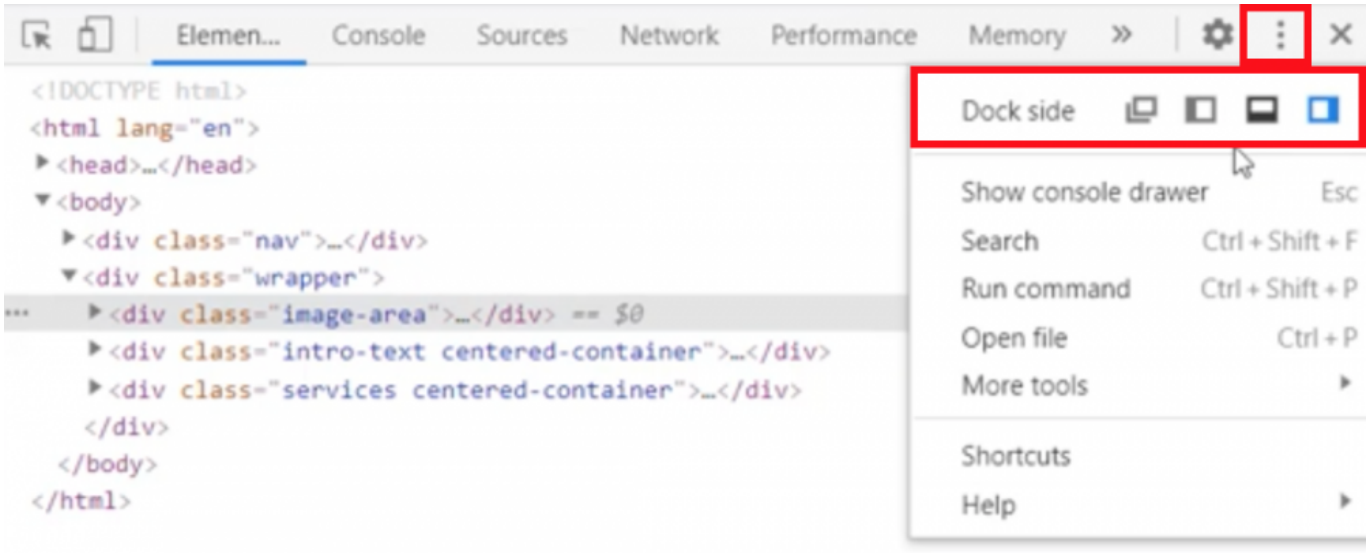
Right-click on any web page and press **Inspect** (or use keyboard shortcut shown) to open the browser's dev tools.



© 2021 Zenva Pty Ltd. All rights reserved.

Positioning Chrome Dev Tools

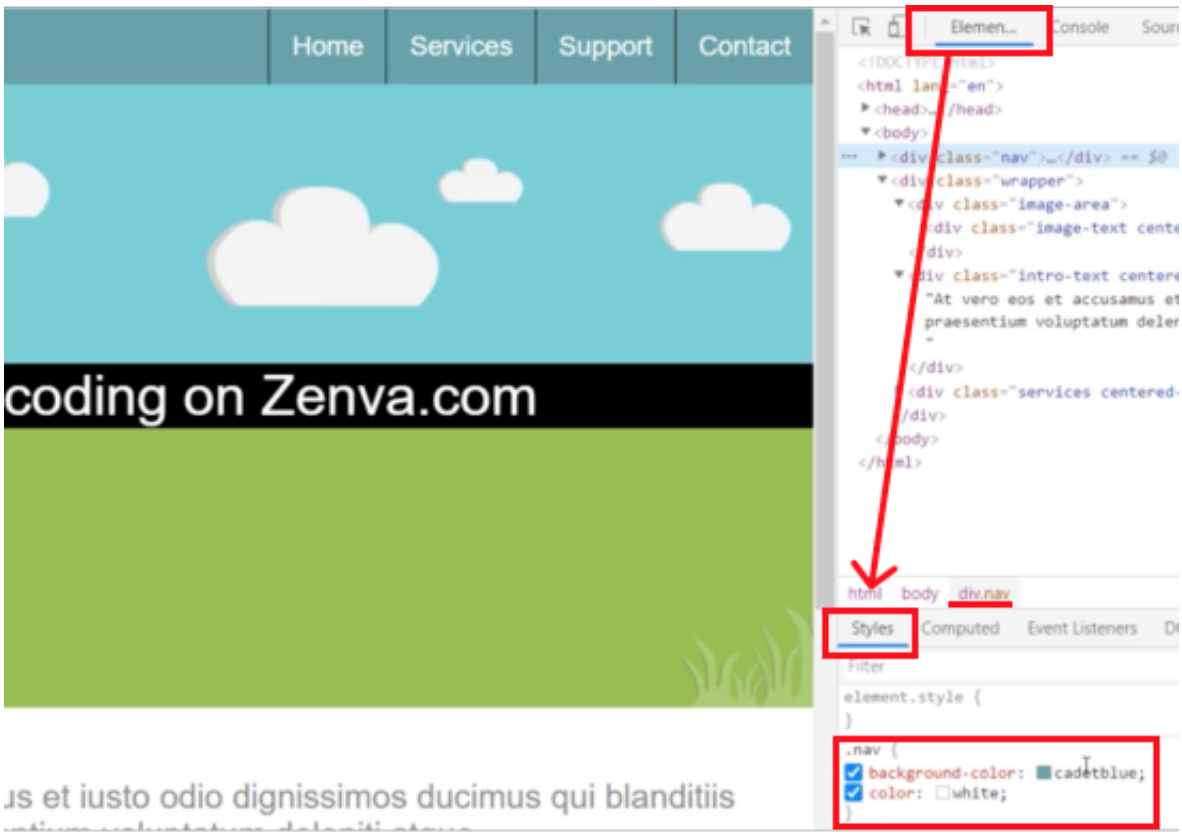
Using the icon shown you can dock Chrome Dev Tools against the left, right, bottom or even undock the Dev Tools to make them a floating window that can be moved to another monitor if you wish.



Elements Tab

On the upper **Dev Tools** panel, select the **Elements Tab**. Hover over any of the **div** tags to see the corresponding **element** highlighted in web page. Select an element and in the lower **Dev Tools** panel select the **Styles Tab**. These are the CSS styles applied to this element. For example, select the **div** with a **class** of “nav” (Top Menu) and notice the two **CSS properties** (background-color and color) and the **values** assigned to them for the **CSS Class Selector** of **.nav**

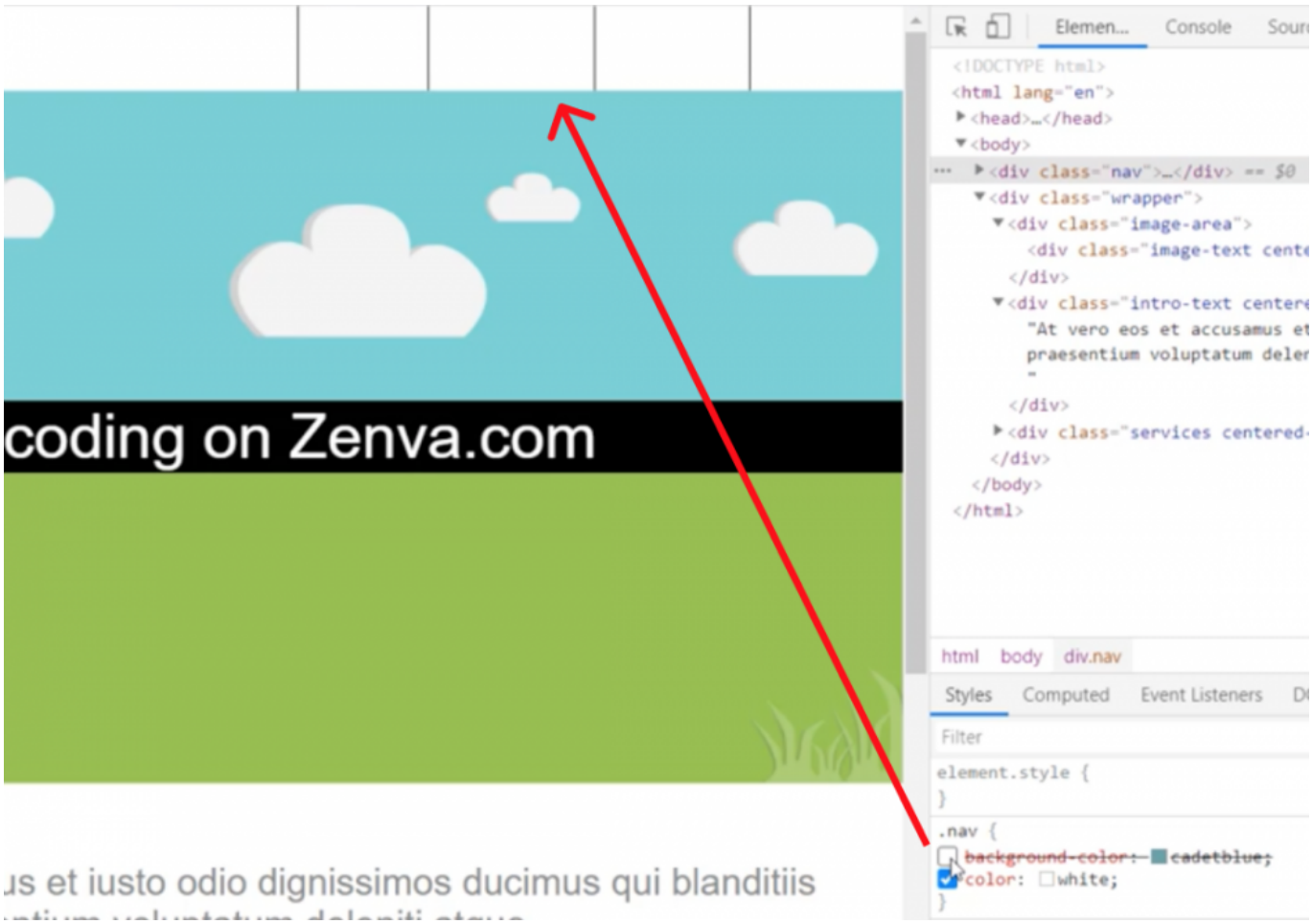
© 2021 Zenva Pty Ltd. All rights reserved.



Styles Tab

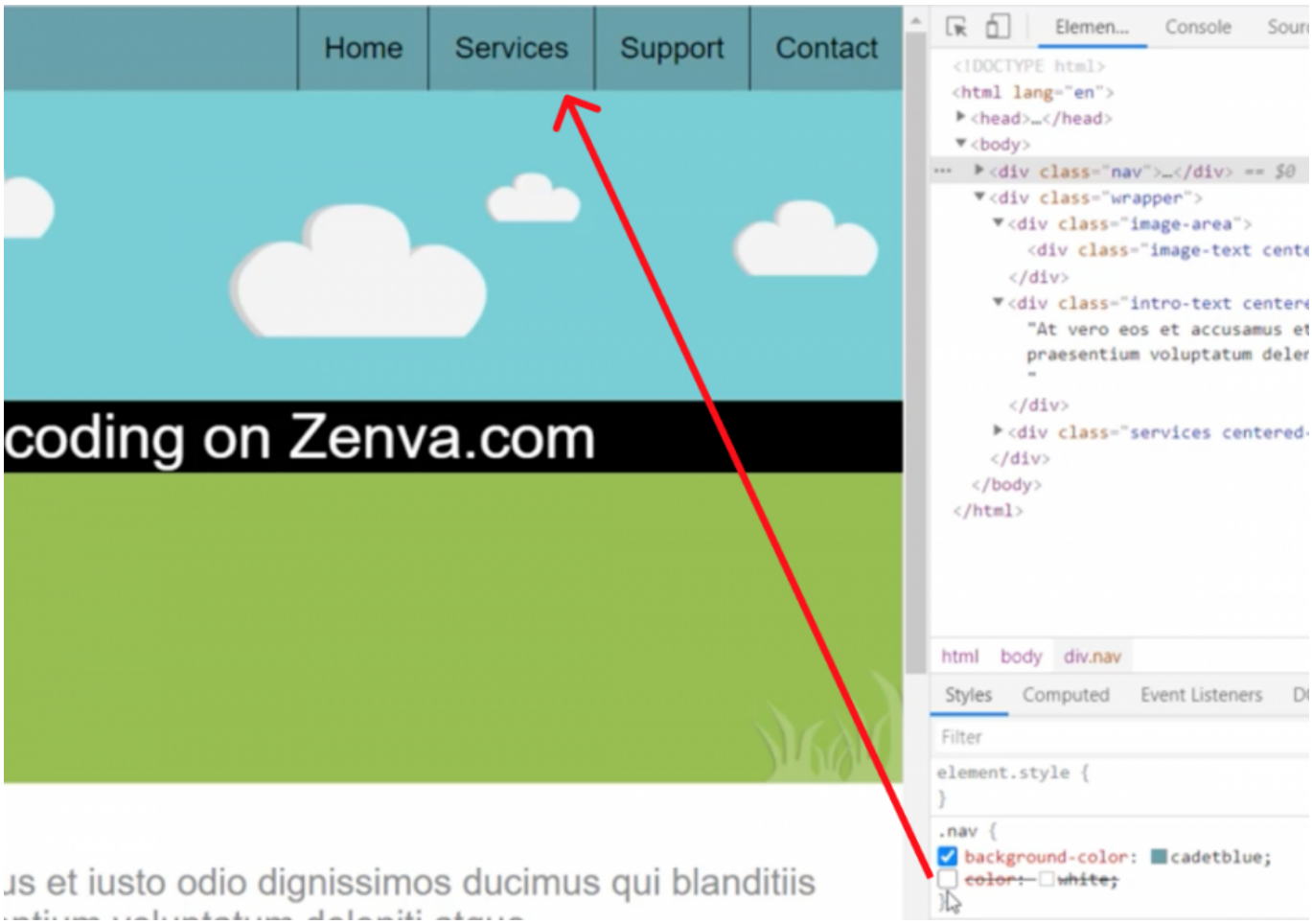
On the bottom panel of the **Dev Tools** you can see the styles that are applied. In this image, you can see the **styles** being applied to the Top Menu (class=”nav”). Notice the two **CSS properties** (background-color and color) and the **values** assigned to them for this **class** (.nav). You can interact with a website by changing its CSS. For example, remove the **background-color** style by clicking on the checkbox next to this **property**. Notice what happens to the Top Menu when its background-color disappears (not visible because we have the white text on a now default white background).

© 2021 Zenva Pty Ltd. All rights reserved.



For another example, bring back the **background-color** and **uncheck** the **color** property. Notice that the text in the Top Menu changes to its **default** of black.

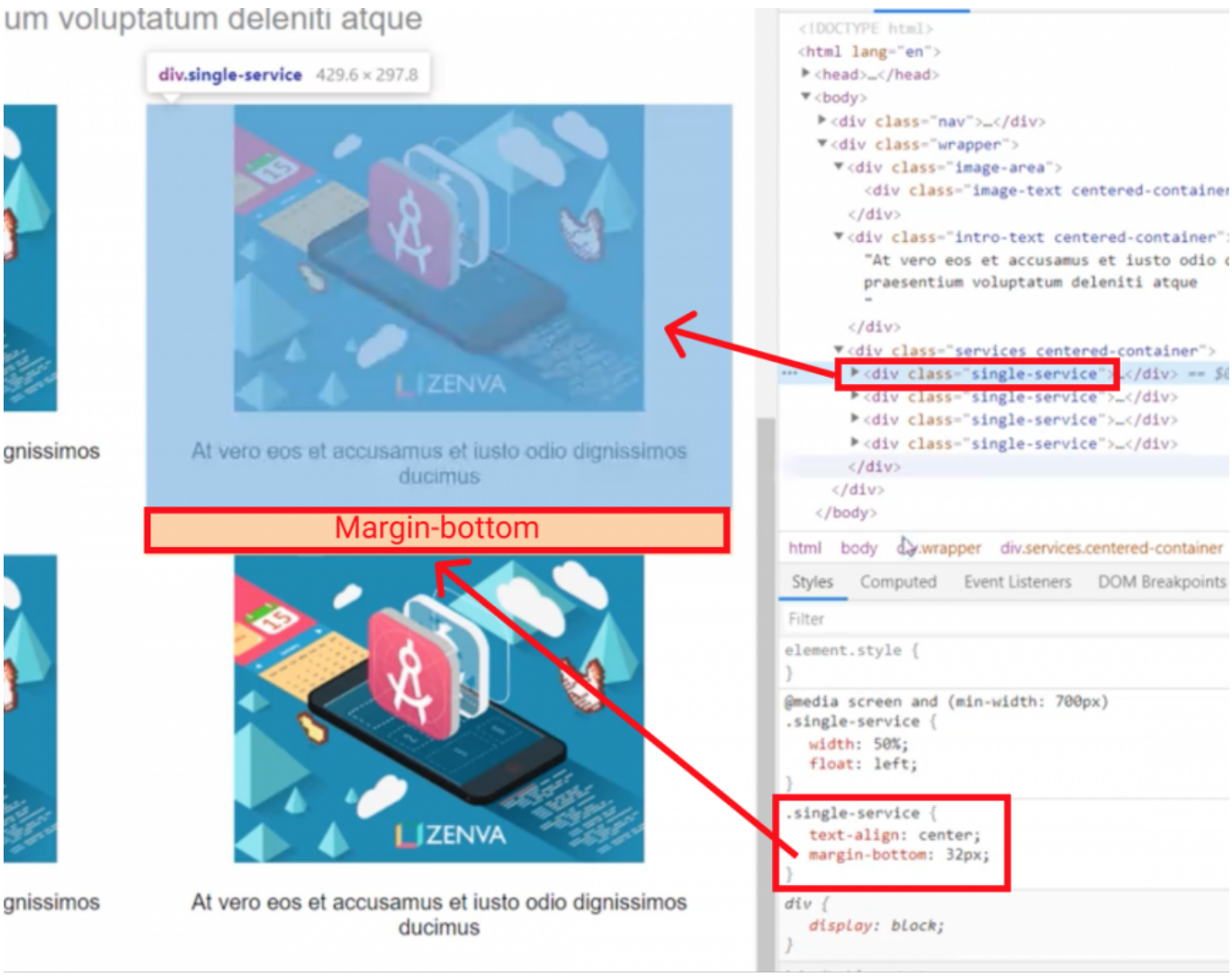
© 2021 Zenva Pty Ltd. All rights reserved.



Styles Tab (more detail)

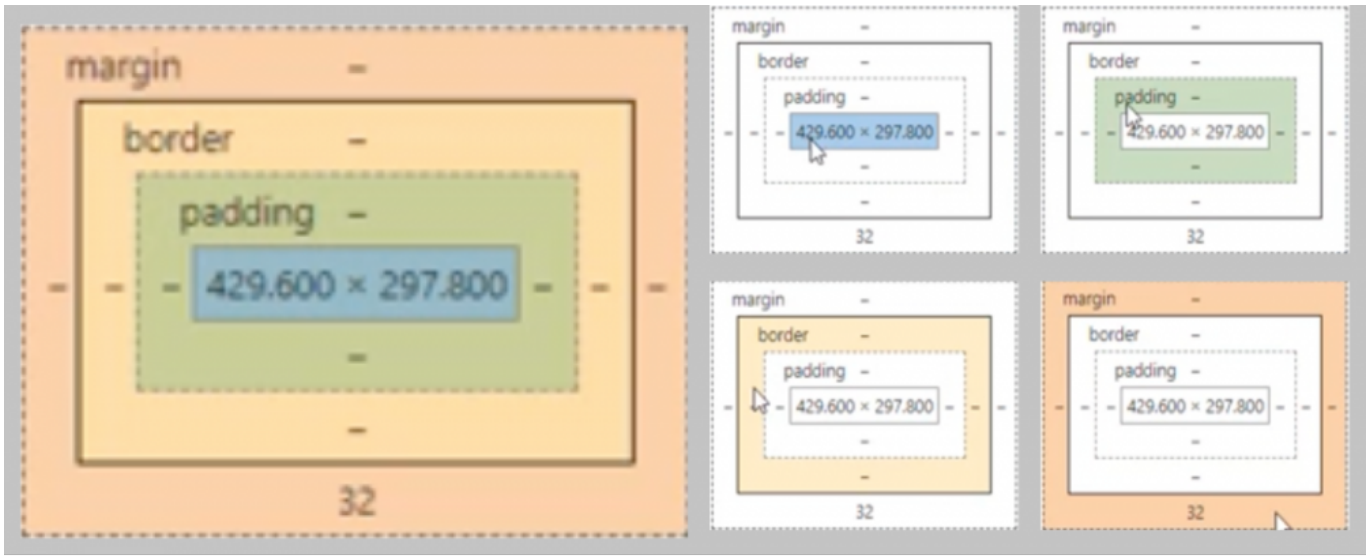
For a different example, select one of the **div** elements with a **class** of “single-service” (Services) and notice the two **CSS properties** (text-align and margin-bottom) and the **values** assigned to them for the **CSS Class Selector** of **.single-service**

© 2021 Zenva Pty Ltd. All rights reserved.



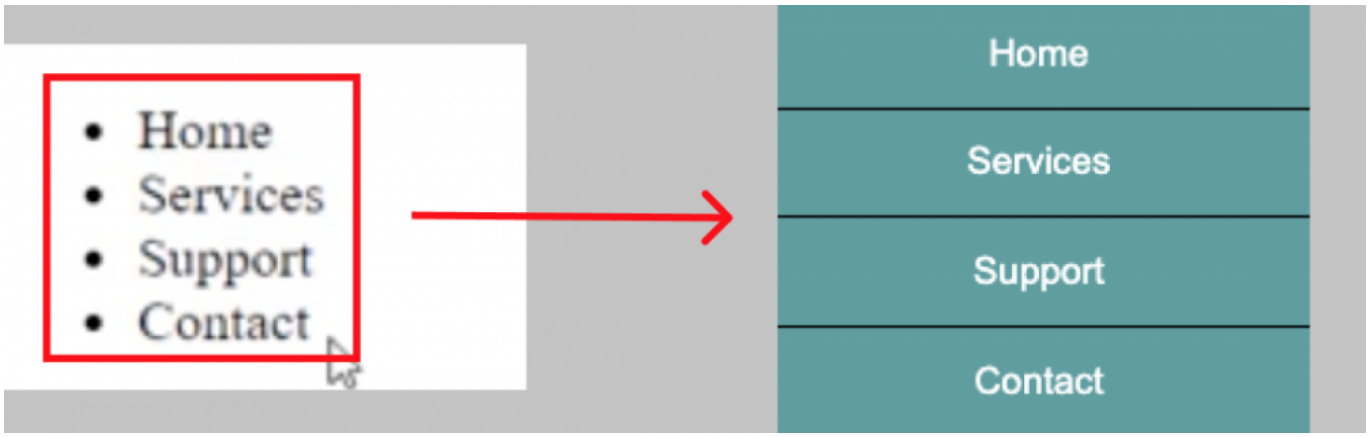
Notice the element selected on the web page and the orange color area at the bottom of the selection. This shows the 32px margin currently applied to the **div** with a **class** of “single-service”. This margin is what separates one “single-service” element from the one below it. We could adjust this margin in the **Dev Tools**, instead of in code and refreshing the browser, to determine our optimum choice for separation. Once satisfied we could then make any required changes to our code to make the adjustment permanent (all changes made in the Dev Tools disappear when the page is refreshed). Scroll down within the **Styles Tab** to see more detail. The **box** helps us visualize the size of the element as well as its padding, border, and margin.

© 2021 Zenva Pty Ltd. All rights reserved.



© 2021 Zenva Pty Ltd. All rights reserved.

In this lesson, we are going to be applying CSS to our **navbar** (Top Menu). To do this, we will add **CSS** styles in our **stylesheet** (style.css). The goal of this lesson is to transform our from a plain bullet list to a styled menu.



Body Element

The first step is to setup the overall styles for our website. We do this inside **style.css** by targeting the **body** tag and applying **CSS properties** and **values** inside curly braces **{css-properties-list}**

```
body {
    background-color: white;
    color: black;
    font-size: 16px;
    font-family: Arial, Helvetica, sans-serif;
    padding: 0px;
    margin: 0px;
}
```

CSS Property and Value	Description of Web Page Effect
background-color: white;	Defaults the **background color** of the web page and all elements on the web page to **white**. If other elements do not specify a background color then this default will be used
color: black;	Defaults the **font color** of all text on the web page including all text within elements on the web page to **black**. If other elements do not specify a font color then this default will be used
font-size: 16px;	Defaults the **font size** of all text on the web page including all text within elements on the web page to **16 pixels** (16px). If other elements do not specify a font size then this default will be used
font-family: Arial, Helvetica, sans-serif;	Defaults the **font family** of all text on the web page including all text within elements on the web page to **Arial**. If the Arial font is not available then **Helvetica** will be used. If Helvetica font is not available then the browser's default **sans-serif** font will be used. If other elements do not specify a font family then this default will be used
padding: 0px;	Browsers generally provide some **padding**

© 2021 Zenva Pty Ltd. All rights reserved.

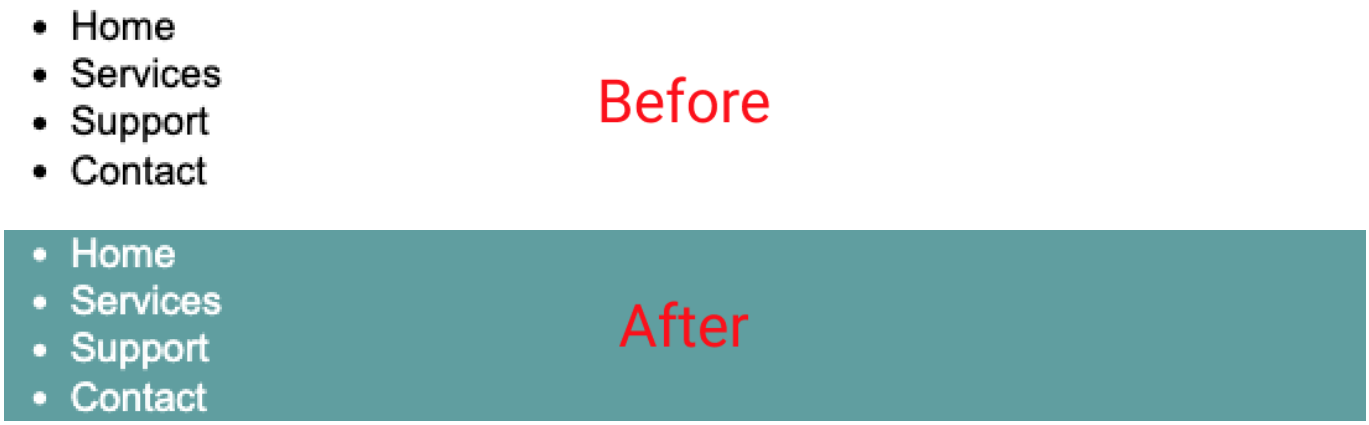
	within web pages by default. Since we want to take full control of styling our responsive website we change the default padding to be zero. If other elements do not specify a padding then this default will be used
margin: 0px;	Browsers generally provide some **margin** around web pages by default. Since we want to take full control of styling our responsive website we change the default margin to be zero. If other elements do not specify a margin then this default will be used

Nav Class

When we setup our HTML structure we placed our **navbar** (Top Menu) inside a **div** tag with the **CSS Class Name** of “nav”. Inside **style.css** we target this element using the CSS Class Name **Selector** (dot + class-name which is **.nav**). Inside the curly braces that following we place the CSS rules **{css-properties-list}** to style our **navbar** container.

```
/* existing code */
body {
}

/* new code */
.nav {
    background-color: cadetblue;
    color: white;
}
```

CSS Property and Value	Description of Web Page Effect
background-color: cadetblue;	Changes the **background color** of the **navbar** to **cadetblue**
color: white;	Changes the **font color** of all **navbar text** to **white**



Nav Class – Unordered List

The next step is to style the **container** for our **Menu Items** ( ) which is an **Unordered List** ( ). While we could have given the **<ul>** and each **<li>** its own class, we don’t need to because we can use the container Class Name of “nav” and target **<ul>** and **<li>** tags inside it. We do this by using both the **Class Name** of “nav” and the **ul** tag name followed by curly braces and our CSS

© 2021 Zenva Pty Ltd. All rights reserved.

rules. This approach targets only child elements of an element with a Class Name of “nav”.

```
/* existing code */
body {
}

/* existing code */
.nav {
    background-color: cadetblue;
    color: white;
}

/* new code */
.nav ul {
    margin: 0px;
    padding: 0px;
    list-style-type: none;
}
```

CSS Property and Value	Description of Web Page Effect
padding: 0px;	Ensures there is no **padding** around the ** **
margin: 0px;	Ensures there is no **margin** around the ** **
list-style-type: none;	Removes the default bullet point from each ** **

Nav Class – List Item

The next step is to style the **Menu Items** (tags) which are nested inside an **Unordered List** ( ) nested inside the **div** container with Class Name of “nav”. While we could have given each **<li>** its own class, we don’t need to because we can use the container Class Name of “nav” and the **ul** tag to target the **<li>** tags inside it. We do this by using the **Class Name** of “nav”, the **ul** tag name and the **li** tag name followed by curly braces and our CSS rules. This approach targets only child elements of an Unordered list inside an element with a Class Name of “nav”.

```
/* existing code */
body {
    background-color: white;
    color: black;
    font-size: 16px;
    font-family: Arial, Helvetica, sans-serif;
    padding: 0px;
    margin: 0px;
```

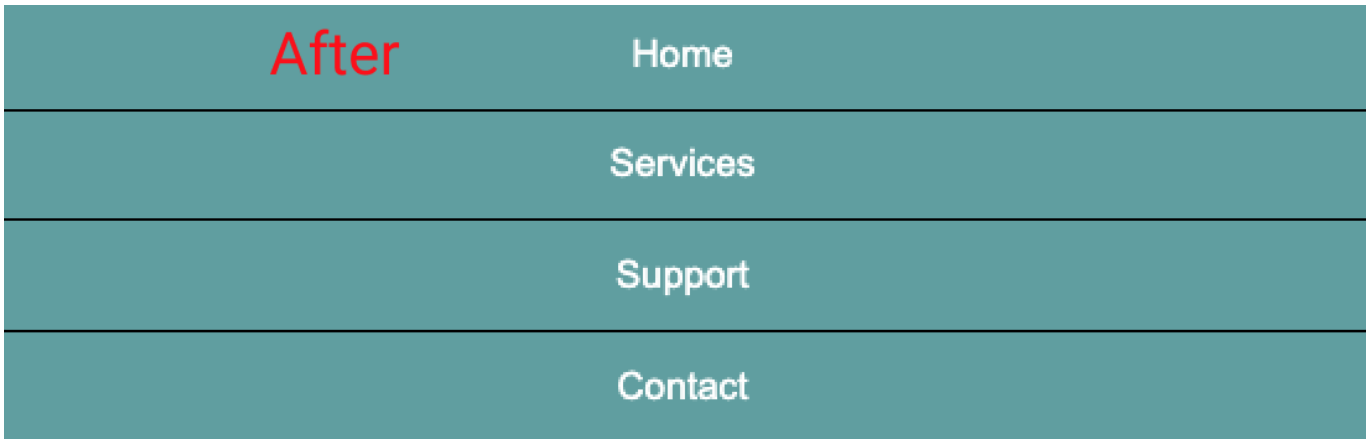
© 2021 Zenva Pty Ltd. All rights reserved.

```
}

/* existing code */
.nav {
    background-color: cadetblue;
    color: white;
}

/* existing code */
.nav ul {
    margin: 0px;
    padding: 0px;
    list-style-type: none;
}

/* new code */
.nav ul li {
    padding: 16px;
    text-align: center;
    font-size: 1.2rem;
    border-bottom: 1px solid black;
}
```

CSS Property and Value	Description of Web Page Effect
padding: 16px;	Sets the **padding** around each ** ** to be 16 pixels on all sides (16px)
text-align: center;	Keeps the **text** of each ** ** in the center of its block even as block size changes
font-size: 1.2rem;	Sets the **font size** of the **text** of each ** ** to be 1.2 times the **root** element's **font size**. In this example, the root is the **body** element which we set the font size to 16px. We multiply by 1.2 to get the ** ** text font **size** (1.2 x 16px = **19.2px**)
border-bottom: 1px solid black;	Sets a bottom border so each menu item has a one pixel solid black line separating each item to give the appearance of individual buttons

This provides a styled vertical menu. This vertical stacking of **<li>** elements is the default behaviour and we have not introduced the CSS styling yet to get the responsive website appearance we want in the end.

© 2021 Zenva Pty Ltd. All rights reserved.



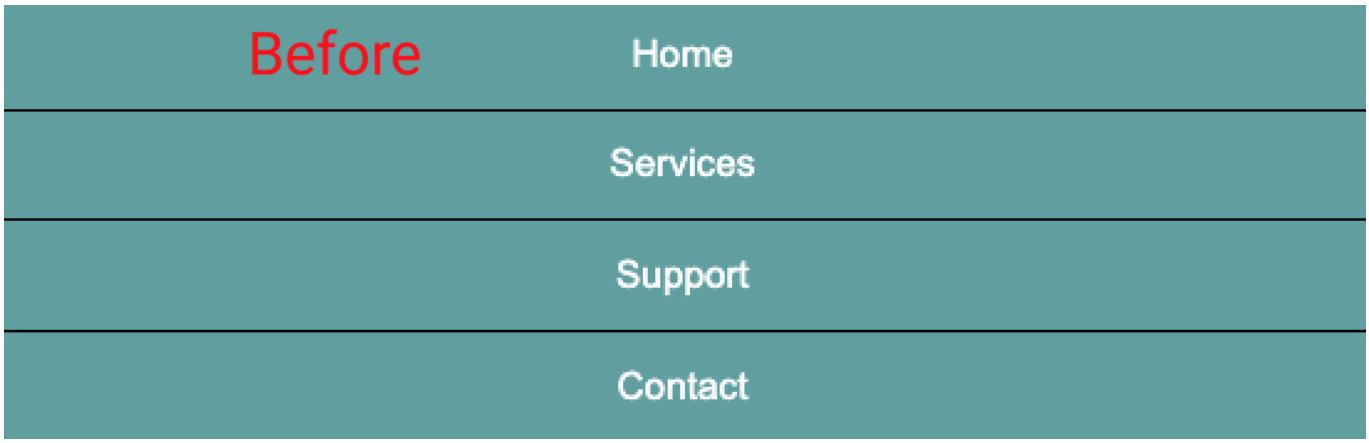
Nav Class – Hover Effect

It is common with menu items to have some effect when the mouse hovers over it. To do this we add what is known as a **CSS pseudo-class** for **hover** and then provide the CSS rules to change the visual state. For our example, we will **change** the **background color** to **black** when the mouse hovers over a menu item. Notice that the **pseudo-class** is added to the CSS Selector using a colon (:) and the name of the effect (.nav ul li**:hover**). Inside the curly braces that follow we provide the temporary CSS properties we want to change while the mouse is hovering over the menu item. As soon as the hover ends the original CSS properties take affect again. To have this effect transition gradually we modify the CSS styles for the element to include the **transition property** (transition: background-color 0.2s;) which changes the background color over a 0.2 second period. This adds a more professional style effect.

```
/* existing code */
body {
}

/* existing code */
.nav {
    background-color: cadetblue;
    color: white;
}

/* existing code */
.nav ul {
    margin: 0px;
    padding: 0px;
    list-style-type: none;
}

/* modified code */
.nav ul li {
    padding: 16px;
    text-align: center;
    font-size: 1.2rem;
    border-bottom: 1px solid black;
    /* add transition property */
    transition: background-color 0.2s;
}

/* new code */
```

© 2021 Zenva Pty Ltd. All rights reserved.

```
.nav ul li:hover {
    background-color:black;
}
```



© 2021 Zenva Pty Ltd. All rights reserved.

In this lesson we will be styling our **banner** (Image Area) including the **text overlay** of “Learn coding on Zenva.com” **centered** with a **black background** as shown in the image below. To do this, we will add **CSS** styles in our **stylesheet** (style.css).

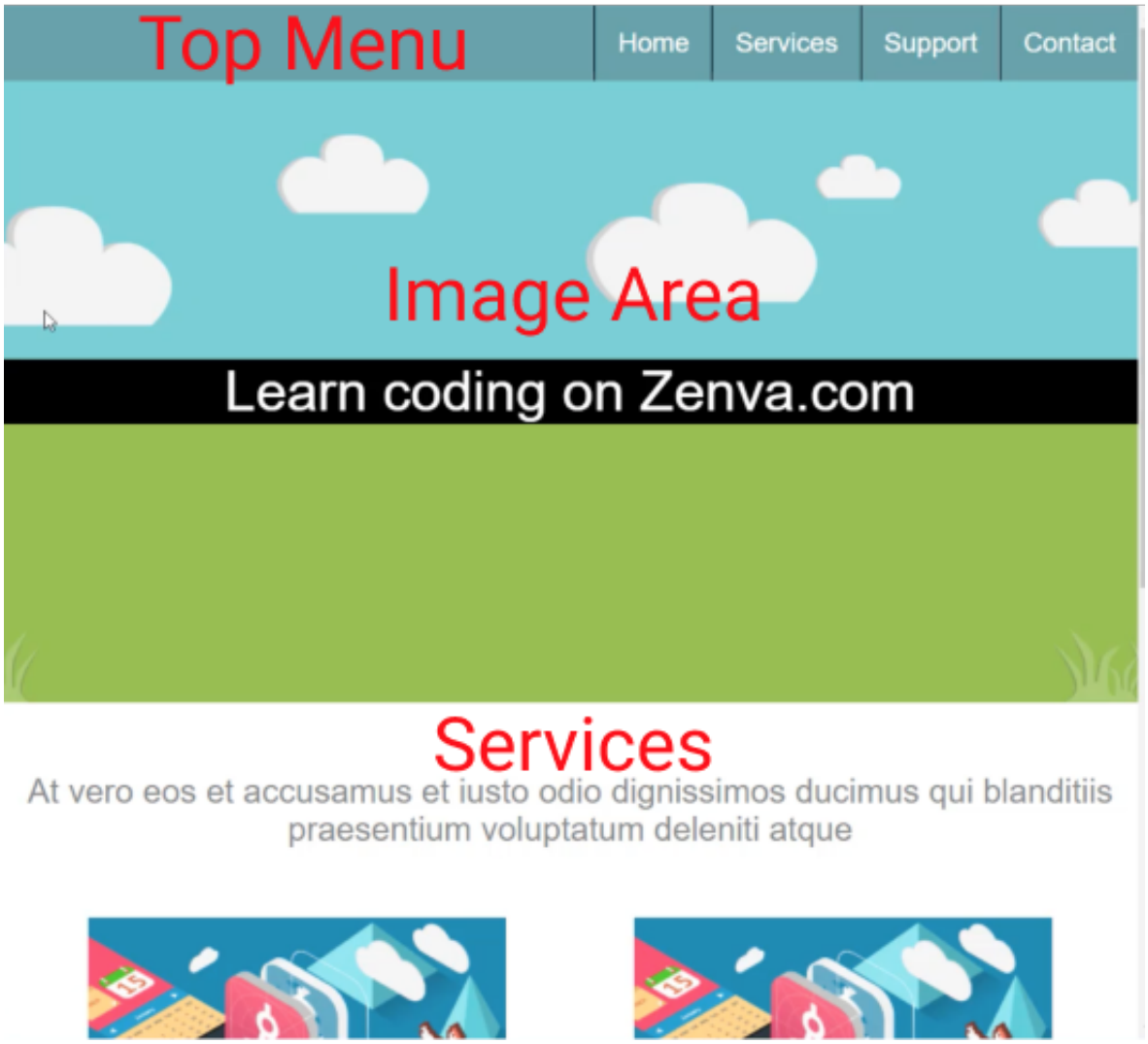


Image Area Class

In our HTML we defined the **Image Area** as a **div** with a CSS Class of “image-area”. Nested inside is a **div** with a class of “image-text centered-container” with text content of “Learn coding at Zenva.com”.

```
<!-- existing code -->
<div class="image-area">
  <div class="image-text centered-container">
    Learn coding at Zenva.com
  </div>
</div>
<!-- existing code -->
```

In our **CSS stylesheet** (style.css) we can now style this banner area to include a background image as shown in the image above. We use the **CSS Class Selector** of dot plus class name (**.image-**

© 2021 Zenva Pty Ltd. All rights reserved.

area) to apply styles to the Image Area **div** element.

```
/* new code */
.image-area {
    background-image:url("Images/Background.png");
    padding-top: 200px;
    padding-bottom: 200px;
    background-position: center center;
    background-size: cover;
}
```

CSS Property and Value	Description of Web Page Effect
background-image: url(“Images/Background.png”);	Sets the **background image** to the **Background.png** image file located in the **Images** folder. Notice that the **url()** includes the relative **path** (folder/filename)
padding-top: 200px;	Sets the **top padding** to 200 pixels (200px) which moves this element’s contents (in this case the div with the text “Learn coding...”) 200 pixels down from the top of the Image Area **div**
padding-bottom: 200px;	Sets the **bottom padding** to 200 pixels (200px) which moves this element’s contents (in this case the div with the text “Learn coding...”) 200 pixels up from the bottom of the Image Area **div**
background-position: center center;	Sets the **background image position** to be located in the **center** (horizontally and vertically). Since the **Background.png** is larger than the space inside this element, we still only see a portion of the actual image
background-size: cover;	Sets the **background image size** to be just large enough to **cover** the space available inside this element. The **Background.png** image is now visible inside this **div** element

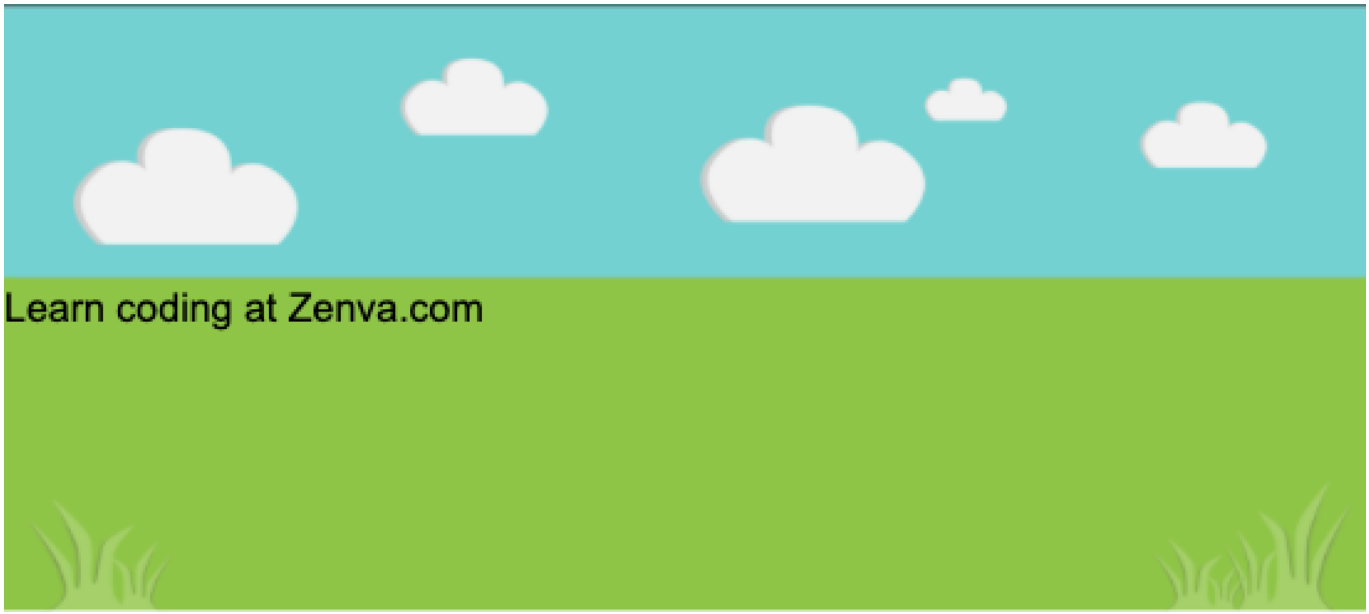


Image Text Class

© 2021 Zenva Pty Ltd. All rights reserved.

In our **CSS stylesheet** (style.css) we add to our existing styles two additional classes (**.image-text** and **.centered-container**). Together, these classes are used to style the text inside our banner (Image Area) to obtain our desired text display result as shown in the image above.

```
/* existing code */
body {
    background-color: white;
    color: black;
    /* font-size used as root for rem calculations */
    font-size: 16px;
    font-family: Arial, Helvetica, sans-serif;
    padding: 0px;
    margin: 0px;
}

/* Top Menu Styles omitted*/

/* existing code */
.image-area {
    background-image:url("Images/Background.png");
    padding-top: 200px;
    padding-bottom: 200px;
    background-position: center center;
    background-size: cover;
}

/* new code */
.image-text {
    text-align: center;
    background-color: black;
    color: white;
    font-size: 2.5rem;
    max-width: 1020px;
    margin: 0px auto;
    padding: 0px 16px;
}
```

CSS Property and Value	Description of Web Page Effect
text-align: center;	Keeps the text “Learn coding…” in the center of its block even as block size changes
background-color: black;	Sets the **background color** of the **text** “Learn coding…” to be **black**
color: white;	Sets the **font color** of the **text** “Learn coding…” to be **white**
font-size: 2.5rem;	Sets the **font size** of the **text** “Learn coding…” to be 2.5 times the **root** element’s **font size**. In this example, the root is the **body** element which we set the font size to 16px. We multiply the root font size by 2.5 to get the **text font size** (2.5 x 16px = **40px**)
max-width: 1020px;	Sets the **maximum width** the text **div** can be. Even if the screen was very large the text **div** would not get any larger than 1020 pixels
margin: 0px auto;	Positions the text **div** container horizontally to be in the center of its parent (**div** with a **class** of

© 2021 Zenva Pty Ltd. All rights reserved.

	“image-area” in this example). The margin is automatically adjusted on the **left** and **right** to keep the text **div** centered as the window is resized
padding: 0px 16px;	Sets **padding** on **top** and **bottom** to be zero pixels. Sets **padding** on **left** and **right** to be 16 pixels (16px) which ensures that **horizontally** there will always be at least 16 pixels between the text and the element’s border

© 2021 Zenva Pty Ltd. All rights reserved.

In this lesson we will be working on styling our **Intro Text** and our **Services**. To do this, we will add **CSS** styles in our **stylesheet** (style.css).

Intro Text Class

In our **HTML** we defined the **Intro Text** as a **div** with two **classes** (**intro-text** and **centered-container**) containing about 15 words of text.

```
<!-- existing code -->
      <div class="intro-text centered-container">
        At vero eos et accusamus et iusto odio dignissimos ducimus qui
        blanditiis praesentium voluptatum deleniti atque
      </div>
<!-- existing code -->
```

In our **CSS stylesheet** (style.css) we add to our existing styles two additional classes (**.intro-text** and **.centered-container**). Together, these classes are used to style the **Intro Text** above the **Services** section. Our goal is to obtain the text display format as shown in the image above for **Intro Text**. Since the class **.centered-container** is used by both the **Image Area** and **Intro Text**, it was coded in the last lesson. Notice that both classes (**.intro-text** and **.centered-container**) contain a **Padding** property. In CSS, the last style to be applied wins. Therefore, the **top** and **bottom padding** applied to the **Intro Text** is 50 pixels not zero (as defined in **.centered-container** style rules).

© 2021 Zenva Pty Ltd. All rights reserved.

```
/* existing code */
body {
    background-color: white;
    color: black;
    /* used as root for rem calculations */
    font-size: 16px;
    font-family: Arial, Helvetica, sans-serif;
    padding: 0px;
    margin: 0px;
}

/* Top Menu Styles omitted */
/* Image Area Styles omitted */

/* existing code */
.centered-container {
    max-width: 1020px;
    margin: 0px auto;
    /* top|bottom left|right */
    padding: 0px 16px;
}

/* new code */
.intro-text {
    color: grey;
    font-size: 1.5rem;
    text-align: center;
    padding-top: 50px;
    padding-bottom: 50px;
}
```

CSS Property and Value	Description of Web Page Effect
color: grey;	Sets the **font color** of the **Intro Text** to be **grey**
font-size: 1.5rem;	Sets the **font size** of the **Intro Text** to be 1.5 times the **root** element's **font size**. In this example, the root is the **body** element which we set the font size to 16px. We multiply the root font size by 1.5 to get the **text font size** (1.5 x 16px = **24px**)
text-align: center;	Keeps the **Intro Text** in the center of its block even as block size changes
padding-top: 50px;	Sets the **top padding** to 50 pixels (50px) which moves the **Intro Text** 50 pixels down from the **Image Area**
padding-bottom: 50px;	Sets the **bottom padding** to 50 pixels (50px) which moves the **Intro Text** 50 pixels up from the **Services Area**

© 2021 Zenva Pty Ltd. All rights reserved.

Single-Service Class

In our **CSS stylesheet** (style.css) we add two additional styles (**.single-service** and **.single-service p**). Recall from previous lessons, placing the **p** tag selector after the **.single-service** class in CSS means select all **Paragraph** elements (**<p>** tags) that are nested inside elements containing a **class** of “single-service”.

```
<!-- existing code omitted -->

<!-- div repeated 4 times -->
<div class="single-service">
  <img src="Images/ZenvaBanner.png" />
  <p>At vero eos et accusamus et iusto odio dignissimos ducimus</p>
</div>

<!-- existing code omitted -->
```

The **div** with a **class** of “single-service” containing an **Image** (**<img>** tag) and a **Paragraph** (**<p>** tag) occurs four times in our **HTML** file but we only need to define the **CSS** style once to have it applied to all four HTML occurrences.

```
/* existing code */
```

© 2021 Zenva Pty Ltd. All rights reserved.

```
body {
    background-color: white;
    color: black;
    /* font-size used as root for rem calculations */
    font-size: 16px;
    font-family: Arial, Helvetica, sans-serif;
    padding: 0px;
    margin: 0px;
}

/* Top Menu Styles omitted*/
/* Image Area Styles omitted */
/* Intro Text Styles omitted */

/* new code */
.single-service {
    text-align: center;
    margin-bottom: 32px;
}

.single-service p {
    padding: 0px 16px;
}
```

CSS Property and Value	Description of Web Page Effect
text-align: center;	Aligns both the **Image** and the **Paragraph** inside the **div** with a **class** of “single-service” to the center of its block even as block size changes
margin-bottom: 32px;	Provides 32 pixels of space between each **div** with a **class** of “single-service” and any HTML element below it
padding: 0px 16px;	Sets **padding** on **top** and **bottom** to be zero pixels. Sets **padding** on **left** and **right** to be 16 pixels (16px) which ensures that the **Paragraph** (**<p>** tag) will always have at least 16 pixels of space in the **horizontal** direction between its text and the element’s border

© 2021 Zenva Pty Ltd. All rights reserved.

Before

At vero eos et accusamus et iusto odio dignissimos ducimus

At vero eos et accusamus et iusto odio dignissimos ducimus

© 2021 Zenva Pty Ltd. All rights reserved.

After

At vero eos et accusamus et iusto odio dignissimos ducimus

At vero eos et accusamus et iusto odio dignissimos ducimus

© 2021 Zenva Pty Ltd. All rights reserved.

In the previous lessons we have coded and styled a company website. In this lesson, we finally add the CSS to make our website responsive. To do this, we will continue to add **CSS** styles to our **stylesheet** (style.css) file to adapt our layout depending on the size of our screen display (for example, horizontal menu on a computer and vertical menu on a phone).

Responsive Design

Responsive design is important because when we develop our website or app we don't know exactly what device (computer, tablet, phone) it will be viewed on. For this reason, we need to design our websites to respond and adapt in accordance with how it is being viewed and used. In this course, we used **HTML** to define the structure of our company website and added **CSS classes** to allow for styling our website from an external **stylesheet** (style.css) file.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" />
    <meta name="description" content="This is what Google will show." />
    <link rel="icon" href="favicon.ico" />
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <div class="nav">
      <ul>
        <li>Home</li>
        <li>Services</li>
        <li>Support</li>
        <li>Contact</li>
      </ul>
    </div>
    <div class="wrapper">
      <div class="image-area">
        <div class="image-text centered-container">
          Learn coding at Zenva.com
        </div>
      </div>
      <div class="intro-text centered-container">
        At vero eos et accusamus et iusto odio dignissimos ducimus qui
        blanditiis praesentium voluptatum deleniti atque
      </div>
      <div class="services centered-container">
        <div class="single-service">
          <img src="Images/ZenvaBanner.png" />
          <p>At vero eos et accusamus et iusto odio dignissimos ducimus</p>
        </div>
        <div class="single-service">
          <img src="Images/ZenvaBanner.png" />
          <p>At vero eos et accusamus et iusto odio dignissimos ducimus</p>
        </div>
        <div class="single-service">
          <img src="Images/ZenvaBanner.png" />
          <p>At vero eos et accusamus et iusto odio dignissimos ducimus</p>
        </div>
        <div class="single-service">
```

© 2021 Zenva Pty Ltd. All rights reserved.

```
          <img src="Images/ZenvaBanner.png" />
          <p>At vero eos et accusamus et iusto odio dignissimos ducimus</p>
        </div>
      </div>
    </div>
  </body>
</html>
```

A key component of Responsive Design is **CSS Media Queries**. Media queries are useful when you want to modify your site or app depending on a device's general type or specific characteristics such as screen resolution or browser display width. For example, a media query can shrink the font size on small devices, increase the padding between paragraphs when a page is viewed in portrait mode, or bump up the size of buttons on touchscreens.

Understanding Media Queries

The **CSS Media Query** gives you a way to apply CSS **only** when the **viewport** (browser window or device screen) matches a rule that you specify. In our example we want to have one layout when the viewport is narrower than 700 pixels and a different layout when the viewport is wider than 700 pixels (often referred to as the breakpoint). To code a media query within a **CSS stylesheet** start a new line with **@media**.

```
@media media-type and (media-test-expression) {
  /* CSS rules go here */
}
```

Media Query Parts	Description
@media	Tells CSS we are using a media query
media-type	The **media type** tells the browser what kind of media **(print** or **screen)** this code applies to
(media-test-expression)	Inside parentheses we place our **media test expression** which evaluates to **true** or **false**. For **screen,** we often check if the **viewport** (browser window and device screen) is **wider** (min-width) or **narrower** (max-width) than a certain pixel value. For example, (min-width: 700px) tests if the viewport is wider than 700 pixels
{ /* CSS rules go here */ }	The CSS rules that will be applied if the **media-type** matches and the **media-test-expression** evaluates to **true**. These styles will override any other styles previously set in CSS

Applying Media Queries

To make our website responsive we will be adding a media query, **@media screen and (min-width: 700px)**, to change our website layout if the **viewport** is **wider** than **700 pixels**.

```
body {
  background-color: white;
  color: black;
  font-size: 16px;
  font-family: Arial, Helvetica, sans-serif;
```

© 2021 Zenva Pty Ltd. All rights reserved.

```
  padding: 0px;
  margin: 0px;
}

.nav {
  background-color: cadetblue;
  color: white;
}

.nav ul {
  margin: 0px;
  padding: 0px;
  list-style-type: none;
  font-size: 0px;
}

.nav ul li {
  padding: 16px;
  text-align: center;
  font-size: 1.2rem;
  border-bottom: 1px solid black;
  transition: background-color 0.2s;
}

.nav ul li:hover {
  background-color: black;
}

.image-area {
  background-image: url('Images/Background.png');
  background-position: center center;
  background-size: cover;
  padding-top: 200px;
  padding-bottom: 200px;
}

.centered-container {
  max-width: 1020px;
  margin: 0px auto;
  padding: 0px 16px;
}

.image-text {
  text-align: center;
  background-color: black;
  color: white;
  font-size: 2.5rem;
}

.intro-text {
  color: grey;
  font-size: 1.5rem;
  text-align: center;
  padding-top: 50px;
  padding-bottom: 50px;
```

© 2021 Zenva Pty Ltd. All rights reserved.

```
}

.single-service {
  text-align: center;
  margin-bottom: 32px;
}

.single-service p {
  padding: 0px 16px;
}

@media screen and (min-width: 700px) {
  .nav ul {
    text-align: right;
  }

  .nav ul li {
    display: inline-block;
    border-bottom: 0px;
    border-left: 1px solid black;
    text-align: left;
  }

  .single-service {
    width: 50%;
    float: left;
  }
}
```

CSS Property and Value	Description of Web Page Effect
text-align: right;	Aligns the Top Menu (** ** inside **div** with **class** of “nav”) to the right
display: inline-block;	Changes the display of each Menu Item (** **) to **align horizontally** (side-by-side) instead of the default vertical list
border-bottom: 0px;	Removes the **bottom border** we added as a default website style for each Menu Item (** **) when the Menu was displayed vertically
border-left: 1px solid black;	Adds a one pixel solid black **border** to the **left** of each Menu Item (** **) creating a separation (button look) to our Menu when it is displayed horizontally
text-align: left;	Aligns the contents of each Menu Item (** **) to the left
width: 50%;	Changes the **width** of each **Service** container (**div** with **class** of “single-service”) and all its contents to be 50% of its default value
float: left;	Floating an element allows it to stack up against one another. To **float** to the **left** means that all the elements (**div** with **class** of “single-service” in this example) will move as far as it can to the left until it touches the edge of its containing box or another floated element. The result is these elements stack up against each other

© 2021 Zenva Pty Ltd. All rights reserved.

Congratulations on creating a Responsive Company Website!

© 2021 Zenva Pty Ltd. All rights reserved.