

Table of Contents

Introduction.....	2
Hour 1 – Some Background Knowledge.....	3
The Very Basics of HTML.....	3
The Very Basics of CSS.....	5
The Very Basics of the Bootstrap Framework.....	8
So What’s Bootstrap Studio?.....	10
Hour 2 – Getting Started.....	11
The User Interface.....	11
Adding Content.....	12
Basic Styling.....	14
The Generated Code.....	15
Margin and Padding.....	15
Hour 3 – The Navbar.....	18
Adding the Navbar.....	18
Add a Drop Down List.....	19
Navbar Options.....	19
Adding More Pages.....	20
Testing.....	21
Hour 4 – Four Useful Components.....	22
Carousel.....	22
Gallery.....	24
Article List.....	25
Google Maps.....	26
Hour 5 – Rows and Column.....	28
Rows and Columns Explained.....	28
Example – About Me.....	29
Footer.....	32
Hour 6 – Themes and Custom Styles.....	34
Using Themes.....	34
Custom Styles.....	35
Responsive Styles.....	37
Hour 7 – More About Styles.....	39
Defining Colors.....	39
Overriding.....	41
Hover.....	43
Changing the Menu Icon.....	44
Hour 8 – Flexbox.....	45
Flexbox Explained.....	45
Example – Arranging Icons in an Article List.....	46
Example – Pushing the Footer to the Bottom of the Page.....	47
Hour 9 – Custom Components.....	48
Adding Custom Components to the Library.....	48
Custom Code.....	49
Example – Use Custom Code to Add Inline SVG.....	49
Hour 10 – Forms.....	52
Smart Forms.....	52
Hour 11 – (Optional) Quick Introduction to Coding.....	55
Custom Functionality Using a Little Javascript.....	55
Resetting the Form.....	57
Debugging.....	58
Extra Example: Interactive SVG.....	59
Hour 12 – Publishing, Exporting and Final Thoughts.....	63
Publishing via Bootstrap Studio.....	63
Exporting.....	64
Class Project and Final Thoughts.....	65
Index.....	66

Introduction

Hi there and welcome to this course. In this course we are going to learn all about the desktop software named Bootstrap Studio. Bootstrap Studio is a drag-and-drop front-end website designer. We can use it to easily create beautiful and responsive websites without the need to write any code.

In this course we won't go over every single detail, but rather we will learn a lot the fundamentals so that, by the time you finish this course, you could easily find and figure out any information you need. People who are more result-oriented might be tempted to speed things up a bit by skipping the explanation of fundamentals, which is perfectly fine. After all, you know your own schedule and priorities better than I do! But with a better understanding of fundamentals you will more easily be able tailor and customize your website to all sorts of needs.

This course is structured into 12 digestible bite-sized chapters, starting simply and getting progressively more advanced. As a rule of thumb each chapter might take about an hour to complete, but of course each individual is different, so it's best to study at your own pace. If you are a slow and steady learner like me it might take a little longer, and if you are one of those learners who just zoom through it you'll be a bit faster.

How you follow the course is completely up to you. I recommend you follow it in the given order and I also recommend that you follow along with all the examples to get some hands-on experience. When you follow along it's best not to copy me precisely, but use your own creativity and ingenuity to give your own flare to things. Experimenting and playing around with things is a great way to learn new things. When your website is done or partly done why not share it as a project ? I'd be really interested to see what you've come up with.

If you already have a bit of experience in web-development you will probably be able to skip some of the chapters. Whatever your background though, I recommend you take the time to make sure you understand things well enough before moving on to the next chapter. And don't be shy about occasionally going back to revise something.

In the course we will start by going over very briefly the basics of html and css. Next we will get acquainted with the user interface, and learn to add some simple content and formatting. After that will learn to add some more advanced components like the Navbar, the Carousel and more. We will learn about rows and columns. We will also learn about custom styling. After that we learn about flexbox. We will learn to create our own custom components and we will learn to add interactive forms. Finally we will learn to publish our website on a free Bootstrap Studio domain.

To follow this course first make sure you have Bootstrap Studio installed on your computer. You can buy the software from this website: <https://bootstrapstudio.io/>. If you would like to try the software out before buying it there's a browser-based demo. There's also a 30 day money back guarantee.

This course is designed to be accessible to people who have no web design experience, nor any experience coding! If you already have some experience, that's great and you will probably be able to speed through the course. But even if you are a complete beginner this course is accessible for you. You just need a little patience and persistence and you will be making beautiful responsive websites in no time!

Hour 1 – Some Background Knowledge

In order to learn Bootstrap Studio the most efficient route is to start by making sure you understand the very basics HTML / CSS first. Understanding HTML / CSS will make it that much easier to learn Bootstrap Studio. But don't worry: there's no need to know this stuff in detail. Just basic understanding will help save you a lot of head-scratching later on.

This chapter will cover everything that a non-techie needs to know to get a flying start learning Bootstrap Studio. If you already know something about HTML / CSS then by all means skip this chapter and head on to the next. If you are a complete beginner though, then let's get started!

The Very Basics of HTML

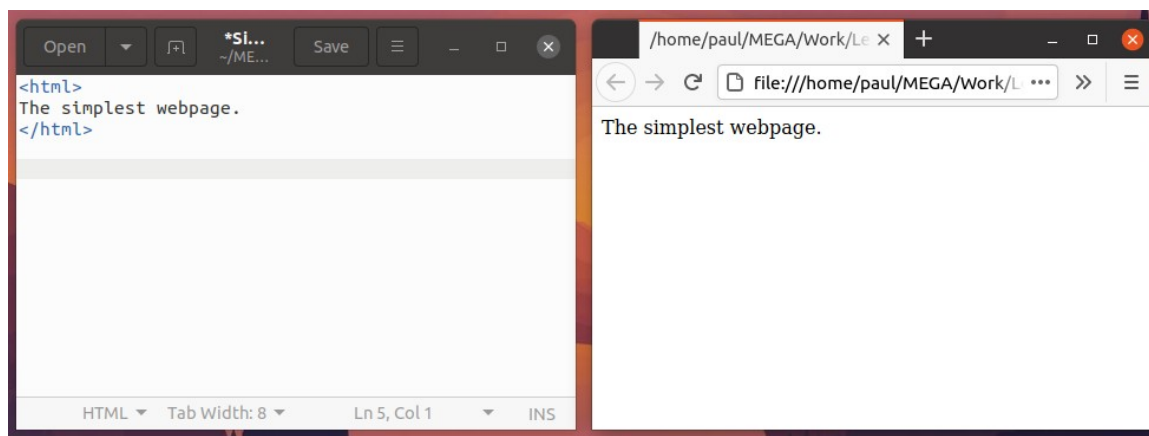
Keypoint: "It's just a simple text file containing special instructions for browsers."

HTML stands for 'Hyper-Text Markup Language'. An HTML file is simply a plain text file which can easily be written by a human being, and can also easily be understood by any web-browser. To create a simple HTML file follow the next couple of steps:

- Start up a simple text editor, and create a new file. For example, on windows you could use notepad.
- Add the following text:

```
<html>
The simplest webpage.
</html>
```

- Save the file as 'SimplePage.html' in a convenient folder on your hard drive.
- Find the file and open it up with your favorite web-browser. Tile the text-editor and the web-browser side by side on the monitor. You should get a result similar to this.



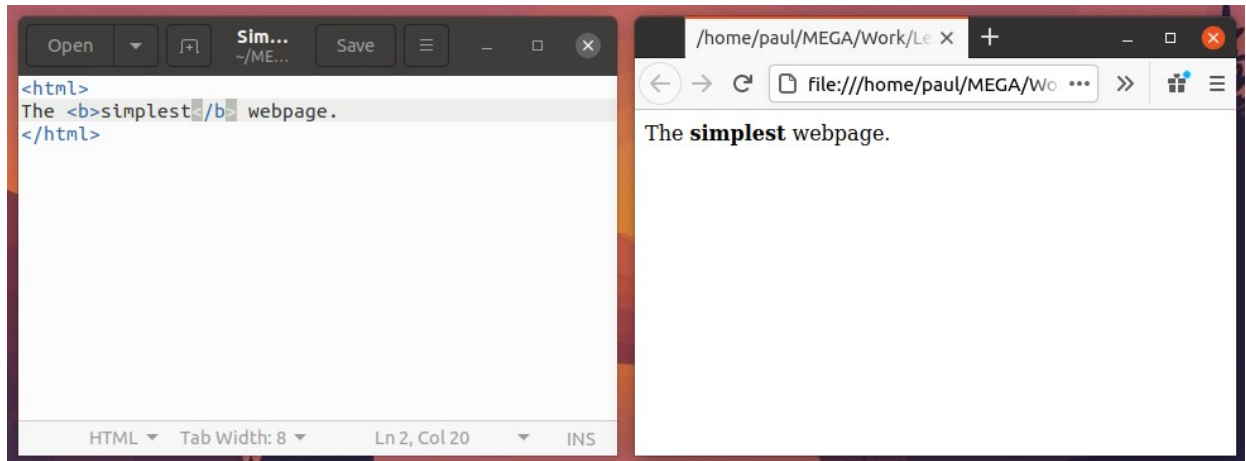
My text editor is Gedit (Linux)

My browser is firefox

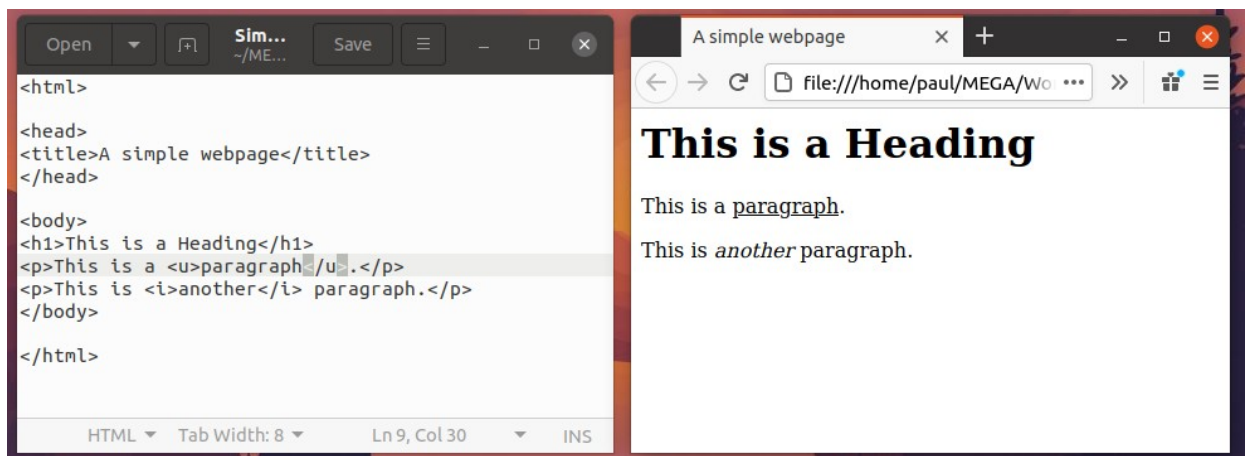
- You have probably noticed a few interesting things:
 - The file starts with the `<html>` and ends with the tag `</html>`. These are examples of tags.
 - `<html>` is a starting tag

Bootstrap Studio in 12 Hours

- `</html>` is an ending tag
- The two tags are basically telling the browser that everything in between is HTML code.
 - The text in the middle appears on the browser, but the tags do not appear.
- Next modify the text by adding another pair of tags as shown below. Then refresh the browser. Can you see what the tags ` ... ` have done?



- That's right! ` ... ` makes all the text between the two tags bold.
- Next modify the text file as shown, save the file, and refresh the browser. See which tags you can identify. Can you guess what they do?



- Got them all? Here is a summary:
 - `<head> ... </head>` provides information about the web-page
 - `<title> ... </title>` gives the title of the page. In this case you can see that the title on the tab of the browser has changed.
 - `<body> ... </body>` shows that everything between these two tags is part of the body of the web-page. The body contains the content of a web-page.
 - `<h1> ... </h1>` signifies that the text between these tags is a heading.
 - `<p> ... </p>` signifies that the text is a paragraph.
 - `<u> ... </u>` shows that the text should be underlined.
 - `<i> ... </i>` shows that the text should be in italics.

And that's basically all you need to know about HTML for now. Pretty simple, right? In fact there are of course a lot more tags, and you can do all sorts of things like add pictures and web-links and change the background etc... But using Bootstrap Studio you won't need to know the specifics of any of that. If you have an inquisitive mind and are curious to learn more there's a lot to find online. The website w3schools (<https://www.w3schools.com/html/>) is particularly useful for learning more about HTML.

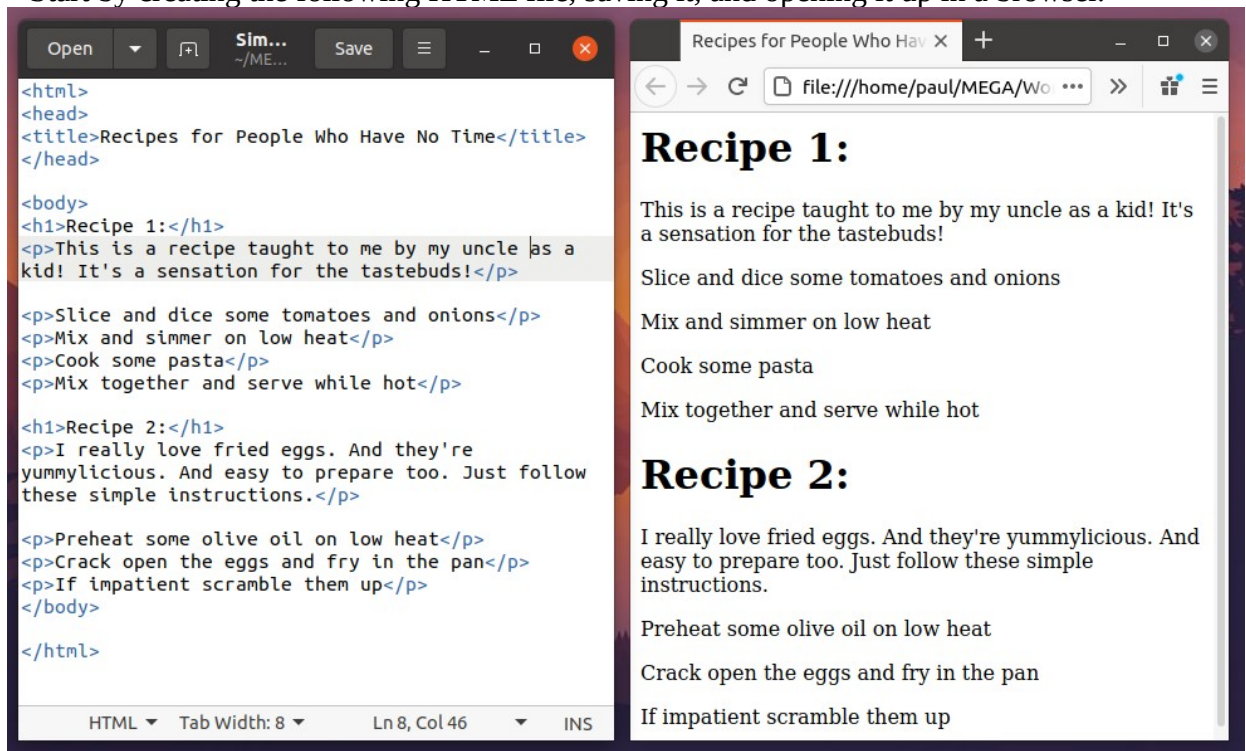
The Very Basics of CSS

Using HTML it is possible to add a lot of styling to a web-page. But this can be quite inefficient! Most websites consist of multiple pages, many paragraphs and many headings. So if you were to set the style for each individual page, each individual heading each individual paragraph etcetera etcetera it would be quite time consuming. Not to mention that if you (or your client) wanted a change you'd have to go back and individually change the styling for each and every element!

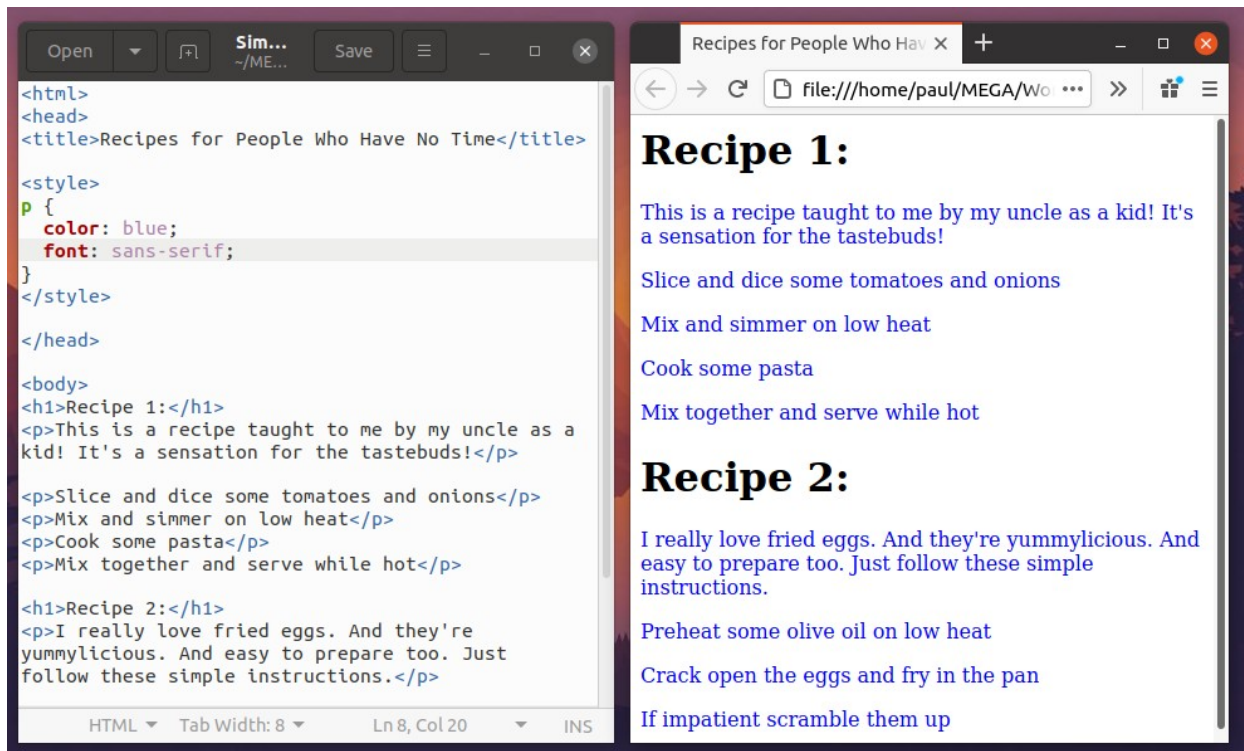
Keypoint: "HTML is for content. CSS is for styling."

So that's where CSS comes in. CSS stands for Cascading Style Sheets, and essentially it's nothing more than a way to set the styling for all elements of a particular style in one go. And if you need to make a change, you can easily go back and you only need to change it once!

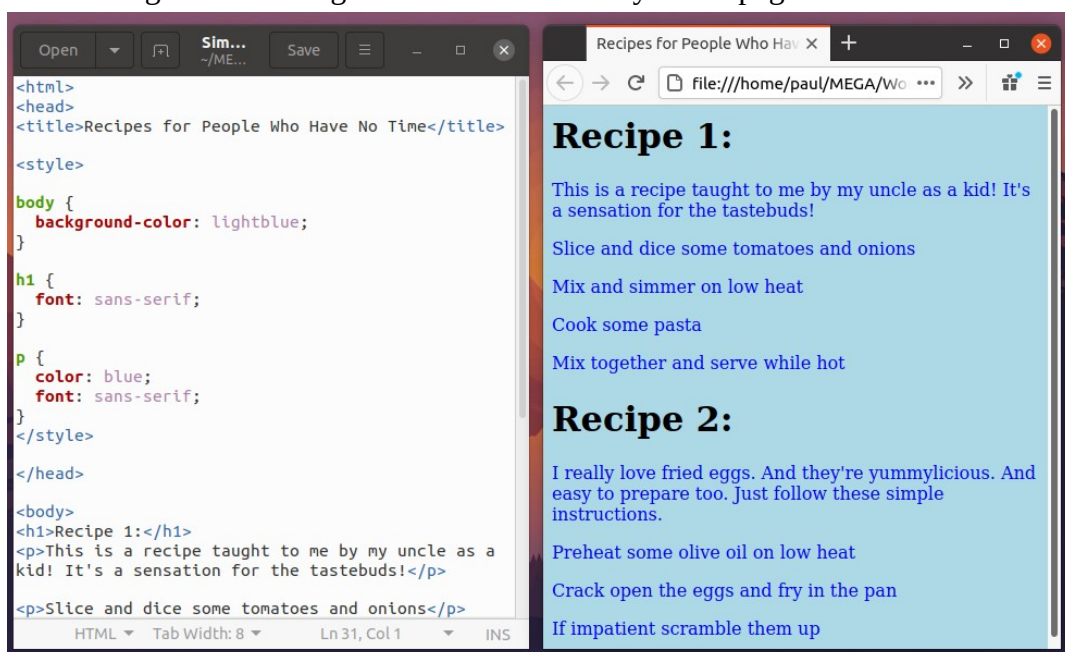
- Start by creating the following HTML file, saving it, and opening it up in a browser.



- We learn a few things:
 - First of all we learn that I'm a terrible cook! But that's not really that relevant right now.
 - Secondly we see that it was quite easy to set this up, but the styling still leaves a lot to be desired.
- Let's modify this by adding a `<style> ... </style>` section to the head of the document.



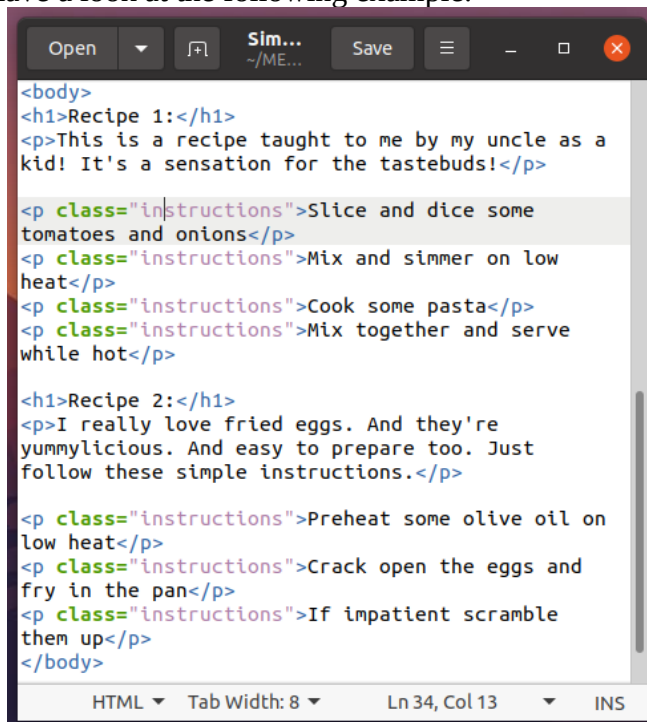
- As you can see we have added a section to the head of the document using the pair of tags `<style> ... </style>`
 - The letter p tells the browser that the style should be applied to all the paragraphs in the body of the page.
 - Everything between the curly brackets are instructions to the browser about the styling.
 - In the example above the text in all the paragraphs is set to blue and the font is set to sans-serif.
 - Usually the style is specified in a separate file, but to keep things simple
- We can also set the styling of the other elements. In the example below we set the font of the heading and the background color of the body of the page as well.



The problem is, of course, that you don't necessarily want *all* paragraphs to be the same. Perhaps you would like different types of paragraph. In the example it would be great if we had a special styling for the instructions of each recipe so that it would be clearly different from the introduction text.

To do this we add a class attribute to the relevant tags. Then we modify the CSS code to create a specific styling for those tags which have this class attribute. You can think of a class as a kind of identifier to let the browser know what kind of styling to give to which element. You can give one or more classes to any one tag. If a tag has no class the default styling will be applied.

Let's have a look at the following example:



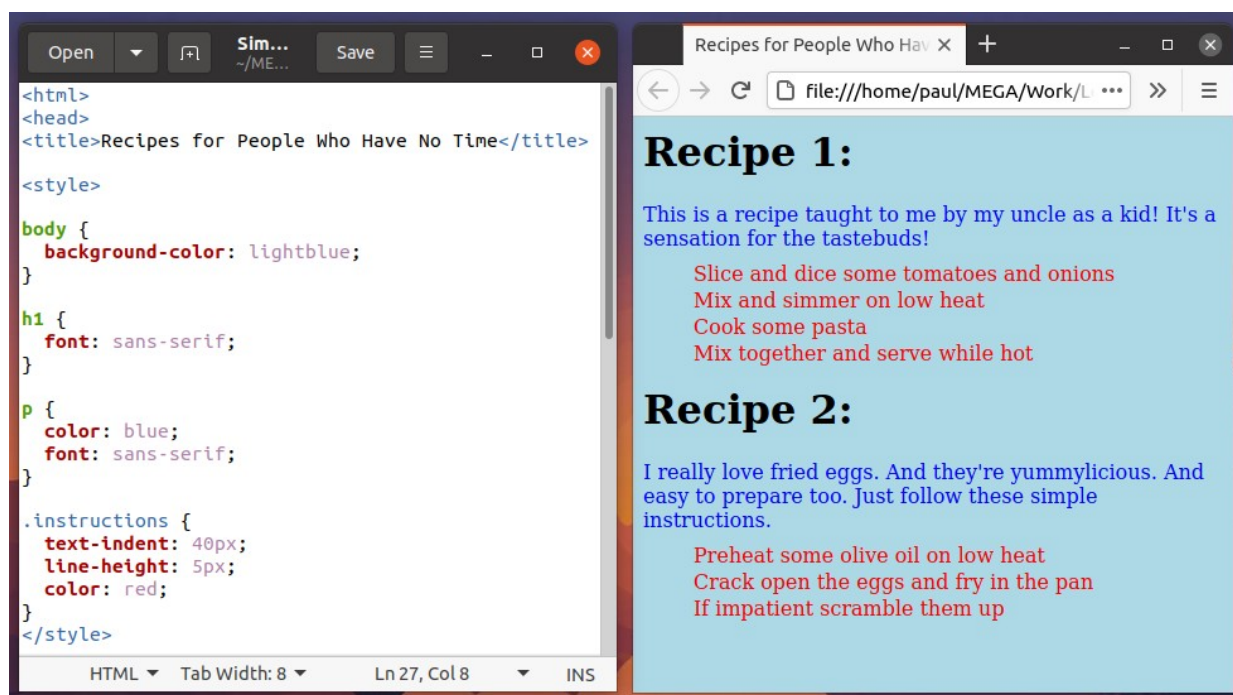
```
<body>
<h1>Recipe 1:</h1>
<p>This is a recipe taught to me by my uncle as a
kid! It's a sensation for the tastebuds!</p>

<p class="instructions">Slice and dice some
tomatoes and onions</p>
<p class="instructions">Mix and simmer on low
heat</p>
<p class="instructions">Cook some pasta</p>
<p class="instructions">Mix together and serve
while hot</p>

<h1>Recipe 2:</h1>
<p>I really love fried eggs. And they're
yummylicious. And easy to prepare too. Just
follow these simple instructions.</p>

<p class="instructions">Preheat some olive oil on
low heat</p>
<p class="instructions">Crack open the eggs and
fry in the pan</p>
<p class="instructions">If impatient scramble
them up</p>
</body>
```

- First we modify the html to add a class named instructions to each paragraph containing a recipe instruction. Do this by adding the text `class="instructions"` to the tag.
- Next we add a specific style entry to style any element with the instructions class. As you can see from the example below, we have added
 - Text-indent: 40px means that all text with the instructions attribute are indented to the right by 40 pixels.
 - Line-height: 5px sets the line spacing between these paragraphs to 5 pixels.
 - Color: red sets the text color to red.



And that's really all there is to it. Some simple CSS code in the style element in the head of the document is a simple way to add a consistent yet elaborate styling to your page. And more importantly, it's easy to modify and update the styling later.

The Very Basics of the Bootstrap Framework

For a long time HTML and CSS was more than enough to create pretty good websites. But that all changed when phones and tablets became commonplace. A website that's clear and easy-to-use on a monitor with keyboard and mouse might be terrible when viewed on a small touch screen. That's when responsive websites started to become popular.

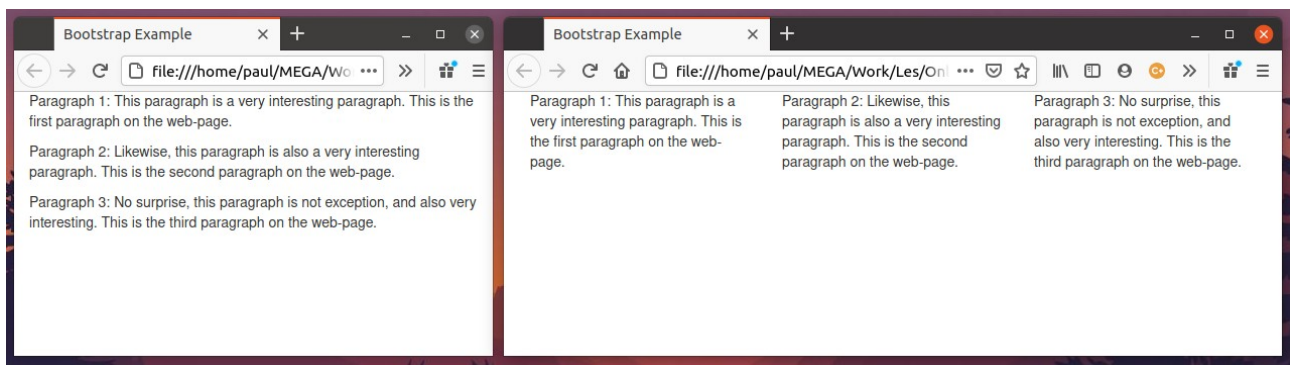
With responsive website the design of the website changes depending on what type of device you are using. This way you can create a single website that is clear and easy to navigate no matter what type of device you are using.

Writing the computer code to make websites responsive can be a tricky thing though. Essentially that's what the Bootstrap *framework* is. The Bootstrap framework (not to be confused with Bootstrap Studio) is a free and open-source *set of tools* which helps web developers to easily create responsive websites. Using the Bootstrap framework it is not necessary for each individual developer to reinvent the wheel and create their own set of tools from scratch!

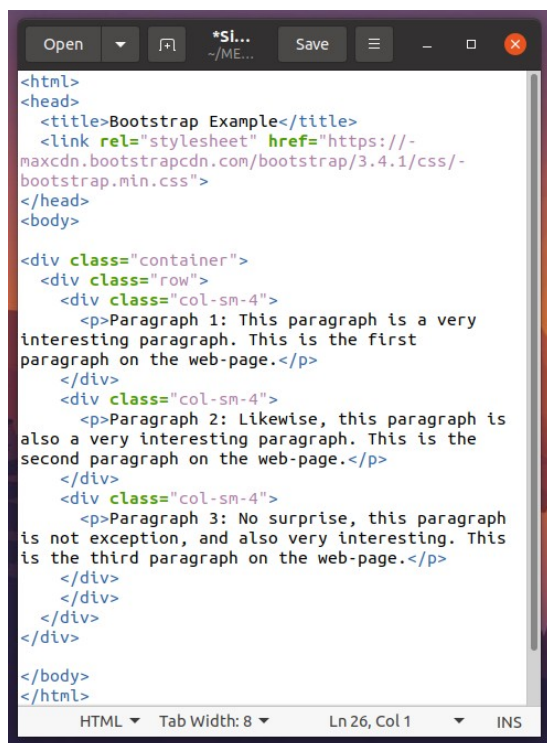
Keypoint: "The Bootstrap Framework is like a toolbox we can use to help use create 'responsive' websites"

Have a look at the following web-page. The one on the left has the paragraphs stacked vertically, a layout appropriate for a phone. The one on the right has the paragraphs arranged next to each other, a layout more suited for a wider screen.

Bootstrap Studio in 12 Hours



Would it surprise you to know that both web-pages were created by the exact same HTML file?



```
<html>
<head>
  <title>Bootstrap Example</title>
  <link rel="stylesheet" href="https://-
maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/-
bootstrap.min.css">
</head>
<body>

<div class="container">
  <div class="row">
    <div class="col-sm-4">
      <p>Paragraph 1: This paragraph is a very
interesting paragraph. This is the first
paragraph on the web-page.</p>
    </div>
    <div class="col-sm-4">
      <p>Paragraph 2: Likewise, this paragraph is
also a very interesting paragraph. This is the
second paragraph on the web-page.</p>
    </div>
    <div class="col-sm-4">
      <p>Paragraph 3: No surprise, this paragraph
is not exception, and also very interesting. This
is the third paragraph on the web-page.</p>
    </div>
  </div>
</div>

</body>
</html>
```

Right now we don't need to learn about the technical side of things. Just have a quick look at the following things.

- In the head we have a 'link' tag which references a website. This is a link to the Bootstrap Framework tools.
- We have a number of 'div' elements with various class identifiers like container, row and col-sm-4.
- We don't need to know exactly why these class identifiers work. It's enough to know that they are giving special instructions to the framework about what kind of element it is, and how the element should respond depending on what kind of device it's on. We will learn more about this later!

So What's Bootstrap Studio?

Bootstrap Studio is a drag-and-drop website editor. The user interface is used to create the layout and styling of the website in much the same way that one would use a word processor to write a report. Click and drag elements like headings, paragraphs, pictures, links, toolbars and many more into place. The Bootstrap Studio software will then create all the necessary HTML, CSS and Bootstrap code for you!

In addition to all the regular elements like headings, paragraphs, images etc... Bootstrap Studio also has a lot of interesting components like galleries, navbars, slideshows and many more. You can easily drag and drop them onto the page and customize them.

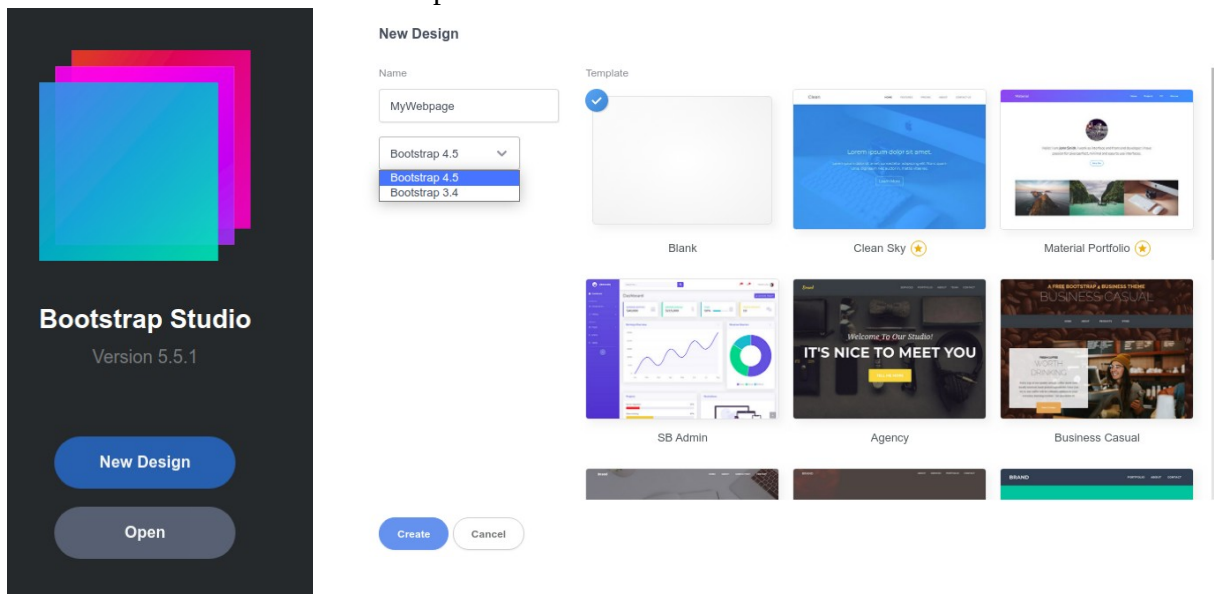
Keypoint: “Bootstrap Studio is a word-processor for websites.”

And that's all the background theory we need to get started. So no need to wait. Let's get going!

Hour 2 – Getting Started

So finally we are going to get started learning about Bootstrap Studio. Next we will get acquainted with the user interface and the learn to create a simple page. But first let's create our first project.

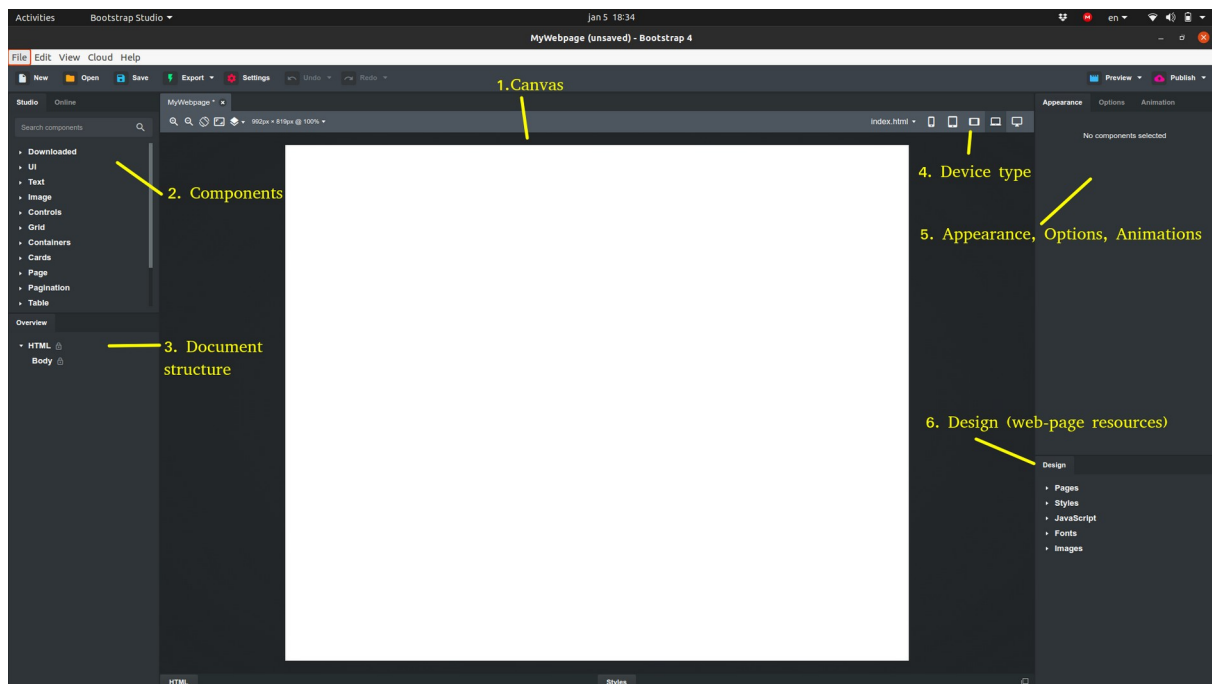
- When you open up Bootstrap Studio for the first time you have the option to create a new design or to load an existing one.
- Clicking on New Design we see that there are a lot of templates available.
- To keep things simple, let's choose an empty template. Set the name of your project, pick the latest version of the bootstrap framework and click on Create to continue.



The User Interface

- The user interface appears. (If you see a list of icons in the columns you can widen them by clicking and dragging on the column edge)
- At a glance, we see the following panels:
 1. Canvas: this is where we will see the first preview of the website
 2. Components: these are elements like heading, paragraph etc which you can drag and drop onto the canvas. All sorts of simple and elaborate components exist.
 3. Overview/Document Structure: This panel gives an overview of the components that have been added to the page.
 4. Device Type: Quickly change the size of the window to get an idea of how the website responds to different devices.
 5. Appearance, Options, Animations: With these tabs you can adjust the styling of the selected element.
 6. Design: In this tab we see the resources associated with the page. This includes the html files, css files (styles), javascript, fonts and images.

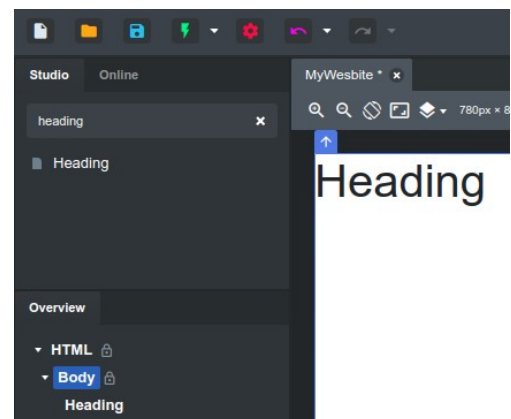
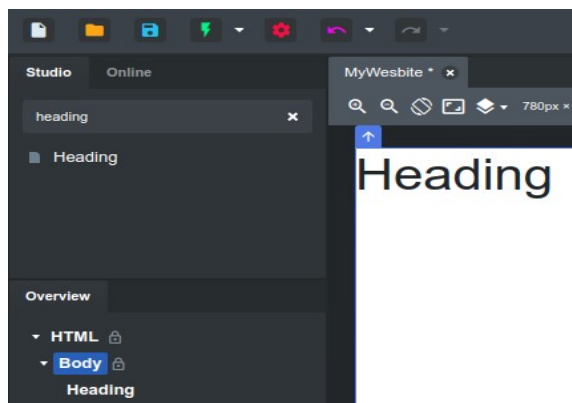
Bootstrap Studio in 12 Hours



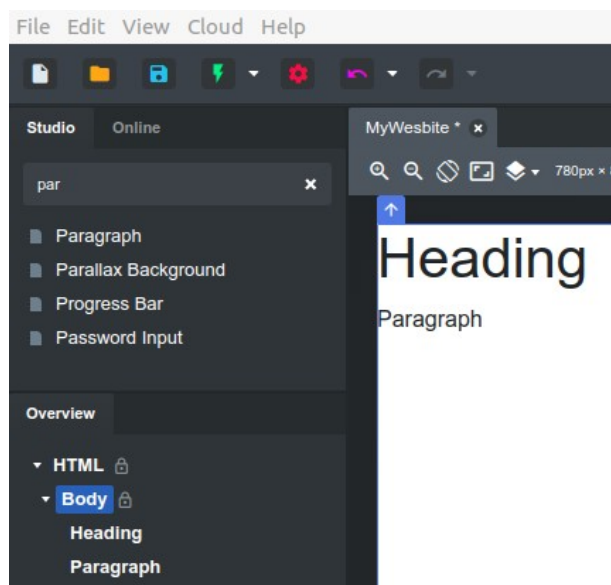
Adding Content

Next we will add some content to the page.

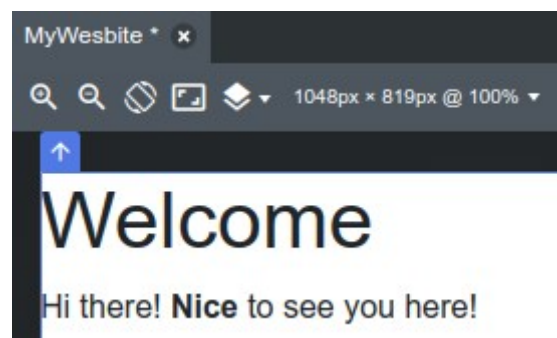
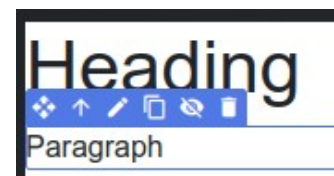
- Go to the Studio tab in the left column and search for 'heading'
- Click and drag the heading onto the canvas
- You will see that the heading appears both on the canvas and also in the overview panel in the bottom of the left column.



- Repeat the last couple of steps to add a paragraph below the heading.



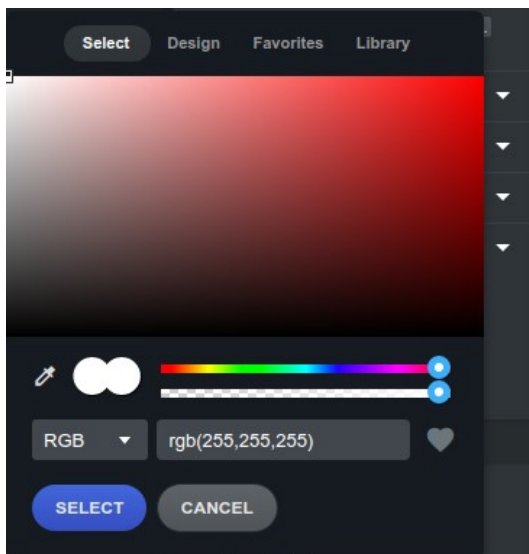
- Select the paragraph. You can select the paragraph either by clicking on the element in the overview panel or by clicking on it in the canvas.
- A number of icons appear above the paragraph:
 - Move: move the paragraph to another location.
 - Select parent: the parent of an element is the element within which it is contained. You can easily find it in the overview panel. In this case the 'Body' element is the parent of the paragraph, and the 'HTML' element is the parent of the body.
 - Edit: Modify the text in the paragraph.
 - Duplicate: makes a copy of the selected element.
 - Hide: makes the selected element invisible.
 - Delete: removes the selected element.
- Let's edit the paragraph. You can also do this by double click on the element in the canvas.
 - As you can see you can just type the text you want. Also a simple text styling bar appears.
 - Let's type the following text: "Hi there! **Nice** to see you here!"
- Let's do the same with the Heading.
 - When you select the heading you notice an extra option to set the level of the heading. Let's keep it on H1 for now.
 - Change the heading to: "Welcome"



Basic Styling

Now that we know how to add some basic elements, let's also learn to add some basic styling.

- Select a component and go to the 'Appearance' panel. We see the following:
 - In the top of the panel we see the currently selected element in blue, as well as all its parenting elements.
 - (Be careful: the styling of the parent elements also effect the styling of the element itself. This can be useful, but it can also cause super confusing situations if you are not aware of it.)
 - Below that we have a drop-down menu with options "Style Attribute" and "Create Block"
 - Style Attribute let's us style the currently selected elements.
 - Create Block allows us to create styling for a new type of element.
 - Below that we have all sorts of styling options categorized under various headings, including Layout, Font, etc....
 - (Click on the triangles to open / collapse the categories)
- Select the paragraph and make sure that "Style Attribute" is selected.
- Under Font → Color we can set the color. I will set my font color to blue. You pick any color you like.
- Next we will set the background-color of the body.
 - Select the body in the Overview panel
 - Under Background → Bg Color click on the white square to select a non-standard color.
 - You can use the eye-dropper tool, input a RGB or HEX value, or use the palette.
 - RGB specifies the color by mixing the red, green and blue components of the colors. Each component has a value of between 0 and 255. Rgb(255,100,0) mixes the maximum amount of red with some green and no blue, for example.
 - HEX is just like RGB but uses hex numbers to specify the red, green and blue components. This is interesting for some coders, but if you don't know about hex-numbers then it's absolutely fine to stick to rgb only. If there's no way around it though, you can always convert RGB values to HEX using for example this website: <https://www.rgbtohex.net/>
 - I will set my background to a light beige.



The Generated Code

The website we have made will eventually be exported to html and css files. These files can be inspected quite easily.

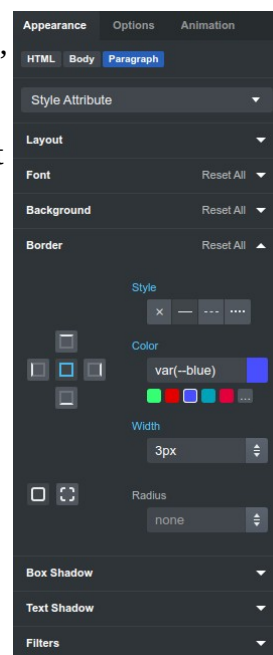
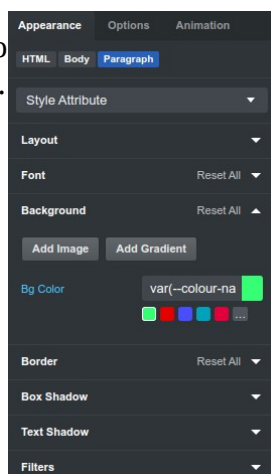
- On the bottom of the window you will see a panel containing two tabs: html and styles.
- Click and drag the top border of the panel to pull it up.
 - In the html tab you see html code of the document.
 - Click on the triangles to open and collapse different elements.
 - It's not necessary to understand too much here, but hopefully you still recognize some of the tags!
 - When adding new components to the canvas the html code will automatically updated.
 - Below the html panel there is an attribute panel. Here you can add and inspect the ID, classes, and other attributes.
 - The styles tab shows all the styling relevant for the selected element.
 - Selecting the paragraph, you see that the element inherits some styling from the parent and also the margin has been set by the bootstrap framework.
 - We also see where we have colored the font blue.

One good thing about working in this way is that it's a super easy way to learn more about HTML / CSS. If you are in a curious mood you can just drag and drop some elements into the page, or change some styling and see how it affects the code.

Margin and Padding

Margins and padding are basic HTML / CSS properties. We use margins and padding to modify the layout of the elements and their contents. Each element takes up a certain amount of space on our page. To visualize how much space we can easily add a border and a background color to each element. Let's do this for the paragraph:

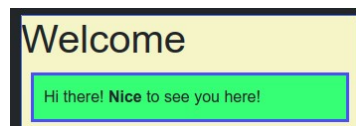
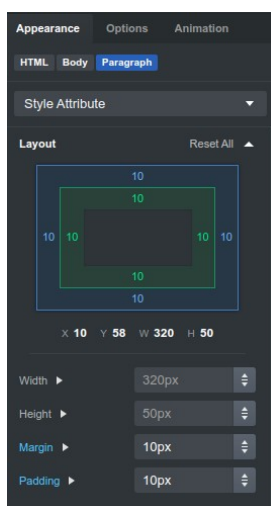
- Select the paragraph either on the canvas, or in the 'Overview' panel in the column on the left hand side.
- In the 'Appearance' panel (in the column on the right hand side), under background, we can pick a background color. I will pick green.
- Next set the border (also in the 'Appearance' panel). You can set the style of the border, the color, the width, etc... I will set the border style to solid line, and I will set the color to blue.
- As a result the border and the fill color show us how much space the element is taking up right now. We see that the element fills up the entire width of the page, and is as tall as the content.



Bootstrap Studio in 12 Hours

Now that we have a clear idea about how much space the element is taking up we can play around with the size and placing of the element:

- Make sure the paragraph is selected and go back to the 'Appearance' panel and explore the options under 'Layout'
- We see a blue / green schematic with 4 values below. Right now the default values are shown in grey.
 - The schematic gives a quick overview of the current margins and padding.
 - Width: the width is the same as the width of the page (in my case it's 340 pixels).
 - Height: in this case the default value of the height depends on the font size and how many lines the of text there are.
 - Margin: margin specifies the outer distance. In the default case for paragraphs the lower margin is set to 16 pixels.
 - Padding: padding shows the space between the border and the content of the element.
- You can change any of these values by clicking and dragging on the up / down arrows for each value, or by changing it manually.
 - For example, changing the margin to 10 pixels sets all the margins (top, bottom, left and right) to 10 pixels.
 - Setting the padding to 10 pixels adds the space between the content and the border.

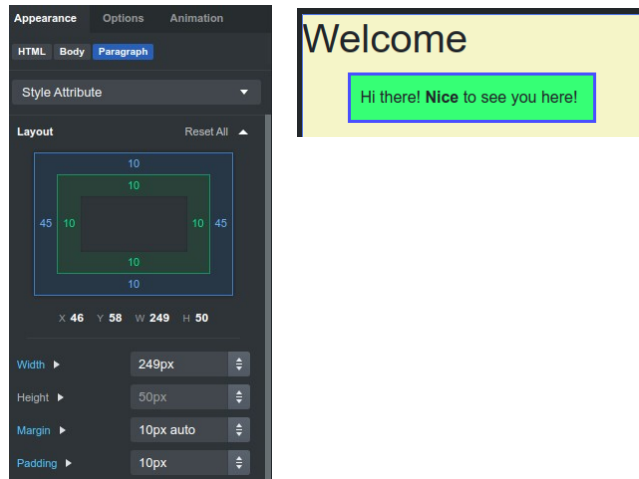


Notes:

- The width of the element needed to decrease to provide enough space for the margins.
- The height of the element needed to increase to provide enough space for the padding.
- The text in the panel changed from gray to white showing.
 - Gray shows that the default value is being used.
 - White shows that the value has been set manually.
 - If you want to go back to the default value, simply delete the value you added.
- To set specific margins and padding for the top, bottom, left and right edges we click on these triangles and we can set the values here.
- Incidentally, clicking on the triangles next to width and height allows us to set the maximum and minimum width and height of the element.
- Handy hint: if you set the margin to auto it will automatically make the left and right margins equal, so that the element will be centered in the page.

Bootstrap Studio in 12 Hours

- First decrease the width of the paragraph. We see there ends up being a larger empty space on the right than on the left.
- Set the margin to “10px auto”.
- Now the top and bottom margins are set to 10px, and the left and right margins are set equal, placing the element to the middle of the page.



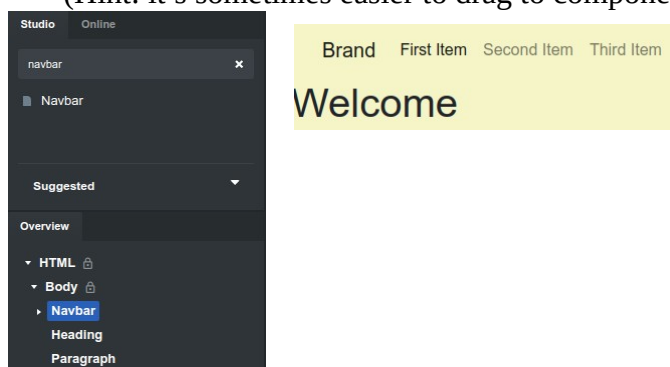
Now we have a basic understand of the way Bootstrap Studio works, we can add some basic elements and we can do some basic styling. Adding more elaborate elements is just as easy adding simple ones. In the next chapter we are going to add a Navbar. So for now take it easy and have a well-deserved break and I'll see you in the next chapter.

Hour 3 – The Navbar

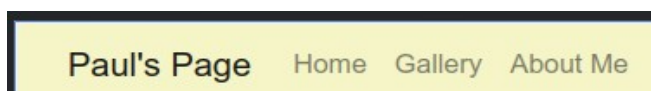
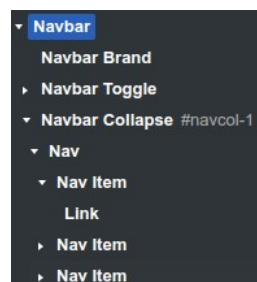
So far we have been working with very simple elements. Up next is a more complex element: the navbar. Navbar is short for navigation bar, and is usually implemented as a bar at the top of the page in which the main sections of the website is written out. Each item on the bar usually contain a link, so that the user can easily find and jump to the section they need.

Adding the Navbar

- In the components panel search for ‘Navbar’
- Click and drag the component to the top of the canvas
 - (Hint: it’s sometimes easier to drag to component directly to the Overview panel)



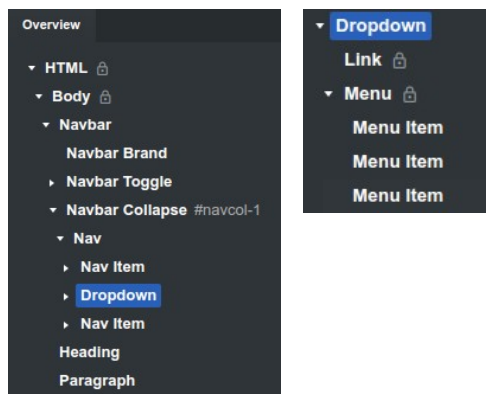
- The Navbar is significantly more complex than what we’ve used so far. Looking in the overview panel we see that the Navbar contains:
 - Navbar Brand: This is essentially the name of the website. Let’s set the text. I will name mine “Paul’s page”
 - Navbar Toggle: This contains a menu-icon. When the width of the page is small enough the Navbar show’s this icon instead of the nav items.
 - Navbar Collapse contains Nav which in turn contains the Nav Items. Each Nav Item in turn contains a link.
 - The links contain the web-address of the other pages on your website, or can even contain the address of other websites.
 - You can set each item to Active or Inactive. To avoid confusion we deactivate all of them for now.
 - We can easily add more by duplicating. (Best use the overview panel to make sure you duplicate the right element)
 - Likewise, the most straightforward way of removing Nav Items is by deleting them in the Overview panel.
 - In my case I will use all 3 Nav Items and set the text to “Home”, “Gallery” and “About Me”



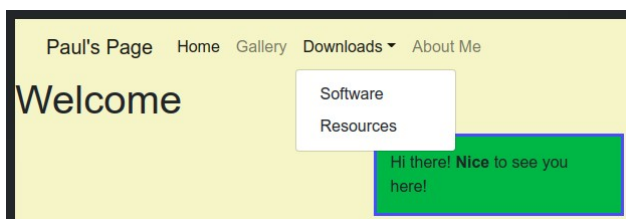
Add a Drop Down List

Sometimes it's useful to add a drop down list to the Navbar. There are a few ways to do this. One way is to add it from the component column:

- Go to the components panel and search for “dropdown”
- Click and drag the component to the overview panel, making sure it is placed between the two Nav Items
- The Dropdown contains a link and a menu, and the menu contains 3 menu items.



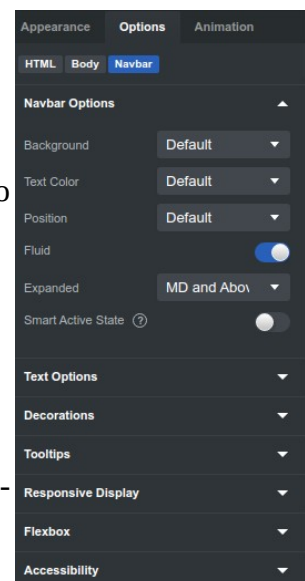
- Double click the Dropdown component in the overview panel to open / close the drop-down list on the canvas.
- Next we edit the text of the Dropdown and Menu Items. As an example I will add the text “Downloads” in my dropdown, add the text “Software” and “Resources” in the menu items and delete the third menu item.
- If you want to add a fourth menu item the easiest thing to do would be to duplicate one of the existing menu items and modify it.



Navbar Options

The Navbar comes with some specific options. Let's have a look:

- Make sure the Navbar is selected and go to the Options panel.
- In the top of the panel we have the options specific to the Navbar:
 - We can set the background and text color. We can keep it on default, or we have a choice of many different theme colors.
 - We can also set the position. The options include default, fixed to top, fixed to bottom and sticky top. I feel like setting mine to sticky top.
 - Fluid means that the width of the content of the Navbar will change gradually as the width of the window changes. If fluid is deactivated the width of the content will also change depending on the width of the window, but in discrete steps.
 - With expanded we can set when we'd rather see a menu icon or the nav items. In this case it will show the nav items for medium-



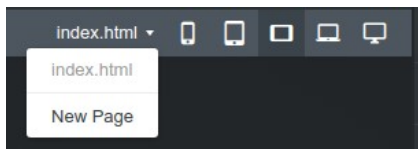
sized windows and above and will show the menu icon for screen sizes that are smaller than medium

- Finally, Smart Active State determines whether the Navbar shows which page is currently active. In other words, the navbar item linked to the current page will be set as active, with the remaining set as inactive. I will set mine on. This way the user will be able to see at a glance which page he is she is currently viewing.

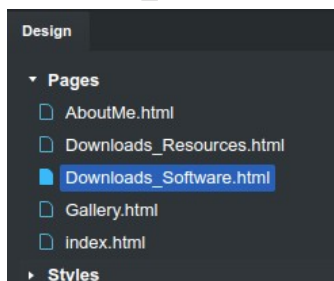
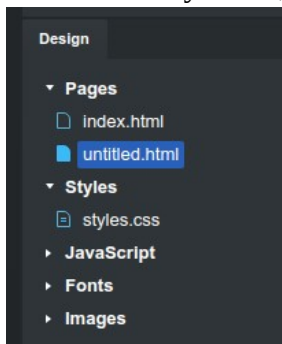
Adding More Pages

You will no doubt have noticed that none of the items we have added are linked to anything yet. Before linking the Nav and Menu Items let's first create a few more pages.

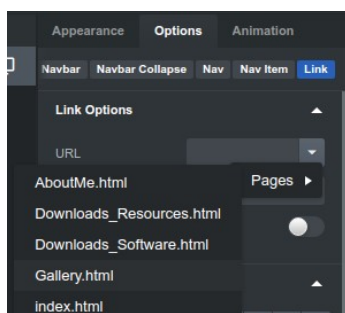
- You will see in the top right of the canvas that we are currently working on page named index.html
 - index.html is called the 'home page' and is the default page that will be shown if someone surfs to your web-address of the website without entering the specific page name.
- Click on the arrow next to "index.html" and click on "New Page"



- A new page named "untitled.html" appears. Go to the Design panel (bottom of the right column) and find the webpage under the Pages heading.
- Right-click or press F2 to rename the file. I will rename mine to AboutMe.html
- Repeat the process to create a new web-pages for each Navbar Item in your Navbar. I will create "Gallery.html", "Downloads_Software.html" and "Downloads_Resources.html".



- Next let's go back to index.html.
- Select the link in one of the Navbar Items and go to the Options tab in the column on the right hand side.
- Under the heading "Link Options" we can set the URL. We can add any web-address here, including external sites like your brother's or sister's website, or your business partner's website or whatever. In this case we want pick from among the pages we have just created.

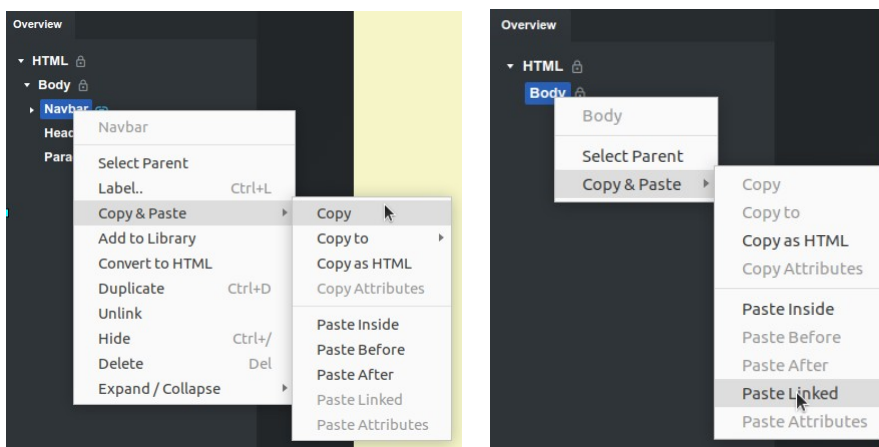


Bootstrap Studio in 12 Hours

- Let's repeat this for the other pages. In my case for index.html, AboutMe.html, Downloads_Resources.html, Downloads_Software.html and Gallery.html.

We would like to have the toolbar on the top of every page. Normally we would just copy and paste the toolbar. However, if we did that and later we decided to change it, we would need to be extra careful to remember to change all copies. This is extra work and also creates a situation where mistakes might happen. With Bootstrap Studio we use 'paste linked' to copy and paste the element, so that if we change one of the copies all the others would automatically change too.

- Select the Navbar in the Overview panel, right click, and choose Copy & Paste → Copy.
- Go to each page one by one, right-click on the canvas and choose Copy & Paste → Paste Linked.



Testing

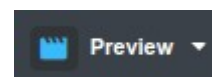
To test whether all the links are working it's best to start up a preview of the page.

- On the top right click on the preview button.
- A window pops up. Click on the toggle to enable the preview
- Click on Open in Browser to view it
- Now you can test all the links.
- You will notice that, because we

Preview Settings

Preview Port
8000 ☐ Preview is Enabled

Preview URLs



Hour 4 – Four Useful Components

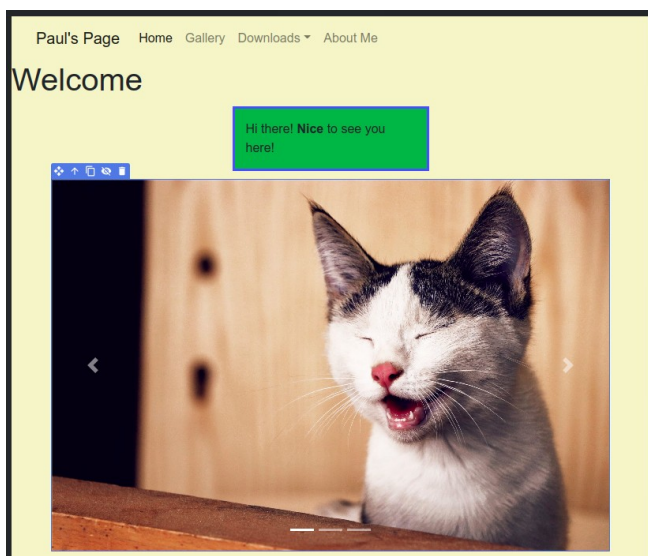
In this chapter we will learn about 3 useful components that will quickly add a lot of flash to your site.

- Carousel: a handy slide show.
- Gallery: a list of thumbnails which can be viewed in a pop-up window.
- Article list: similar to gallery, a list of items which also includes a description.

This chapter is a bit more result-oriented: the emphasis is on getting it done, rather than on explaining how or why it works. Future chapters will explain more about the why of it so that you will more easily be able to modify existing components and create you own. But for now let's have a look at how these existing components work.

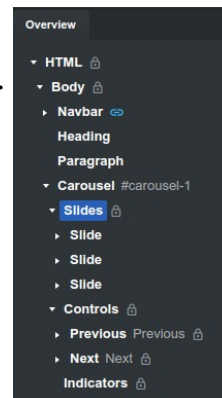
Carousel

A carousel is essentially a slideshow that can be used to cycle through a series of photos, slides or info graphics.



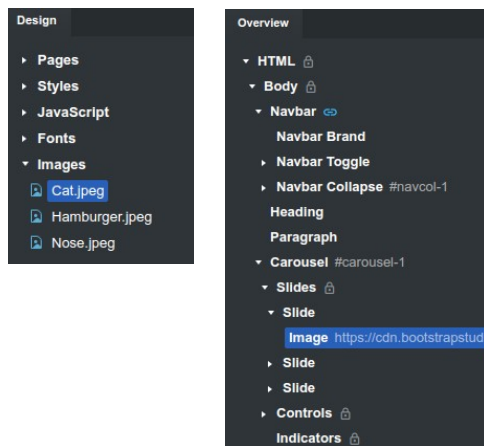
Adding a carousel in Bootstrap Studio is a simple matter.

- Find the carousel among the components and drag it to the overview panel.
- The carousel consists of slides and controls.
 - Slides contain the images. As usual you can easily add and remove slides by duplicating or deleting.
 - Controls include the next and previous buttons, as well as an indicator to illustrate which image is currently being displayed.
- Let's add some pictures to this carousel. First we need to import images. Hint: you can find some pretty nice public domain images on www.pexels.com.
- Import them into Bootstrap Studio by clicking and dragging the files into the Design panel. They will appear in under the heading "Images"

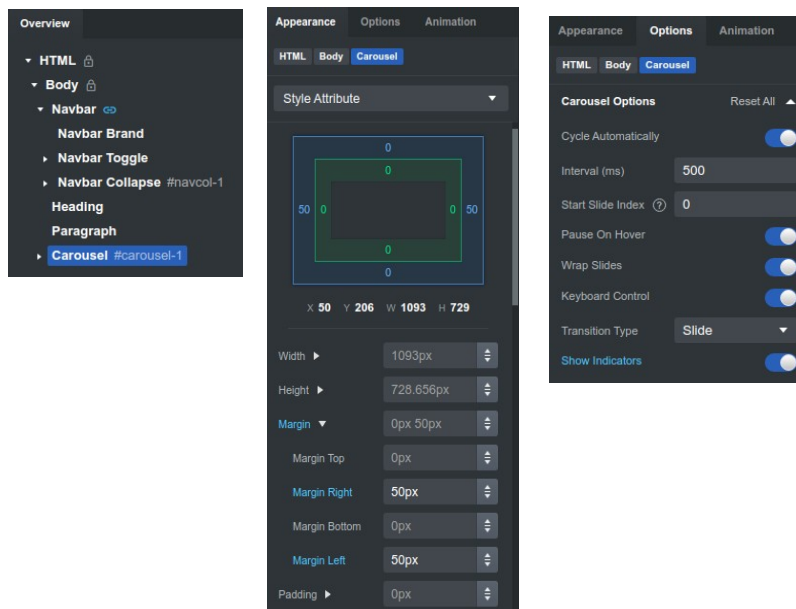


Bootstrap Studio in 12 Hours

- Select one of the images and double click. (Hint: it's easiest to use the Overview panel for this)
- A window pops up with a list of imported images. Select the image you want and press ok.

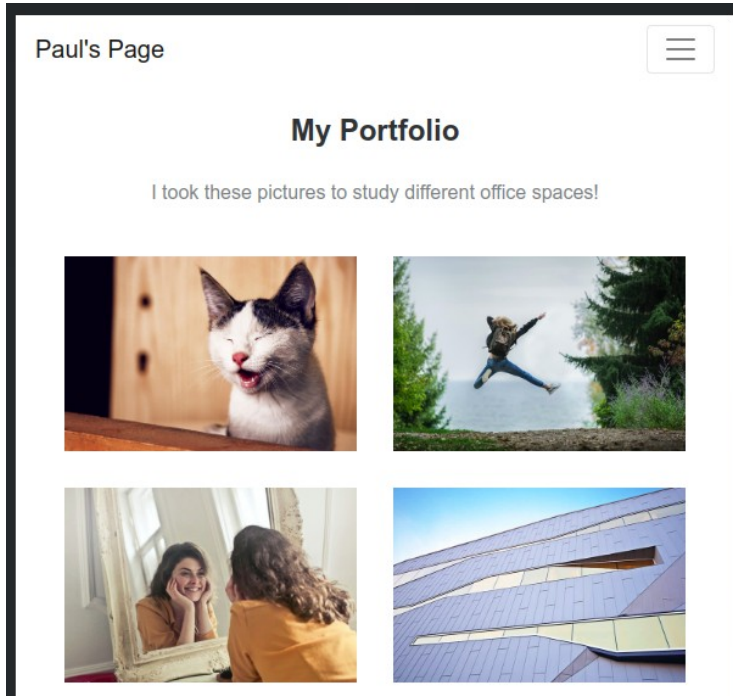


- Repeat the previous steps to add the other images too.
- The carousel automatically expands to take up the entire width of the page. To change this:
 - Select the carousel component
 - In the appearance panel set the left and right margins to 50px
- The carousel comes with some options. We can modify these options using the Options panel.
 - Here we can set all sorts of options like the interval, pause on hover, transition type etc...
 - These are pretty self-explanatory so I will save some time by not explaining them here too. It could be useful to play around with them for a bit though.



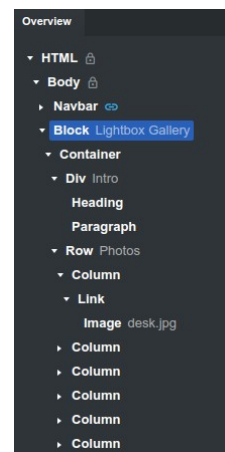
Gallery

A gallery is a useful way to show off your portfolio. It consists of a grid of images, which the user can click on to see it in a larger view.



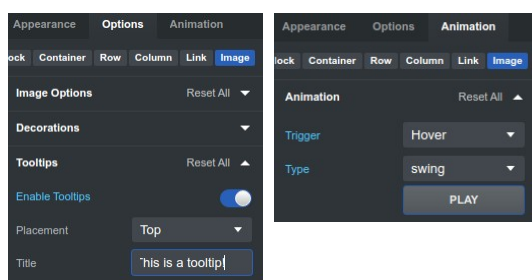
Follow these simple steps to add a gallery.

- Go to the page Gallery.html and add a 'Lighthouse Gallery' component to the page.
- Looking at the canvas we see a heading and a paragraph followed by a grid of pictures.
 - We can edit all the text in the usual way.
 - When we resize the page the layout of the grid changes.
 - When we click on any picture it opens it in a popup window.
- Looking at the Overview panel we see that the Gallery is a Block which contains a Container within it.
- The Container contains a Div and a Row.
 - In CSS a Div is short for division, and can be used to group different elements together.
 - A Container is simply a Div with the container class attribute. (Recall from the first chapter that a class attribute is simply a label or identifier that let's the browser know what styling to use for this element) We will learn more about containers in a future chapter.
 - The Row contains a number of Columns. Rows and Columns are used to create responsive grids. We will learn more about Rows and Columns in a future chapter.
 - We can add and remove Columns by duplicating and deleting them.
- To change one of the images simply double-click on it and select the image.



Bootstrap Studio in 12 Hours

- (Make sure you have imported the image you want first: just click and drag the image from your file manager to the Design panel in Bootstrap Studio)
- We can also set a tooltip for each image. Just select the image and in the Options panel go to Tooltips → Enable Tooltips.
- Finally let's add an animation.
 - Select an image and go to the Animation panel in the upper right column.
 - Under trigger select Hover. This way the animation will be triggered whenever the user hovers with the mouse over the image.
 - You can select the type of animation too. For example, swing is pretty funny.
 - You can get a preview of the animation using the play button, and you can test it in the preview.



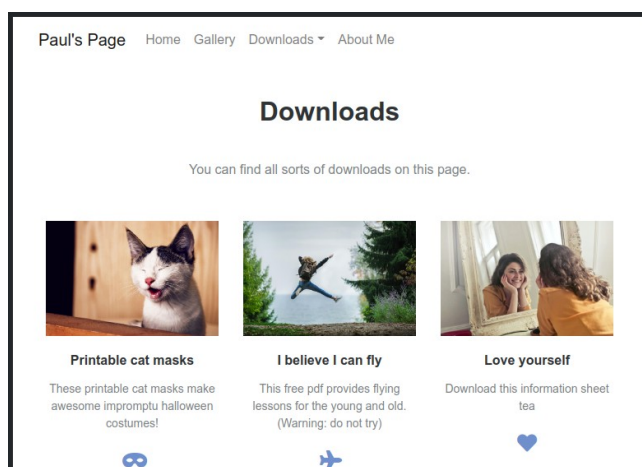
- One more thing to do is to test the page. Let's click on preview and see if it works.
- We have found one mistake: the link does not correspond to the thumbnail.
 - Let's correct this by selecting the link of the first slide.
 - Then, under options, update the URL to pick the correct image.

As you can see, this was a pretty easy way to add a very nice gallery to our website! In the next section we will be adding an Article List.

Article List

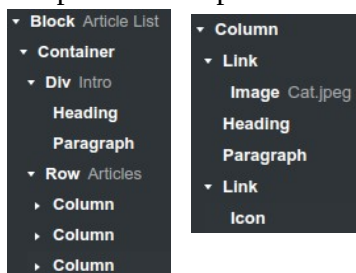
An article list is a great way to direct surfers to more specific content in your website. It combines a thumbnails, descriptions and links into a responsive grid.

As usual, just find 'Article List' in the components panel and drag-and-drop it onto the page or into the Overview panel. I will do this on the page 'Downloads_Resources.html'.



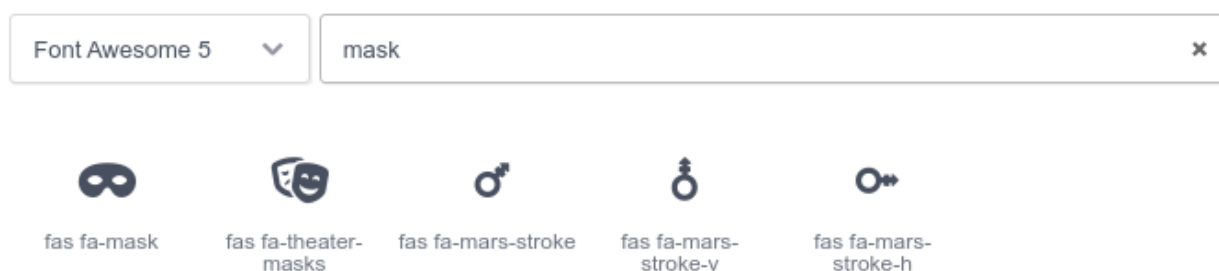
Bootstrap Studio in 12 Hours

- Just like with Gallery we get a flexible layout, whereby the grid of images rearranges depending on how wide the page is, but in this case we also have a description and a link.
- Looking in the overview panel we see that the article list consists of a Div and a Row.
 - The Div contains the heading and the introduction.
 - The Row contains Columns.
 - Each Column contains a Link with an Image, a Heading / Paragraph, and a Link with an Icon.
- To change the pictures simply double-click on it and select the picture you want (make sure the picture is imported to Bootstrap Studio first). You can also edit the text in the usual way.



- You can set the URL of the links by selecting the link and going to the Options panel.
- You can also change icons by double clicking on the element.
 - All sorts of icons are available in different categories.
 - I will set the three icons by searching for 'mask', 'plane' and 'heart' all in the category 'Font Awesome 5'

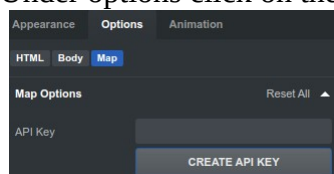
Choose an Icon



Google Maps

If you would like customers to be able to easily find you then adding a embedded google-map to you website is the way to go. To add a map you need to make an api key first though so google can identify your project. With a free api key your project can load 25000 maps per day! If you need more loads than that and you would need to pay!

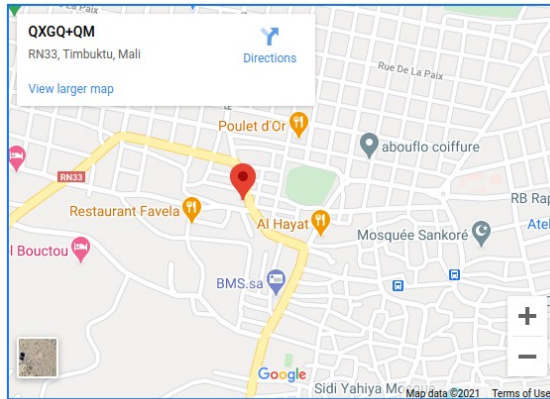
- Drag and drop 'map' to the page.
- Under options click on the button "CREATE API KEY"



- We are redirected to a website where we can log in to our gmail account.

Bootstrap Studio in 12 Hours

- I will create a project to link the api to, but you can also link it to an existing project if you have one.
- Next we can restrict the key. If you don't there's a risk someone else could use your key.
- Once created I can copy the key and paste it to the options panel.
- As you can see the map is now functional!



- Remember you can always go to google's api console to modify or remove your key.

Hour 5 – Rows and Column

The standard Bootstrap way of creating a responsive grid is to use rows and columns. The grid is said to be responsive because the layout of the grid depends on how wide the screen is.

Rows and Columns Explained

Before getting started a little explanation would be helpful here. The terms row and column are a bit misleading in this case, because it suggests that:

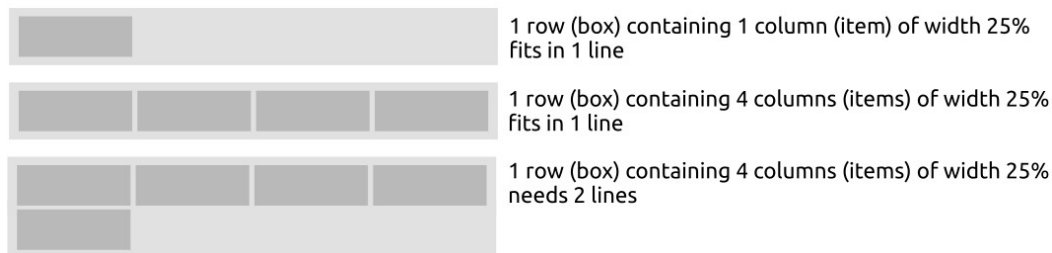
- a. the number of columns in the grid is static.
- b. Rows and columns are equal and equivalent.

This is actually not the case and it can be a little confusing to think in terms of rows and columns.

In fact, rows are more like boxes, and columns are more like the items arranged within. For the sake of this explanation only I will rather use the terms ‘box’ and ‘item’. I’ll switch back to saying ‘row’ and ‘column’ when applying the explanation in the example.

So a row is like a box containing a number of items. Each item has a certain width and prefers to be arranged horizontally from left to right. As long as there is enough space they will all be arranged inline. If there is not enough space, the later items will be bumped down to the next line.

So, for example, if you have a box containing an item which has a width of 25% of the page, it would easily fit. In fact, you could just fit 4 items four of such items, and they would all be arranged in a single line. If we would add one more such item, there would not be enough space left in the top line of the box, and so it would need to be bumped down to the next line.



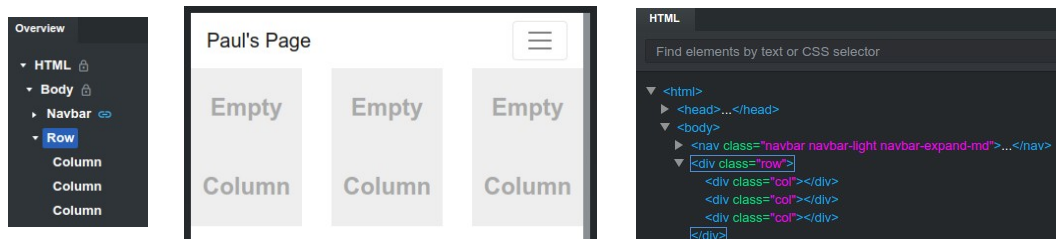
In the bootstrap framework the developer can pick how wide each item is depending on the width of the screen. So, for example, the developer might set the width of the each column to 100% for small screens, to 33% for medium-sized screens, and to 25% for large and extra large screens. That means only one would fit per line in small screens, three would fit side-by-side in medium-sized screens and four would fit side-by-side in large and extra large screens.

In the next section, we will set up a simple ‘about-me’ page as an illustrative example which will hopefully make the above explanation a little more intuitive!

Example – About Me

Now that you have a rough understanding of rows and columns, let's use an example to explain more about the details and also get some hands-on experience. In this section we will create a simple responsive grid on the About Me page.

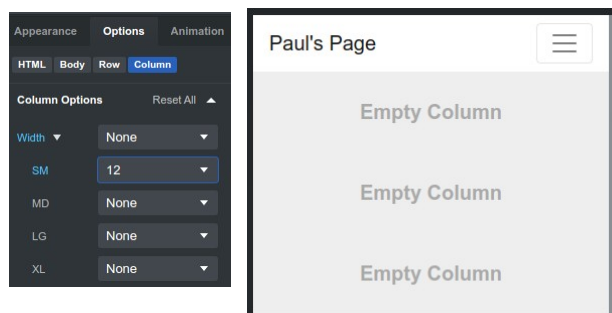
- First of all, go to AboutMe.html and we drag a row to the canvas.
- Next, let's add three columns to the row.
- We can see:
 - the three columns are arranged next to each other
 - from inspecting the generated HTML that rows and columns are simple divs with the classes row and col added



- Select the first column. Under options you can set the width of the column.
 - Setting the width to 1 sets the width to one twelfth of the available width. There is now more space available for the two standard columns, and they will expand.
 - Setting the width to 12 sets the width to 100% of the available width. This pushes the other two columns down.
 - Setting the width to 6 sets the width to 50% of the available width. This leaves only enough 50% of the width for the second and third columns. If the width of the page decreases, it pushes the third down.
 - Hint: see how changing the width of the column affects the HTML code.

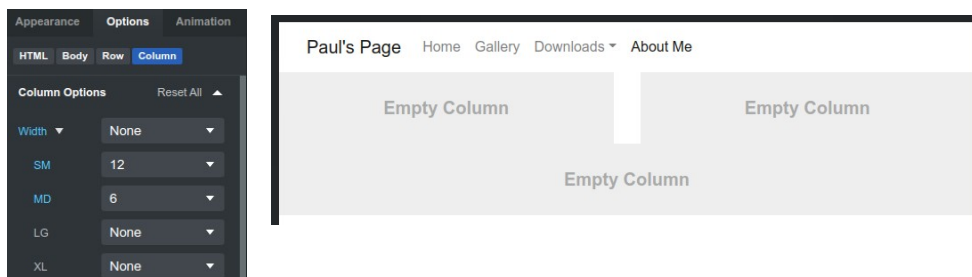


- Next we will set the width for different screen-sizes.
 - First set the column width back to none. (This will turn it back to a standard column for now)
 - Click on the triangle to expand the width options.
 - For small screens we want to arrange the columns vertically. Set the SM-width for each column to 12. (This makes each column take up the entire width for screen-sizes SM)

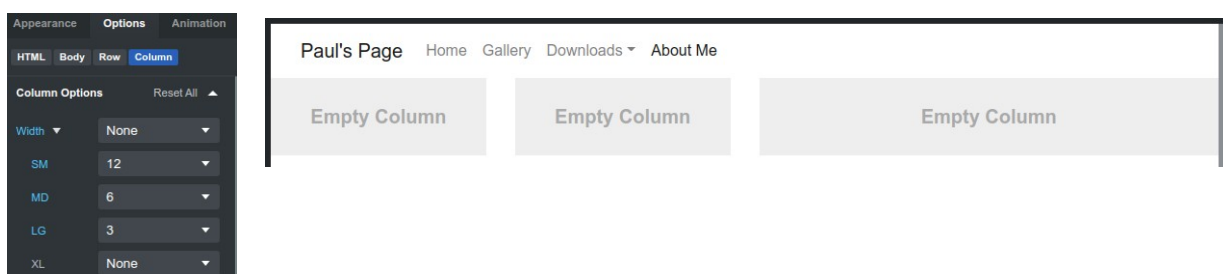


Bootstrap Studio in 12 Hours

- For medium-screen sizes we would like to arrange the first two columns side-by-side, and we would like to push the third column down. So let's set the width to 6, 6 and 12 respectively.



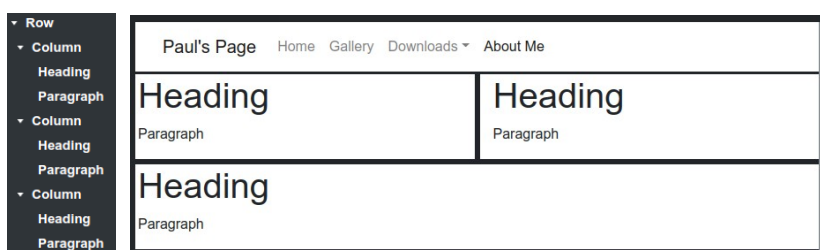
- Finally, for large and extra large screens we would like the three columns arranged next to each other, but we would like the third column to be twice as wide as the first and second. Set let's set the width to 3, 3, and 6 respectively.



- Hint: keeping the width for XL as none means that it will use the settings for LG.

So that's how to set up a basic responsive row / column. It's a good idea to play around and experiment with it a bit, to get a better feel for it. Next we will add some content and styling.

- Let's start by adding a heading and a paragraph to each column. We can edit the text later.
- Next let's add a border to each column.
 - Hint: hold down ctrl to select multiple elements at the same time.
 - As you can see the columns are so close they're touching.



- We could add some space between the columns by selecting them and then increase the margin!
- But this causes a problem. The extra space taken up by the margins pushes the 2nd column to the next line, and messes up the layout we picked in the previous section.

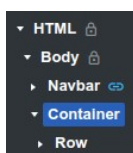
Bootstrap Studio in 12 Hours



- A better solution would be to group the content of the column into a div and add a border to the div instead. This way we can change the margin without worrying about messing up the layout.



- We would like to add some space between the edges of the page and the content of the columns. A container is a very useful element. It is simply a div element with special properties:
 - The width of the container responds to the width of the pages.
 - The container has a certain maximum width.
 - The container is always centered in the middle of the page.
- It's very easy to add the row / columns we have created into a container.
 - First add a container to the canvas.
 - Then drag and drop the row into the container.

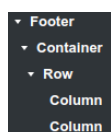


- Finally let's adjust each column to add the finishing touches!
 - First I will edit the text in the columns.
 - Hint: press shift-enter to add a new line to a paragraph.
 - I'm going to add some padding to each div.
 - I will set the border radius to 20px
 - I will also add a profile-picture to the first column instead of the text, and adjust the border radius.
 - Also, I think I will remove the border from these divs, and rather change the background instead.
 - Finally, I will set the height of each div manually to make sure the height stays the same for each.

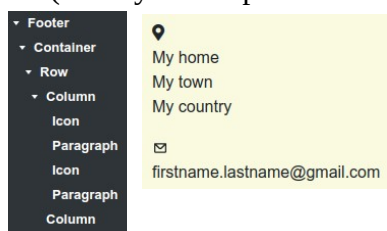
Footer

As a second example we will use rows and columns to create a simple footer. The footer will contain two columns which will always be arranged side-by-side. The second column will contain a responsive grid of it's own, which will have an arrangement that depends on the width of the page.

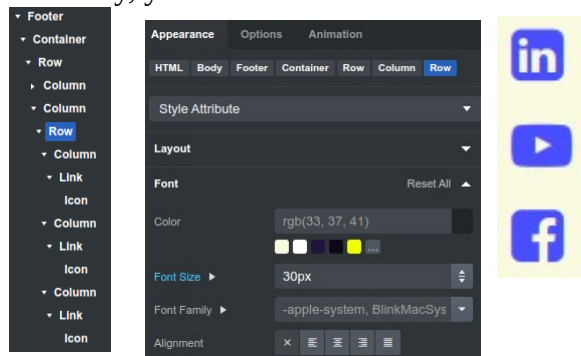
- First add the following items: a footer, add a container within the footer, a row within the container and two columns within the row.
 - A footer element is just a label. We could leave it out, but keeping it in simply makes the html code easier for humans to read.
 - The container automatically adds appropriate margins to the footer.



- Next, to the first column, let's add two icons and two paragraphs.
 - Double click on the first icon to change it to 'map' (Font Awesome 5).
 - Double click on the second icon and change it to 'mail' (CSS Icons).
 - Modify the text of the paragraphs.
(Hint: you can press shift + enter to go to a new line within a paragraph)



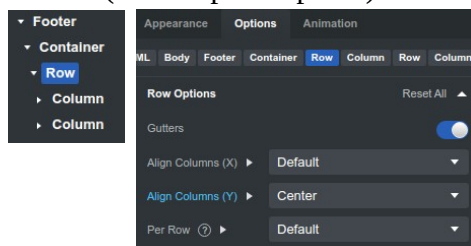
- Now let's add content to the second column.
 - First add a row and a single column. Add a link to the column and add an icon to that.
 - Delete the text in the link and drop in an icon instead.
 - Duplicate the column twice
 - We can change the icons to some social media icons like twitter, snap chat or whatever you like. I will change mine to youtube, facebook and linked-in.
 - These icons are a little small. We can enlarge them by increasing the font size.
 - A quick way to enlarge the font size is to select a common parent and set the font there.
 - In this case the nearest common parent is the row. Select it and set the font to 30px or so.
 - Finally, you can set the URL for each link in the options panel.



- Next let's set the width of the columns:

Bootstrap Studio in 12 Hours

- I want to push the social media icons a bit to the right. So I will set the width of the first column to 9.
- To make sure that the social media icons stay on the right of the page, and never get bumped down to the next line, I will set the width of the second column to 3.
- We want to stack the three social media icons vertically for small screens, and we want to arrange them side-by-side for medium and larger screens. So, for all three columns:
 - Set the default width to 12 (this means for all screen sizes, unless specified otherwise).
 - Then set the MD-width to 4 (this means for all screen-sizes medium and larger, unless specified otherwise)
- Finally we noticed that the icons on the right-hand-side end up a bit lower than the text on the left-hand-side. We can fix this by selecting the first row and setting the y-alignment to Center (in the options panel).



And so we have made a nice responsive grid to use as a footer! It's a simple matter to copy it to the other pages. Remember to use Paste Linked so that any changes will automatically be applied to all the copies. In the next chapter we are going to create some custom styling to style the footer and other elements on the page.

Hour 6 – Themes and Custom Styles

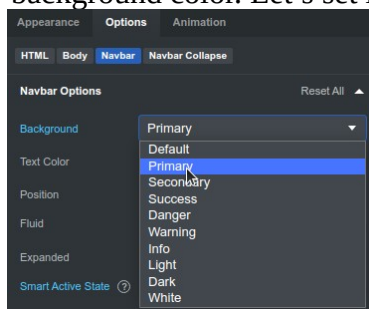
So far we have been using the default Bootstrap colors, and fonts, we've been changing the style of individual elements manually. We can change the look and the style of the website as a whole without needing to manually change every element. We can achieve this by changing the theme or we can modify the style using CSS.

Using Themes

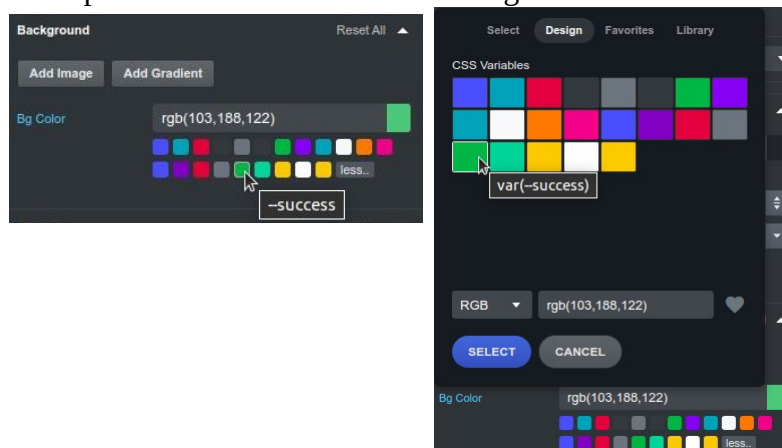
Themes are predefined layouts and styles of a website. Using an existing theme can be a fast and easy way to change the overall look and feel of your website, without needing to pay too much attention to detail yourself.

To demonstrate:

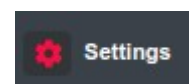
- Select the Navbar. In the options panel we see that we have different options for the background color. Let's set it to Primary.



- On the index page, select the paragraph and go to Appearance → Background.
 - I currently have a non-standard green selected. A list of the standard colors can be seen below the Bg Color input box.
 - Hovering over these colors gives the variable name. I will pick the green with the variable name '--success'.
 - Hint: another way to see the theme-colors is to click on the colored square next to the input box and choose the tab 'Design'.

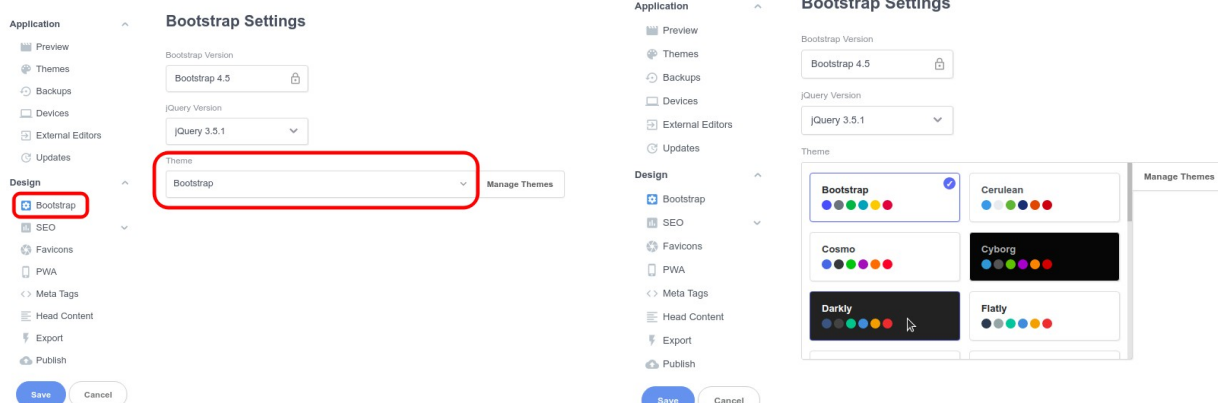


- Now that we have linked the Navbar's background and the paragraph background to variable colors we can change the theme of the website.
 - In the top of the window click on Settings.

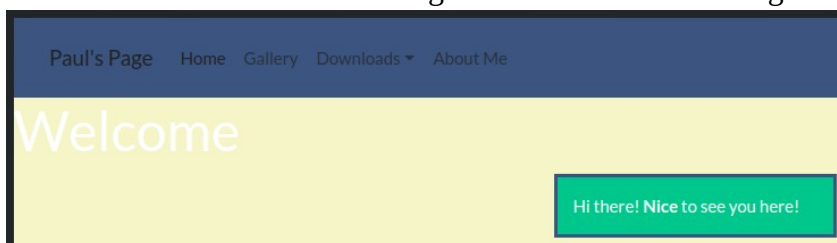


Bootstrap Studio in 12 Hours

- Under Bootstrap → Themes you can pick any number of themes from the list.
- I will pick 'Darkly' and save.



- As you can see the 'Primary' and '--success' colors have changed according to the new theme, as well as the standard text color.
 - Note that the color of the background of the page was set manually and is not linked to a theme-color. So the background color will not change no matter what theme you pick.



To avoid confusion, let's set the theme back to the default (Bootstrap theme) for now. In the next section we will step away from using themes, and create our own custom styling.

Custom Styles

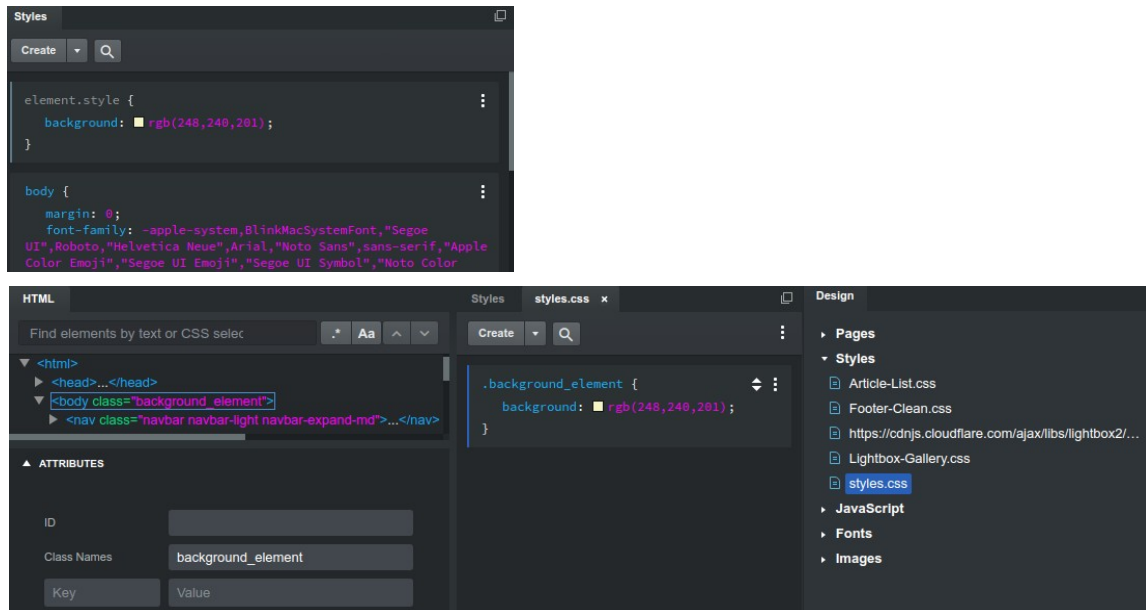
As we have seen, it's a simple matter to set the colors, fonts, margins etc.. of each element using the appearance panel. But this means that if we want to change the overall look and style we would need to go back and set the styling for each and every item. This can obviously be very haphazard and time consuming.

A better way would be to create a CSS style and link the style to the relevant elements using class attributes. In the following we will create a CSS style block which sets the background color to beige. We will then apply the style to all the relevant elements.

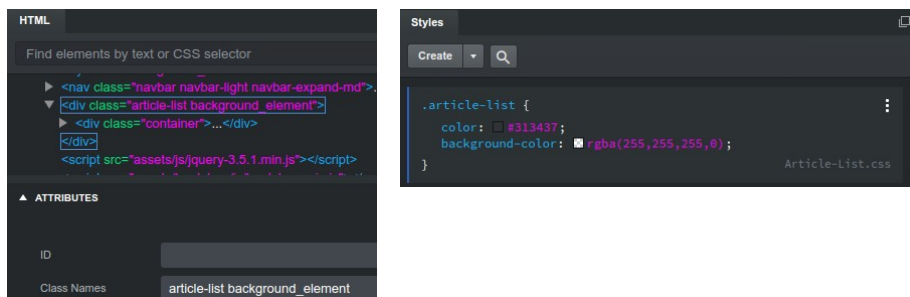
- Go to index.html, select the body, and pull up the code inspector.
- In the CSS panel you see a number of CSS blocks.
 - The first block is element.style. This shows the direct styling we have done using the appearance panel.
 - Some of the blocks have a padlock, showing that they cannot be edited.
- In the HTML panel pull up the attributes.
- Add a new class named 'background_item'. This class will identify all items where we want to set the background color.
- Let's open up the file styles.css

Bootstrap Studio in 12 Hours

- Go to the Design panel and open up the heading named 'Styles'.
- We see a list of css files that have already been included, like article-list.css. These were imported when we added some of the more advanced components to our page.
- Double click on styles.css to open the file. In the Styles panel a tab named 'styles.css' appears. So far it contains only one block for the background color.
- In the Styles panel click on Create. A new CSS block appears named .background_element. Let's type in the same background color here: background: rgb(248,248,201);
- Now the code in element.style has become redundant. We can remove it by going back to the Styles tab, finding 'element.style', clicking on the three dots in the top-right of the block and selecting 'clear'.



- Next go to the page AboutMe.html, select the body, and add the text 'background_element' to Class Names. As you can see, the background color is applied to this page too.
- When we add the same class name to the body of the page Download_Resources.html though we see that a large portion of the page remains white.
 - This is because the article list has a white background by default.
 - We can solve this by selecting the Block, and then adding the class 'background_element' next to the class 'article-list'.
 - Alternatively, you can find the style .article-list in the styles tab, and set the background color to transparent.

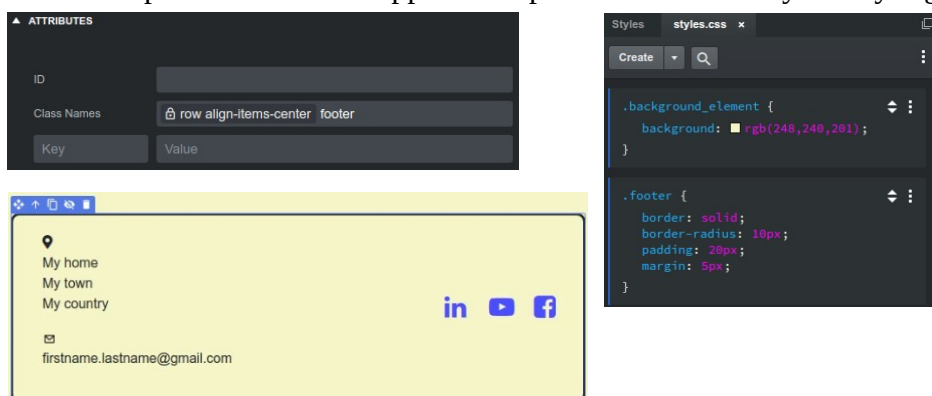


- Finally, we can repeat this for all the pages and all relevant elements.
- Now, if we want to change the background-color of the website, we only need to modify this one CSS block, and all the relevant elements will change.

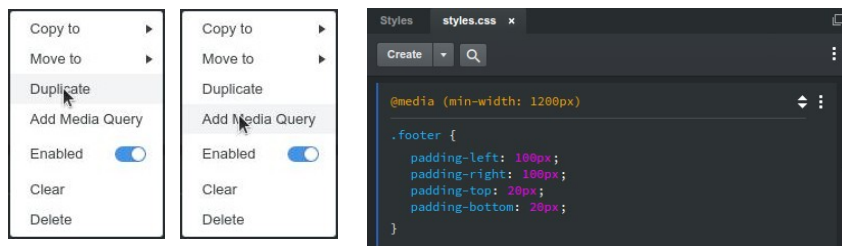
Responsive Styles

We can also add styling which depends on the size of the screen. For example, we can apply small margins if the website is being viewed on the phone and apply large margins if the website is being viewed on a desktop computer. We achieve this using a media query. For example, the footer looks fine on medium sized screens, but on small screens it's a little bunched up, while on large screens it's a little too spaced out. Using media queries we can increase the padding for large screens and decrease the padding on small screens.

- Let's start by adding the default styling.
 - Add a class attribute named 'footer' to the first row within the footer.
 - Make sure the file styles.css is open, select the row and create a new block by clicking on Create, and rename it to '.footer'.
 - Here we will add the default styling:
 - Give it a solid border with a radius of 10 pixels, a padding of 20 pixels and a margin of 5 pixels.
 - (Hint: we don't need to type this in manually. We just make sure we have a component with this attribute selected, then we choose the attribute from the top drop-down list in the appearance panel and we modify the styling from over here.)



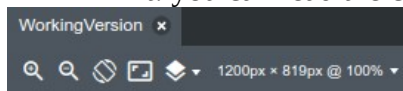
- This looks fine for medium and large screens, but on smaller screens it looks a little squashed and on larger screens it's a little too spaced out.
- Let's increase the padding for larger screens:
 - Switch to XL view.
 - Duplicate the .footer styling block by clicking on the three dots in the top corner and selecting 'Duplicate'.
 - In the duplicated styling block add a media query by clicking on the three dots and choose 'Add Media Query'. The text '@media (min-width: 1200px)' appears.



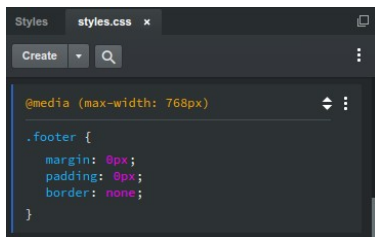
- This means:
 - Apply the styling only when the window-width is at least 1200 pixels wide!

Bootstrap Studio in 12 Hours

- Bootstrap studio has picked 1200 pixels, because we are currently viewing the screen at 1200 pixels (since we switched to XL view).
- Hint: you can read the current window width in the top-left of the canvas.



- Next, let's remove the border and decrease the margin and padding for smaller screens:
 - Duplicate the CSS block one more and modify the query to read: '@media (max-width: 768px)'
 - Modify the duplicated style block to set the margin and padding to zero, and to remove the border.



- With this query we apply the styling only when the window-width is 768 pixels and less.
- Hint: make sure the styling blocks end up in the correct order in your styles.css file. When a style ends up being applied twice it is the second one that takes precedence.

In the next chapter we will continue applying our own custom theme. We will override some the existing styling with our own, add styling for hovering items, and modify the menu-item icon. See you there!

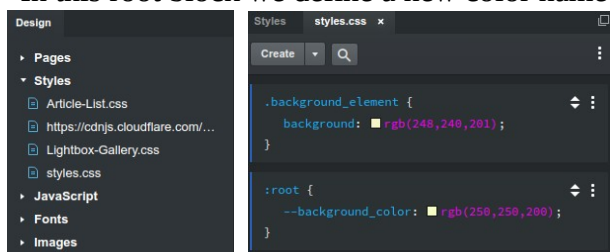
Hour 7 – More About Styles

In the previous chapter we learned to use CSS blocks to add styling to different elements. In this chapter we will finish off styling the header and the footer, and in the process we will learn about overriding, hover and how to change the menu icon.

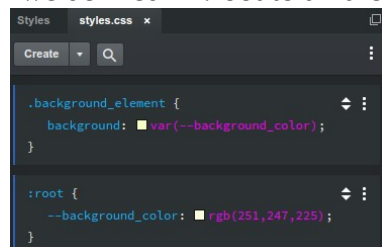
Defining Colors

When we set the background color we picked it directly. If we want to use this color more often it's a good idea to give the color an easy-to-remember name. That way we could more easily apply the color to different elements. This would also make it easy to change the color later. To do this we define the color in a css block named `:root`.

- Make sure the file `styles.css` is open and create a new block by clicking on Create. Rename it to `'root'`.
- In this root block we define a new color named `--background_color`.



- Now that we have defined the color let's use it in the `.background_element`. Modify the code in `.background_element` to include a `var(--background_color)`. This way we apply the color we defined in `:root` to all the elements containing the class `'background_element'`.



- Now look at the standard colors in the appearance panel. You will see that a new color has appeared! Opening up the palette and choosing Design we find our new color under CSS Variables.



Bootstrap Studio in 12 Hours

We can define any number of colors. So, for example, let's already set some of the colors we will use in later chapters.

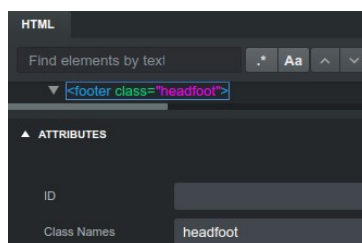
- Add the following colors:
 - headfoot_color: the background color of the header (navbar) and the footer.
 - headfoot_hover_color: the color of the header and the footer when the mouse hovers over it.
 - headfoot_text_color: the color of text of the header and the footer.
 - headfoot_text_hover_color: the color of the of the header.
 - headfoot_shadow: the color of the shadow effect we will use.
- Hint: if you prefer you can click on the colored square to use the color palette to set the color instead of using unintuitive rgb values!

```
:root {  
  --background_color: #rgb(251,247,255);  
  --headfoot_color: #rgb(30,30,60);  
  --headfoot_hover_color: #rgb(12,12,24);  
  --headfoot_text_color: #rgb(255,255,0);  
  --headfoot_text_hover_color: #rgb(255,255,255);  
  --headfoot_shadow: #rgb(12,12,40);  
}
```

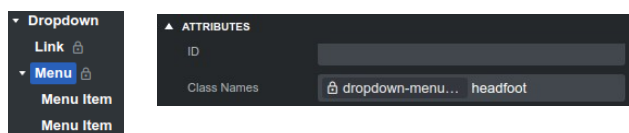
Now that we have defined our color palette, let's apply the new colors to the Navbar and the footer.

- Create a new style block named '.headfoot'. This block will contain our custom styling for the Navbar and the Footer.
- Set the background color to --headfoot_color, set the text color to --headfoot_text_color, and set the text shadow to '3px 3px var(--headfoot_shadow)'.
- We can either type it in manually, or we can use the appearance panel to add the styling.
- Link this style block to the Navbar and the Footer elements by adding the class attribute 'headfoot'.

```
.headfoot {  
  background-color: var(--headfoot_color);  
  color: var(--headfoot_text_color);  
  text-shadow: 3px 3px var(--headfoot_shadow);  
}
```



- We also need to link the style to the Dropdown menu. Select the menu element and add the class attribute 'headfoot' here too.



You have probably noticed that the text color in the navbar and the icons in the footer still have their default color. This is because these colors have already been defined in another CSS block. We can solve this by overriding those CSS blocks. That's the topic of the next chapter.

Finally, let's finish up this chapter by doing a little quick cleaning up in the footer:

- The border does not look great colored yellow like this. In fact Let's replace it entirely with a drop-shadow, and link color to the shadow-color we picked previously.

Bootstrap Studio in 12 Hours

```
.footer {  
  box-shadow: 0px 0px 12px var(--headfoot_shadow);  
  padding: 20px;  
  margin: 5px;  
}
```

- Next we should not forget to update the CSS block for small screen-sizes, by removing the box-shadow rather than the border.

```
@media (max-width: 768px)  
{  
  .footer {  
    margin: 0px;  
    padding: 0px;  
    box-shadow: none;  
  }  
}
```

- Let's also add a little space between for the container within the footer. The best way is to add a little top and bottom padding to the .footer block.

```
.headfoot {  
  background-color: var(--headfoot_color);  
  color: var(--headfoot_text_color);  
  text-shadow: 3px 3px var(--headfoot_shadow);  
  padding-top: 15px;  
  padding-bottom: 15px;  
}
```

Now that we've linked these colors to up our template we can easily play around with our colors!

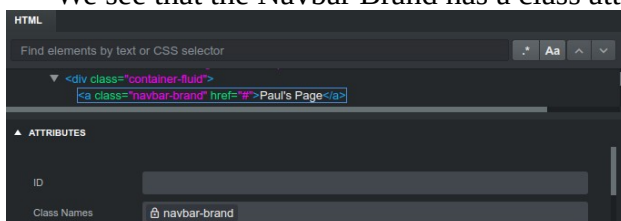
- We can easily change each of our colors.
- Since the canvas updates immediately, it's really useful when trying to find a nice color combination for our site.

Overriding

You will no doubt have noticed that many of the styling blocks are locked, and have 'Bootstrap' written in the bottom right corner. These CSS style blocks are locked in Bootstrap Studio. In theory it's possible to edit these styles, but this is not desirable. That's because when the Bootstrap Framework is next updated you would lose all the changes you made to the Bootstrap CSS files.

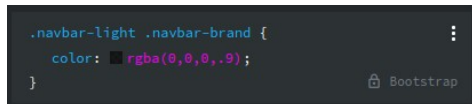
In the previous chapter when we applied a CSS block to certain elements like links and the Navbar, some of the styling did not come through. This is because the styling for these elements was already set in these locked CSS blocks. In order to fix these we need to override the locked blocks.

- To override the styling of an element, we first need to determine which class the element is linked to.
 - Select the Navbar Brand in the Nav bar.
 - Pull up the HTML inspector and pull up the attributes tab.
 - We see that the Navbar Brand has a class attribute named 'navbar-brand'

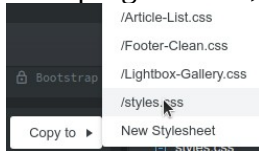


- Going to the styles tab, we get an overview of all the css blocks involving the class attribute 'navbar-brand'. We are looking for the one which sets the font color to black.

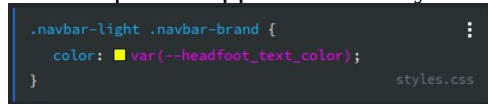
Bootstrap Studio in 12 Hours



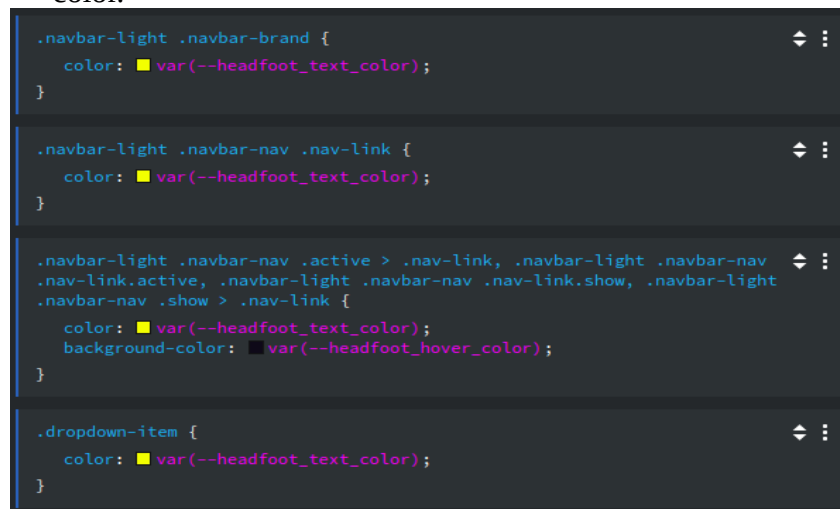
- We cannot edit this block, so we need to override it instead. Click on the three dots in the top right corner, and select Copy to → /styles.css



- A duplicate appears in the styles.css file. Let's edit the duplicate.



- Repeat this with the remaining elements:
 - The active link has a class named 'nav-link active'. Again we are looking for the CSS block which defines the color.
 - The remaining links have a class named 'nav-link'.
 - The drop-down items have a class named 'dropdown-item'.
 - Select each element one-by-one, find the appropriate CSS blocks, copy them to styles.css and modify the code accordingly.
 - For the active nav item we also set the background color of the item, as well as the text color.



- Note: it is important not to change the name of the CSS block. You might think that you could override the attribute by using the only class name of the block, but this does not work because of 'CSS specificity'. What this is is a bit beyond the scope of this course. But for those who are interested here is some extra reading!
 - <https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity>
- Next let's override the icons in the footer
 - When we select the icon to find the class name we bump into a problem. The class cannot be found.
 - In this case the font color is not set by a CSS block directly. It is made blue because it is a link. In CSS the styling of the links is defined by a block named 'a'.

```
a {  
  color: #007bff;  
  text-decoration: none;  
  background-color: transparent;  
}
```

- We could override this in the usual way. But that would set the styling for all the links in the page. We're really only interested in setting the styling of the three icons in the footer. In this case, since we have only three icons it's a better solution would be to add another class specifically for these links.
- Select the Icon and add the class attribute 'headfoot_icon'.
- In styles.css add a simple CSS block setting the color.

```
.headfoot_icon {  
  color: var(--headfoot_text_color);  
}
```

Hover

We would also like to set the style for the Navbar items for when the mouse hovers over it.

- Let's start with the icons in the footer:
 - Duplicate the CSS block named .headfoot_icon and add the text :hover.
 - Change the font color to var(--headfoot_text_hover_color).

```
.headfoot_icon:hover {  
  color: var(--headfoot_text_hover_color);  
}
```

- Next let's set the hover color of the Navbar Brand and the Nav Items:
 - Select the Navbar Brand and, in the Styles tab, search for the CSS block which ends with the code word ':hover' and sets the color.
 - Copy the block to styles.css by clicking on the three dots in the top right corner of the block and selecting Copy To.
 - Modify the code to set the hover colors.
 - Repeat this for the Nav Items.

```
.navbar-light .navbar-brand:focus, .navbar-light .navbar-brand:hover {  
  color: var(--headfoot_text_hover_color);  
  background-color: var(--headfoot_hover_color);  
}  
  
.navbar-light .navbar-nav .nav-link:focus, .navbar-light .navbar-nav .nav-link:ho  
ver {  
  color: var(--headfoot_text_hover_color);  
  background-color: var(--headfoot_hover_color);  
}
```

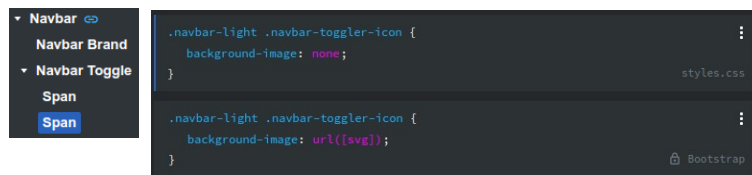
- Finally let's set the hover color for the dropdown menu too.
 - Select a menu-item and search for the block with :hover and which sets the color.
 - Copy it to styles.css
 - Modify the block so that it sets color. In this case we also need to override the background color to!

```
.dropdown-item:focus, .dropdown-item:hover {  
  color: var(--headfoot_text_hover_color);  
  background-color: var(--headfoot_hover_color);  
}
```

Changing the Menu Icon

Finally we have one more item that needs to fit into our customized theme: the menu icon. We want to replace the darkly colored icon with a light colored one. Unfortunately this icon is an image and not an Icon like the ones we added before, so changing the font color will not help.

- Let's start by removing the current menu item image.
 - The menu icon can be found in the second span in the Navbar Toggle. Select it.
 - In the styles tab we find the CSS block which sets the background image of the navbar-toggler-icon to url([svg]). Override this block by copying it to the styles.css file.
 - Replace url([svg]) with none.



- Next we add and an Icon in it's place.
 - Drag and drop an icon into the span.
 - Double click on the icon and find the one named 'fa fa-bars' in Font Awesome 4.
- The icon takes on the styling of it's grand-parent 'Navbar Toggle'. To solve this:
 - We add the class name 'headfoot' to the Icon element
 - Next we manually set the padding to 0px.

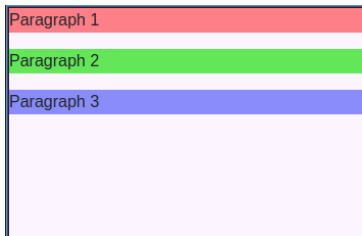
Hour 8 – Flexbox

Flexbox is a CSS layout model which can be used to easily specify the spacing between different elements.

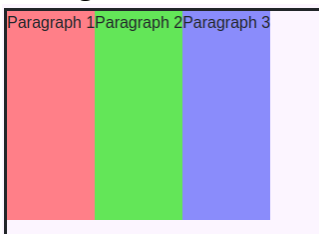
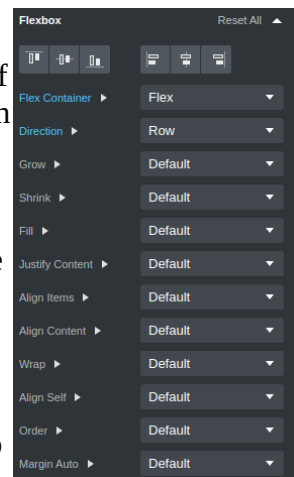
Flexbox Explained

In this section we will explain the basics of the flexbox.

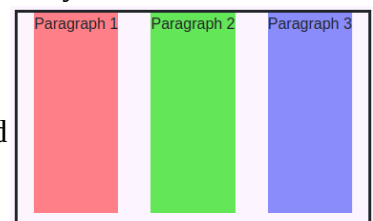
- First let's set things up:
 - Start by adding a simple div to the page.
 - Within that div let's add 3 paragraphs, and let's edit the text to name them paragraph 1, 2 and 3.
 - To help visualize things, let's add a small margin to the div, increase the height of the div and add a border. Let's also set red, green and blue background colors to the paragraphs.



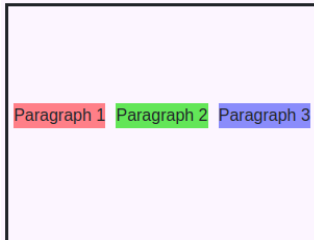
- Under options we see a title Flexbox where we see a number of drop-down lists. We can use these drop down lists to set either properties of the box (in this case the div) or to set properties to the elements within the box (in this case the three paragraphs).
- Let's set some of the box-properties to affect the layout of the paragraphs within.
 - Make sure the div is selected and find the heading 'flexbox' in the Options panel.
 - First of all set the Flex Container to flex.
 - Next set the direction to row. As you can see the paragraphs are now arranged side-by-side.
 - (We can also set the direction to row-reverse or column-reverse to change the order of the paragraphs.)



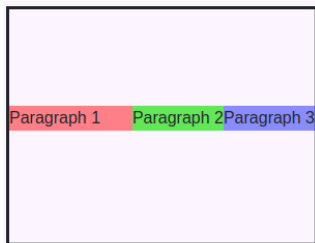
- We can also change the positioning of the paragraphs using 'Justify Contents'.
 - Start, End and Center positions the items at the start, end and center of the row.
 - Between puts the first paragraph at the start and the last paragraph at the end. All remaining paragraphs are spaced equally between them.



- Around spaces the paragraphs equally within the available space. Let's choose this property.
- (If we had set the direction to 'Column' the same positioning would as above, but with the spacing applied vertically rather than horizontally.)
- We can also use Align Items to set a spacing perpendicular to the set direction: for example if we set it to center the paragraphs are sent to the middle of the div.



- We can also set properties to each element within the div. For example:
 - Select one of the paragraphs and set the grow value to yes. We see that the paragraph grows to take up as much of the available space as possible.



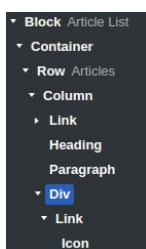
There's still more to learn about Flexbox, but for the sake of this course that's all we need to know for now. If you are interested in learning more I recommend the following site: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Next we will apply what we have learned in a simple example.

Example – Arranging Icons in an Article List

In this example we will use flexbox to modify the article list to add an array of icons.

- Go to the page 'Downloads_Software.html'
- Drag and drop the article list to the page, between the header and the footer.
- Delete the introduction and delete two of the articles.
- Drag and drop a Div into the remaining article, just after the paragraph.
- Delete the link containing the Icon.
- Drag and drop a new link into the Div, edit out the text, and drag and drop a new icon into the link.



Bootstrap Studio in 12 Hours

- Set the font size of the link to 32px.
- Duplicate the link (including icon) twice.
- I will set my icons to the Linux Penguin, the Windows Logo and to 'fa fa-code'.
 - We can also set the tool-tip of each link under options. I will set mine to "Linux build", "Windows build" and "Source".
- Finally let's set up the flexbox:
 - Select the Div and go to the Options panel
 - Set the Flex Container to Flex, the Direction to Row and Justify Content to Around.Now that we have a template for our article we can easily duplicate this to populate the article list.



Article Title

Aenean tortor est, vulputate quis
leo in, vehicula rhoncus lacus.
Praesent aliquam in tellus eu
gravida. Aliquam varius finibus
est, interdum justo suscipit id.



Example – Pushing the Footer to the Bottom of the Page

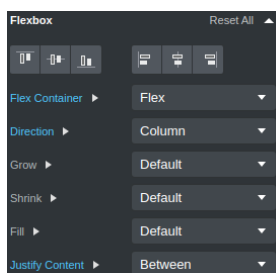
You may have noticed that the footer does not always reach the bottom edge of the page. For example, if we go to the 'About Me' page, for screen sizes medium and larger, the footer does not reach the bottom of the page. This looks a bit ugly.

We can solve this by using flexbox to create an automatic spacing which will push the footer down to the bottom of the page.

- First set the height of the Body element to '100vh'.
 - '100vh' means 100% of the available vertical height. This makes sure that the body element is as big as the available space.



- Next set the flexbox properties of the body to:
 - Flex Container: Flex
 - Direction: Column
 - Justify Content: Between



Hour 9 – Custom Components

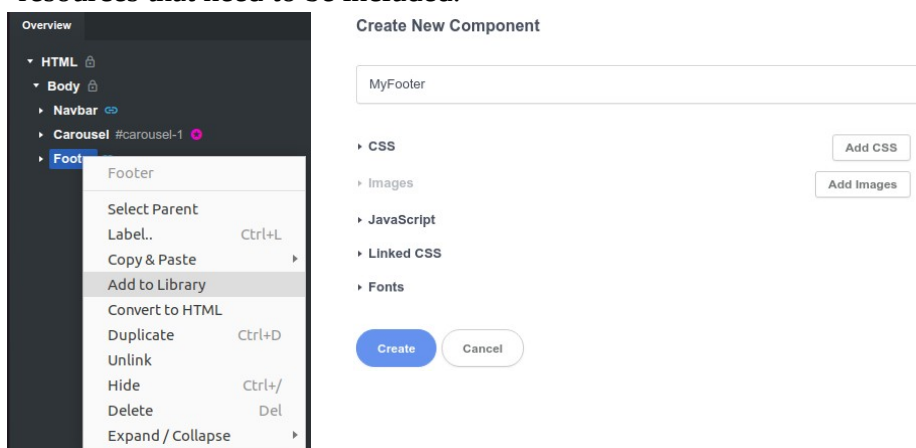
We have seen a number of simple and elaborate elements in the Studio library. But we need not limit ourselves to these. In this chapter we will start by getting a very brief introduction to some more useful components both in the library, and also available online. Next we will learn about building up custom components. Finally we will apply this knowledge to create a personalized logo component.

Adding Custom Components to the Library

As we have seen we don't need to rely on existing components, but we can create our own! We can build up a component from simpler components, as we have done to create our own footer for example. Or we can start with an elaborate component and add to it, or modify it to fit our needs.

Once created, we can copy components to the library. This way we could easily use it in multiple projects without having to recreate the same elaborate component over again. And, if you are feeling generous, we could even share it online. As an example we will add the footer to the library, and use it in a new project.

- In the overview panel, right click on the footer and select 'Add to Library'
- A window pops up. We can set the name of the Component, and we can also select the resources that need to be included.



- The CSS blocks that are directly linked to are already included, but we need to add one more block. Click on the 'Add CSS' button.
- In the window that pops up, under styles.css, we see that we need to include the block named :root, where the colors have not been defined.



Bootstrap Studio in 12 Hours

- In the studio library we see that a new component has been created in the folder ‘user’
- Let’s test it, to see if it works on other projects too. Create a new blank project and drag and drop the new footer onto the page!

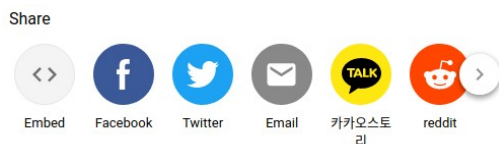
Custom Code

Another very useful component is custom code. Custom code is simply an empty block in which you can type or paste your own code. This can be very useful because many websites and platform over you the option to generate code which you can to ‘embed’ on you website.

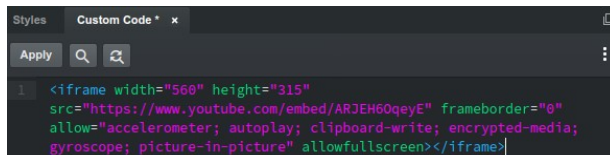
- First let’s go to the ‘About Me’ page and drag and drop a Custom Code element from the library to the canvas.
- First I surf to youtube and find the video I want to embed. I will add one of my Krita tutorial video’s: “Krita - Use Filters to add a glow (in under 2 minutes)”
- Under the video click on ‘Share’.



- Next click on ‘embed’.



- A window with some code shows up. Select the code and copy it to the clip board.
- Back in Bootstrap Studio, double click on the the Custom Code element and paste the code into the available space.



- Unfortunately the custom code does not show up on the canvas. To test the result we need to create a preview.
- We would like to center the video. The most straightforward way to do this is to add a div with automatic margins, and to drop the custom code inside!
- When we copy custom code we need to set the class attribute manually.

We can also use custom code to edit an existing component. Just select the component you want to edit, right-click and choose convert to HTML. This will transform the component into a Custom Code component. When we double-click on the component in the overview panel the html code appears next to the style dialog. Here we can edit and tweak the component in any way we choose!

Example – Use Custom Code to Add Inline SVG

When it comes to pictures and graphics for your website you can choose either raster graphics (like png, jpg or bmp for example) or vector graphics. Raster images are essentially a grid of pixels, while vector graphics are rather a set of shape and color instructions. The most common type of vector graphic file format is called svg!

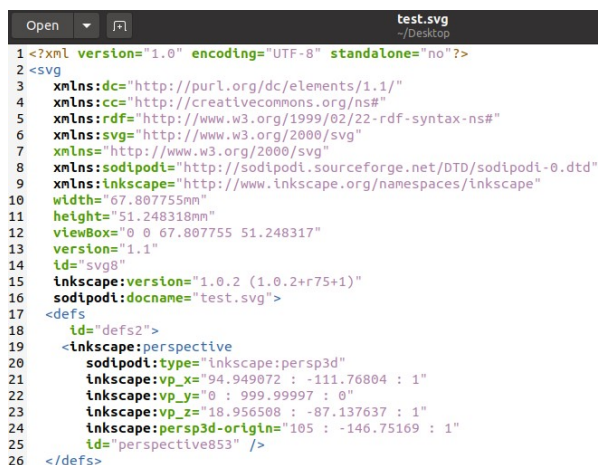
Vector graphics in general and the svg file format in particular have a lot of advantages:

Bootstrap Studio in 12 Hours

- Smaller: Generally the file size of vector graphics files tend to be smaller than those of raster images.
- Responsive resolution: The svg file format has no resolution, so it responds easily to the size of the device.
- Readable file format: Like html, svg consists of elements, tags and properties. This means that it's possible for humans like us to read it, and unambiguous for the computer to interpret. It also means that we have the option to modify the svg file manually and even use Javascript to add some functionality to the file!

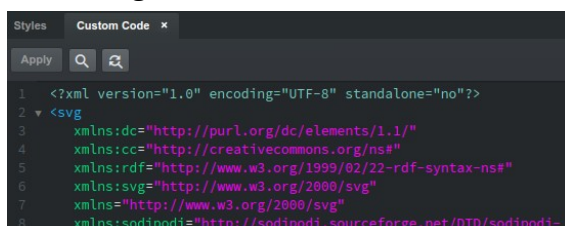
I have created a simple logo in Inkscape. Inkscape is a free and open-source software application used to create vector graphics and it is one of my absolutely favorite pieces of software. If you're interested you can download it from <https://inkscape.org/>. It just so happens that I have a course about Inkscape too! You can find it on Udemy (<https://www.udemy.com/course/getting-start-in-inkscape/>) and Skillshare (<https://skl.sh/2yzcHeE>), or you can find the free version on YouTube (https://www.youtube.com/channel/UCHBc6AIjNg1YapFd6Rer_kA).

- Svg files can be imported into Bootstrap studio in the usual way: just drag the drop the file from your file manager to the design panel in the bottom right corner.
- Another way would be to copy the svg code to a custom code block.
 - First open the svg file in a simple text editor like notepad.
 - We see that it looks quite similar to html. Although most of it looks like gibberish, it still looks a lot more intelligible than jpg data looks!



```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <svg
3   xmlns:dc="http://purl.org/dc/elements/1.1/"
4   xmlns:cc="http://creativecommons.org/ns#"
5   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6   xmlns:svg="http://www.w3.org/2000/svg"
7   xmlns="http://www.w3.org/2000/svg"
8   xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd"
9   xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
10  width="67.807755mm"
11  height="51.248318mm"
12  viewBox="0 0 67.807755 51.248317"
13  version="1.1"
14  id="svg8"
15  inkscape:version="1.0.2 (1.0.2+r75+1)"
16  sodipodi:docname="test.svg">
17  <defs
18    id="defs2">
19    <inkscape:perspective
20      sodipodi:type="inkscape:persp3d"
21      inkscape:vp_x="94.949072 : -111.76804 : 1"
22      inkscape:vp_y="0 : 999.99997 : 0"
23      inkscape:vp_z="18.956508 : -87.137637 : 1"
24      inkscape:persp3d-origin="105 : -146.75169 : 1"
25      id="perspective853" />
26  </defs>
```

- Select all the text by pressing Ctrl + A and copy it to the clip board by pressing Ctrl + C.
- Next go back to Bootstrap Studio, add a Custom Code block.
- Double-click on the custom code block to open it, remove the existing code and paste in the svg code instead.

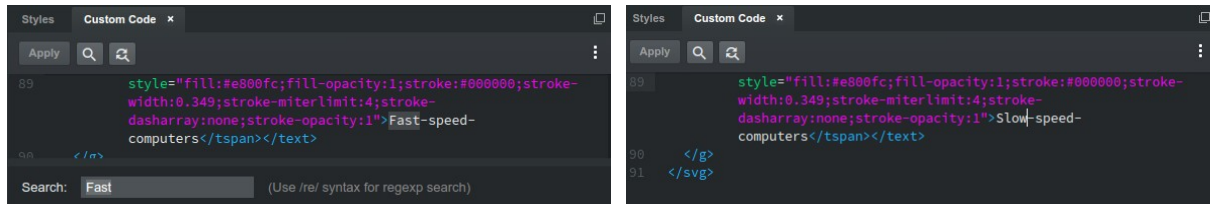


```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <svg
3   xmlns:dc="http://purl.org/dc/elements/1.1/"
4   xmlns:cc="http://creativecommons.org/ns#"
5   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6   xmlns:svg="http://www.w3.org/2000/svg"
7   xmlns="http://www.w3.org/2000/svg"
8   xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-
```

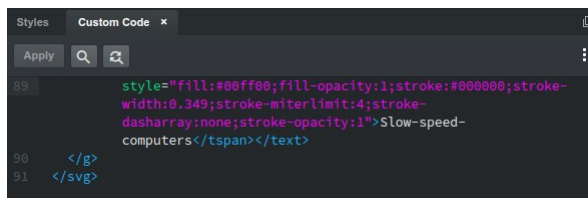
- In this way svg code is embedded in our html. This is called inline svg. There is the cleat disadvantage that it does not show up on the canvas. We need to create a preview to see what it looks like.

Bootstrap Studio in 12 Hours

- Inline svg does have some significant advantages though. For one, since the text is embedded in the code, it can be found by search engines, which is great for seo (search engine optimization).
- Another benefit is that we can make some simple edits to the logo manually:
 - The text on the logo is “Fast-Speed-Computers”. We can change this text easily. Just use the search to find the text, and edit it!

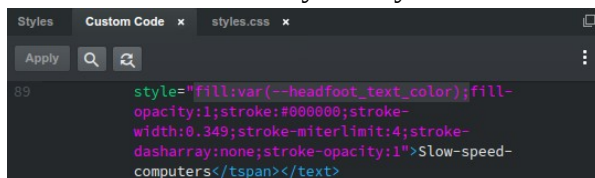


- It's also reasonably simple to change the color of any of the objects in the logo. Each object contains a list of style properties. It is the 'fill' property that sets the style. So for example, to set the text fill color to green we modify the style to read “fill:#00ff00” instead of “fill:#e800fc”.

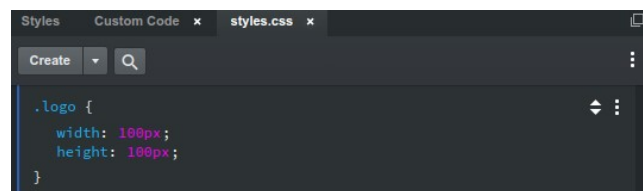
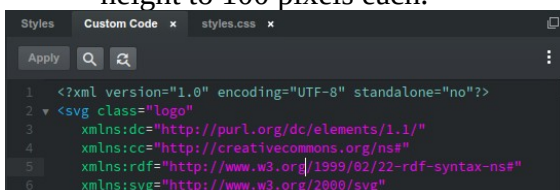


(The values are in HEX. If you prefer to work in rgb values though, it's not too difficult to convert rgb colors to hex. One way is to use the website <https://www.rgbtohex.net/>)

- One cool thing is that we can even link the color to one of the variables we'd defined before. Just modify the style to read “fill:var(--headfoot_text_color)”



- In this way we can easily modify the colors in the logo just by changing the value of the variable.
- A third benefit is that we can even use javascript to control properties of the svg elements. We'll learn more about that in the next chapter.
- Finally, we see that we cannot modify the style of the Custom Code in the appearance bar as we did with the other components. But we can still modify the style manually:
 - First let's add a class attribute to the svg code. Let's name the class simply “logo”.
 - Next add a CSS block named “.logo” to the file styles.css where we set the width and the height to 100 pixels each.



Hour 10 – Forms

In the previous chapter we have seen that it is a simple matter to add all kinds of form items like buttons, text inputs, checkboxes etc to the website. To have them actually do something is another matter though. In this chapter will cover the very basics of adding functional forms to our website.

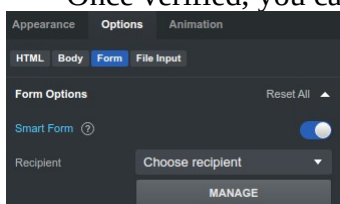
We will start by creating some simple forms and linking them to our email address. These are called smart forms in Bootstrap Studio. In the second section we will learn to get a bit more control over our forms using some very basic javascript. And finally we will round things off with a slightly more elaborate example where we use javascript to edit an svg image.

Smart Forms

After making a form for a user to fill out, Bootstrap Studio comes with a very easy way to link the form to your email address. This way the user can fill out the form online and all the information will be sent to you as soon as her or she click the submit button. To do this we use smart forms. Smart forms can be added to your site without the need of any coding!

Let's get started!

- Start by adding a form component. Let's link it to an email address:
 - Toggle 'smart form' in the options panel.
 - Click MANAGE and then click on 'add recipient'
 - You will need to verify the email using a verification code sent to the address.
 - Once verified, you can choose the recipient email address from the drop down list.

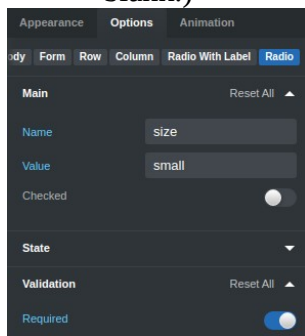


- Let's set up the form to look like this. We'll start with the appearance:

- Add a row to the form, and add two columns to the row.
 - Set the margin of the form to '20px auto'.
 - Add some padding, a border with rounded edges, and a drop shadow.
- Next let's add some radio components:
 - In the first column add a field label and add 4 'Radio with Label' items.
 - Set the labels as shown in the picture above.

Bootstrap Studio in 12 Hours

- We can modify the properties of each of the radio components in the options panel. We want to set the properties as follows:
 - Set the name of each of the radio components to 'size'. (Giving each of the radio buttons the same name puts them in the same group, so that selecting one will deselect the other.)
 - Set the value of each radio component to 'small', 'medium', 'large' and 'extra large' respectively. (This is the information that will be passed on to us via email)
 - Set the validation to required. (This way the user does not have the option to leave it blank.)



- Add another field label and another four radio components to the second column. Set them up in the same way:
 - Set the name to 'color'
 - Set the values to 'red', 'green', 'black' and 'cyan'
 - Set the validation to required
- Let's add the submit button so that we can test the form so far:
 - First let's add a div, and add a button to the div.
 - Select the div and, in the options panel change the alignment to push the button to the right hand side of the form.
 - Select the button and change the Button Type to 'Submit'
- Let's test it out:
 - Generate a preview and push button.
 - Note that if you have not selected any of the radio components the form reminds you to.
 - If the form has been filled out properly a captcha will appear.
 - After passing through the captcha an email with the information from the filled out form will be sent to you.
- We can also add fields that require information in a certain format. For example we can ask for an email address so that we can contact with the user.
 - Add another row with two columns.
 - Drop in a field label and a component named 'email input'.
 - Let's adjust the text and set the column widths to 3 and 9.
 - Select the email input and set the validation to required.
 - Now the user needs to add a valid email address before he or she can submit the form.
- In the component library under the 'form' heading we find a large number of useful form items. For example:
 - Color input allows the user to pick any color from the rgb color set.
 - Telephone input allows the user to enter a telephone number.
 - We even have file input, which allows the user to add a file as an attachment. Let's add it:
 - Add another row with two columns.

Bootstrap Studio in 12 Hours

- In the first column we add a label.
- And in the second column simply drop in the File Input component.

Smart forms is a very powerful feature of Bootstrap Studio, allowing us to set up user forms with no coding required. There are even some existing smart form components, which you can find under the heading UI → Forms. For example, the form ‘Contact Form Clean’ is an elegant contact form that you can easily drag and drop onto your website.

But we cannot do everything with smart forms. Next section we will learn to add some very simple javascript to the website to add some extra functionality. The idea of coding can be a little daunting for some people, but in fact coding is not that hard and actually loads of fun. To ease into it, let’s start off with some tips for beginner coders.

Hour 11 – (Optional) Quick Introduction to Coding

If you are interested in learning javascript then this hour would make a great exercise. If you're not really that interested in coding, or if you don't think coding will be helpful for you, then feel free to skip this chapter and move on to the next, where we will learn about publishing and exporting our website.

In this chapter we will be coding a little in javascript. Many people who are new to coding feel intimidated by it. If you are one of those people then don't worry. In this section we stick to the very basics. Besides, coding is lots of fun! Follow along with me to get started!

Custom Functionality Using a Little Javascript

In this section we will adding a checkbox to the form we previously created. This will allow the user to opt-in to receive promotional emails. If the checkbox is left unchecked the email input field will be left deactivated. If the user checks the checkbox, on the other hand, then the email input field will be activated. Before we get started though, here are some helpful tips for beginners:

1. Be precise ... very precise! Even a seemingly insignificant typo's can easily break the code.
2. Test the code very often. For beginners it's no exaggeration to test the code for every single line added. After all, it's much easier to find a typo in a single line of code than in a number of lines.
3. Try to think like a robot (logically and systematically). Coding is a great exercise for the brain. The more you code the better you get at thinking logically and systemically.

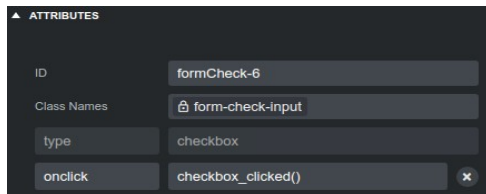
With that in mind, let's get started!

- First let's drop in a checkbox with label, and adjust to label to read 'Please send me newsletters and promotional material!'
- Next select the email input, and under the 'state' in the options panel, toggle 'Disabled'.
- Next we would like to add a javascript file:
 - Go to the design panel and find the heading "Javascript".
 - We see that we already have some javascript. This was imported when we added the lightbox gallery.
 - Right-click on the heading, choose "new", and choose "JS File"
 - Since we will have only one javascript file, let's be very imaginative and name it "javascript.js"
- We would like to create simple function which runs whenever the user clicks on the checkbox.
 - Double-click on javascript.js to edit the file.
 - Add the following function. This simple function simply creates a message dialog with the message "hello"
 - The name of the function is 'checkbox_clicked'
 - All instructions for the function are given between the curly brackets
 - Right now there is only one instruction: 'alert("hello")' which will create a popup with the text message "hello"

```
1 function checkbox_clicked() {
2   alert("hello");
3 }
```

Bootstrap Studio in 12 Hours

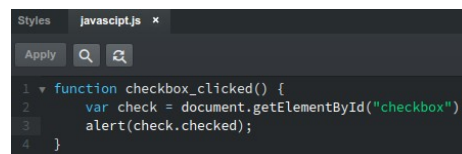
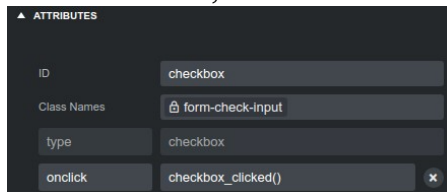
- We would like to run this function when the user clicks on the checkbox.
 - Select the checkbox and open up the attributes panel
 - We add the key 'onclick' with the value 'checkbox_clicked()'



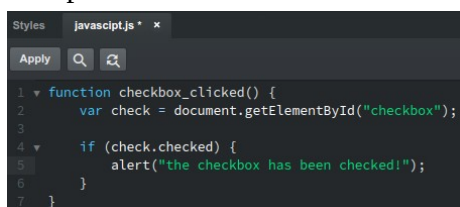
- Create a preview to test what we have so far!



- Now that we have linked a function to the checkbox we can edit the function to enable or disable the email input field.
 - We don't need the alert anymore. Let's remove it.
 - Next we would like to check if the checkbox has been 'checked' or not. To do this we need a way to identify the element.
 - Select the checkbox and open the attributes panel
 - Set the ID to "checkbox"
 - In the javascript file we create a new variable named "check" and use "document.getElementById" to link the checkbox to the variable.
 - Let's add another alert to see whether the checkbox is checked or not.
 - As we can see from the preview the output is 'true' when the checkbox has been checked, and it is 'false' when the checkbox hasn't been checked.



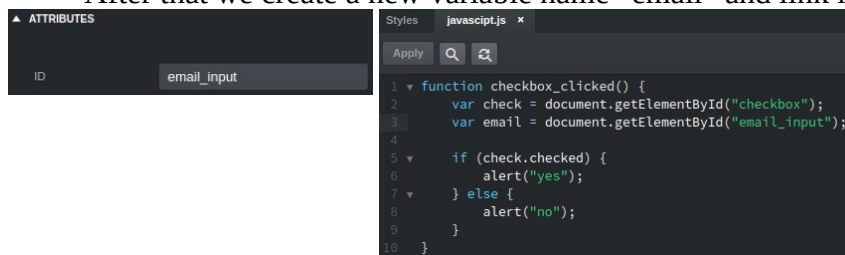
- Now we would like to set up a simple condition.
- The three lines added in the following are an 'if-clause' and consists of the following parts.



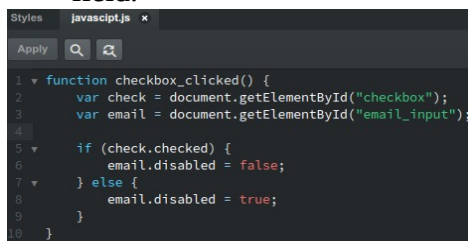
- The if clause starts with the key word "if" followed by a brackets.
- Everything between the brackets is a condition which can either be true or false. In this case the condition is check.checked.
(In other words, if the checkbox has been checked the value of 'check.checked' is 'true'. If the checkbox is unchecked the value of 'check.checked' is 'false')
- Everything in the subsequent curly brackets will only happen if the condition is true.

Bootstrap Studio in 12 Hours

- Let's also add an alert for the case that the checkbox is left blank. One convenient way to do this is to use the 'else' keyword. The extra lines in the code below show how to use the 'else' keyword in combination with 'if'.
 - Everything between the first two curly brackets will run in case the condition is true.
 - Everything between the two curly brackets after 'else' will run in all other cases.
 - So in this case we expect an alert with the text 'yes' if the checkbox has been checked, and we expect an alert with the text 'no' if the checkbox has been unchecked.
 - It's a good idea to test the code frequently, to make sure we haven't made any typos!
- Next let's replace the alerts with some code to enable or disable the email input.
 - First we need to give the email input an id, so that the javascript can find it. Select the input field and, in the attributes panel, add the id "email_input"
 - After that we create a new variable name "email" and link it to the id.



- The we replace the two alerts with code that enables, and disables the email input field.



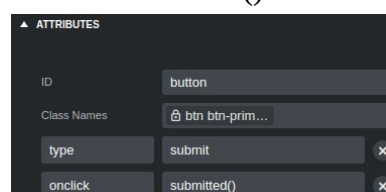
Having tested we see that it works! Hooray! We have one problem though. The form automatically resets the checkbox afterwards, but does not disable the email input field. To fix this we can reset the field ourselves.

Resetting the Form

While making the form, we have noticed that, after the form has been submitted, the email input field is enabled, while the checkbox is unchecked. This can be a little confusing for the user. One way to solve this is to add a function which disables the email input field when the submit button is pressed.

- First of all we create a new function named "submitted()".
- We add a single line which disables the email input.
- And finally we link it to the 'Submit' button by selecting the button and adding an onclick attribute with the value of the new function "submitted()".

```
13 function submitted() {  
14     var email = document.getElementById("email_input");  
15     email.disabled = true;  
16 }
```



As we test it, we see that this solves the problem, but unfortunately results in another problem. When we disable the input field we also disable the validation. That means that we've created a situation in which someone might input an invalid email address.

To solve this we will create our own email checking function. Only problem is I have no idea how to do that! That's ok though, that's what the internet is for. Let's go to a search engine and search for the term "javascript check valid email". Quite quickly we find the following website! <https://ui.dev/validate-email-address-javascript/>

- Let's copy paste the function we find on the website into our javascript file.

```
20 function emailIsValid (email) {  
21     return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);  
22 }
```

- It's not easy to understand all the details of this one line of code, but that's ok too. As long as it does its job properly! The main thing we should realize about this function is:
 - We need to provide one parameter, namely the email address.
 - The function returns a value which can be either true or false.
- Let's use this function to add a new if-condition to the "submitted()" function.

```
12 function submitted() {  
13     var email = document.getElementById("email_input");  
14  
15     if (emailIsValid(email.value)) {  
16         email.disabled = true;  
17     }  
18 }
```

- We call the function we just added, and use the value of the email input field as parameter needed by the function.
 - If the value of the email input field has the form of a valid email address the function will return 'true'. In this case it'll run the next line and disable the email input field.
 - If the value of the email input field does not have the right form, then the function will return 'false', and the next line will be skipped.

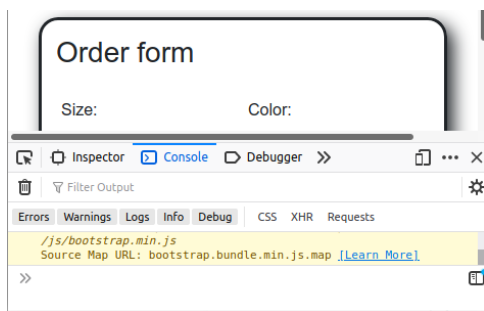
And there we have it!

Debugging

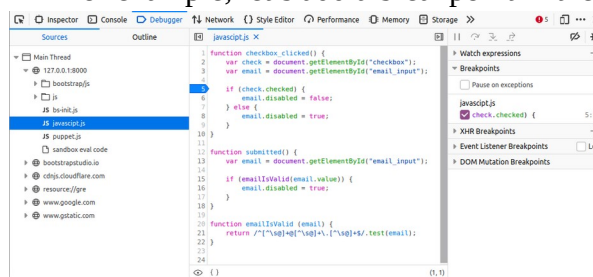
Although coding can be lot's of fun, it can also be quite frustrating with there's a problem with the code and you just can't figure out what's going wrong and why. This is called debugging. As you get more experience you start to get more adept at finding bugs and writing things up in a systematic way that avoids bugs. In the mean time you can expect bug-hunting to take up a significant portion of time in your coding adventures. Luckily there are tools available to help us catch those bugs a little faster. These are called the development tools which come with most browsers.

- Go to the preview and open up the developer tools. In most browsers you can achieve this by pressing F12. A series of panels open up. Right now we are interested in only two of these panels: the console and the sources panel / debugger panel.

Bootstrap Studio in 12 Hours



- The console provides error messages, warnings and log information. If your code fails there will probably be an error message here telling you what went wrong and on which line of code.
- There is also the debugger panel (called debugger in Firefox, and called source in Chrome and Edge) which provides a number of useful debug options:
 - We can add a breakpoint. The code will pause whenever it comes across a breakpoint.
 - For example, let's add a breakpoint in the function 'checkbox_clicked()'



- When we try to change the value of the checkbox the code pauses at the breakpoint.
- Now when we hover over any variable with the mouse we see it's value and it's properties.
- We can also step through the code step-by-step to see what happens.



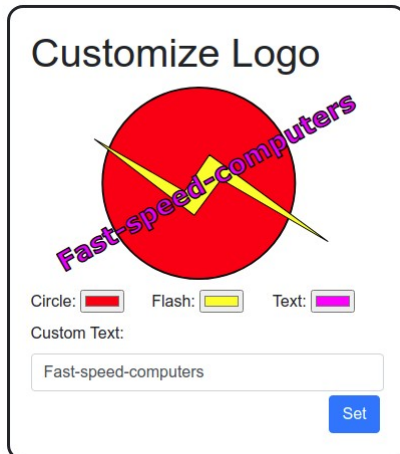
(1) (2) (3) (4)

1. Resume – Continue running the code until the next break point
2. Step over – Runs the next line of code. If there is a function on the current line the debugger executes the entire function without showing us each line of the function.
3. Step into – Runs the next line of code, but if there is a function on the current line the debugger jumps to the function to show us the function.
4. Step out – If we have used 'step into' to inspect the function line by line, we can use step out to skip seeing how the remainder of the function is executed.

Extra Example: Interactive SVG

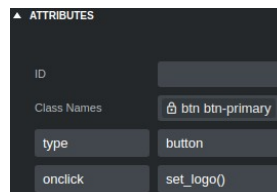
Finally we are going to do one more exercise, just for fun. We will use javascript to dynamically edit svg files. In other words, the user will be able to pick the colors of this logo.

- Let's start by adding all the form components:



- Start by adding a form element and dropping in a heading and three divs.
- Next let's set up the appearance to taste of the form. For example I will set the width to '450px', the margin to '50px auto' and I'll add a border with rounded edges
- In the first div I'll drop in a custom code element, set the alignment of the contents of the div to 'center', and add a little vertical margin.
- Open up the accompanying svg file in a text editor and then use Ctrl+A, Ctrl+C and Ctrl+V to copy the text to the clipboard and paste it to the custom code.
- Let's drop a row into the second div and add three columns. To each column we add a 'field label' and a 'color input'
- Modify the label texts and set the values to #FF0000, #FFFF00 and #FF00FF. (These are HEX values corresponding to red, yellow and magenta).
- Below the row add a 'Field Label' and a 'Text Input'. Let's set the default value to "Fast-speed-computers".
- Finally, add a button to the third div, set the the alignment of the div to right-align, set the text to "Set" and add a little margin. Let's also give the button a little extra top margin to give it a little more space.
- Next we would like to add a function and link it to the button
 - Add a function named `set_logo()` and include a simple alert.
 - Then link it to the button by adding the attribute `onclick = "set_logo()"`.
 - Let's test it to make sure it works! If it works we can remove the alert.

```
26 function set_logo() {  
27     alert("hi there! it works!... (so far...)");  
28 }
```

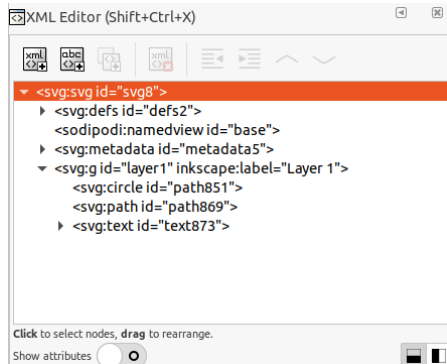


- Now that we've got a function set up we need to find the id's of the three objects.
 - In this case we have only three object: a circle, a lightning bolt and some text.
 - Double click on the custom components. We find the elements 'circle', 'path' and 'text' on lines 67, 73 and 78 of the custom code.
 - Looking more closely at each element we see that the id's of these elements are 'path851', 'path869' and 'text873'.
 - Let's add variables for each of these elements in the javascript.

```
26 function set_logo() {  
27     var circle = document.getElementById("path851");  
28     var flash = document.getElementById("path869");  
29     var text = document.getElementById("text873");  
30 }
```

Bootstrap Studio in 12 Hours

- Hint: an easy way to find the id of any object is to use Inkscape. Just open the svg file in Inkscape open up the XML Editor (Edit → XML Editor). Then select the object to discover or even change the id.



- We also need variables for the three color pickers and the text input.
 - We can set the id value for these elements in the attributes panel. Let's set them to 'colpick_circle', 'colpick_flash', 'colpick_text', and 'text_input'.
 - And we use these id's to create 4 new variables.

```
26 function set_logo() {  
27   var circle = document.getElementById("path851");  
28   var flash = document.getElementById("path869");  
29   var text = document.getElementById("tspan871");  
30  
31   var col_circle = document.getElementById("colpick_circle");  
32   var col_flash = document.getElementById("colpick_flash");  
33   var col_text = document.getElementById("colpick_text");  
34   var text_in = document.getElementById("text_input");  
35 }
```

- Then we use these variables to set the style of the svg elements according to the values on the form.

```
26 function set_logo() {  
27   var circle = document.getElementById("path851");  
28   var flash = document.getElementById("path869");  
29   var text = document.getElementById("tspan871");  
30  
31   var col_circle = document.getElementById("colpick_circle");  
32   var col_flash = document.getElementById("colpick_flash");  
33   var col_text = document.getElementById("colpick_text");  
34   var text_in = document.getElementById("text_input");  
35  
36   circle.style.fill = col_circle.value;  
37   flash.style.fill = col_flash.value;  
38   text.style.fill = col_text.value;  
39   text.textContent = text_in.value;  
40 }
```

- Finally we would like to make sure that the text stays center-aligned! To do this we need to modify the svg code.
 - A quick internet search gives us some hints! We need to set the text-anchor to middle, and express the x-value as a proportion!
 - Open up the custom code by double clicking on it.
 - Got to line 87 and change the value of x to "22%"
 - Also add an additional line: text-anchor="middle"

```
84 transform="rotate(-29.531293)"><tspan  
85   sodipodi:role="line"  
86   id="tspan871"  
87   text-anchor="middle"  
88   x="22%"  
89   y="112.38184"
```

- Caution! I figured this 22% from trial-and-error! This is a very result-oriented approach, because I do not fully understand what is happening! I have tested it on Firefox and Chrome, but if I wanted to do a proper job it'd be best to test it on as many browsers as possible!
- (I don't though, because this is just an exercise! So I'm going to leave it at that!)

Bootstrap Studio in 12 Hours

Anyway, that's all we will cover in this course regarding coding! I hope you enjoyed it. If you did there's still a lot to learn. If you want to learn more about javascript there's a lot of resources and courses out there that will help you on the way. W3 school is as usual a pretty good place to start! (<https://www.w3schools.com/js/default.asp>)

In the next chapter we will learn to publish the website we have created!

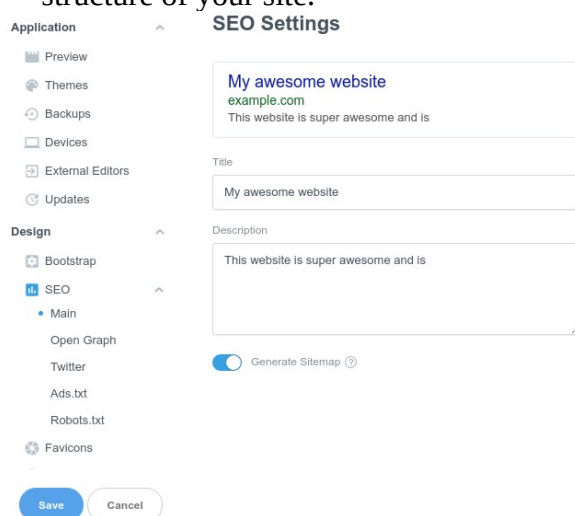
Hour 12 – Publishing, Exporting and Final Thoughts

We have reached the final hour. In this final hour we will learn to put our website online. There are essentially two ways to do this. As we will see in the first section, we can immediately publish the website via Bootstrap Studio. This is quite convenient but we lose a little control. For example, adding external documents like pdf to the website is a little trickier. An alternative is to first export the websites and then upload it to our own server manually. This offers more flexibility but requires a bit more know-how. We will have a look at this in the second section. Finally, in the last section, we will have a quick conclusion and some final thoughts.

Publishing via Bootstrap Studio

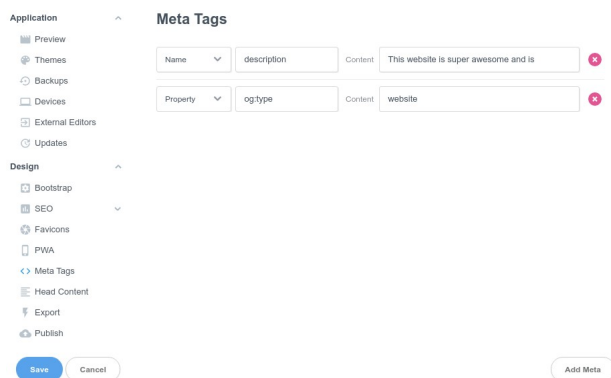
In Bootstrap Studio we can publish the website in three steps. First we should set up the meta data. This helps with search engine optimization (SEO) and with the way the website is displayed amongst the search engine results and when it's open in the browser. Secondly we need to create a new web-address or add and link an existing web-address (domain name). Finally we can add the

- Before we get started let's have a look at some useful web page settings:
 - In the top bar click on settings
 - Under the heading 'SEO' we see we can set the title and the description. The title and description can be seen on the search results page, so writing a useful title and description is instrumental in getting your target audience to click on your site.
 - We can also create a sitemap. The sitemap helps search engines get an overview of the structure of your site.

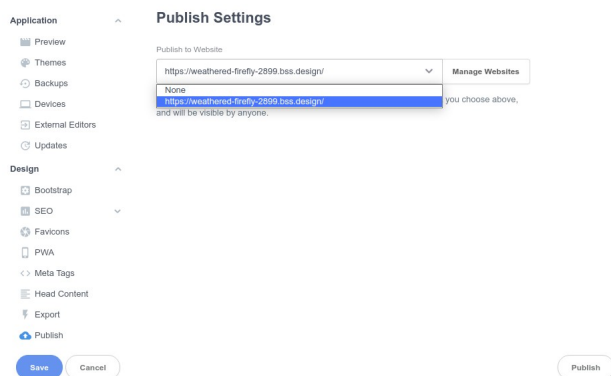


- In Favicons we can set the icon for our website.
- We can also add meta tags. Meta tags tell from information about the website to search engines. If you would like to know more about metatags and why they are useful I recommend this website: <https://www.semrush.com/blog/meta-tag/>

Bootstrap Studio in 12 Hours



- Next let's set up a domain name.
 - Select on Publish
 - Click on the button "Manage Websites"
 - Click on "Add Website"
 - We can either use a Bootstrap Studio subdomain, or we can link an existing domain.
 - If we choose a subdomain our website will end in .bss.design. We also need to choose how long to keep the domain. If we want to keep it up indefinitely we need to pick 'Never'
 - If we choose to use an existing domain we need to register with Cloudflare and create a new DNS record. To do this click on the 'Check' button and follow the instructions provided! We can find more detailed instructions here: <https://bootstrapstudio.io/tutorials/custom-domains>
 - Click on Create to continue, and then click on close.
- Now that we have a domain we can publish our website on it:
 - In the drop-down menu pick the domain we have just created, and click on publish.



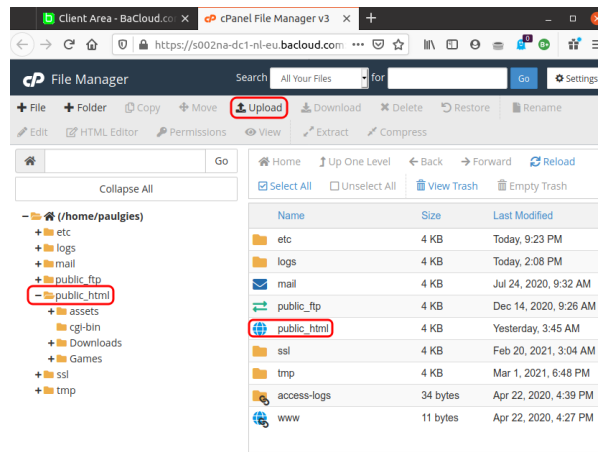
Exporting

It's a simple matter to export the project to html, css and javascript files:

- Simply click on 'Export', select the destination director and click on the export button. In case you would like to include a sitemap make sure to include the URL of you website.
- Go to the export folder to inspect the results:
 - Our html files are all in the export folder.
 - We can test the html files by opening them with a browser, or we can inspect the generated html code by opening them in a text editor.
 - We also have an assets folder. The assets folder contains all the resources the website uses including pictures, css files, javascript files etc...

Bootstrap Studio in 12 Hours

- To get this website online we need a web host. I am using a webhost called BaCloud. Other popular webhosts include Hostgator, Bluehost and many more.
- Once you have purchased a hosting plan we can upload the exported website to the server.
 - Different webhosts allow you to upload in different ways.
 - In the case of BaCloud we use the popular CPanel file manager.
 - We need to upload the exported website to the folder public html.
 - (Hint: you cannot upload entire folders directly. So if you have many folders and subfolders it's easier to compress the files to a zip file, upload it, and then extract the files where needed.)



Class Project and Final Thoughts

I hope you have enjoyed this course.

Remember, when learning a new thing it's usually a good idea to get hands-on experience to solidify what you have learned. As a project make your own website or webpage and publish it. Alternatively you could create your own advanced component. When you are done, post the link to show it off! This way you can I can give you some feedback and you can share your creation with others.

In this course we have learned quite a bit. We have learned almost everything of what Bootstrap Studio has to offer. But remember that there's still a lot of useful and fun stuff to learn out there. As lifelong learners we'll probably never run out of useful and interesting stuff to learn.

Index

100vh.....	47
animation.....	25
api key.....	26
article list.....	25
basic styling.....	14
border.....	15
carousel.....	22
column.....	28
console.....	58
container.....	31
copy components to the library.....	48
CSS.....	5
custom code.....	49
debugging.....	58
drop down.....	19
email input.....	53
embed.....	49
export.....	64
flexbox.....	45
footer.....	32
gallery.....	24
height.....	16
HEX.....	14
HTML.....	3
icons.....	26
import images.....	22
Inkscape.....	50
inline svg.....	50
javascript.....	55
map.....	26
margins.....	15
media query.....	37
navbar.....	18
override.....	41
padding.....	15
paste linked.....	21
preview.....	21
publish.....	63
radio component.....	53
RGB.....	14
row.....	28
search engine optimization.....	63
sitemap.....	63
smart forms.....	52
svg.....	49

Bootstrap Studio in 12 Hours

tags.....	3
theme.....	34
tooltip.....	25
User Interface.....	11
width.....	16
:hover.....	43
:root.....	39