Chapter: 1

**Responsive web design and web development**

**With**

**HTML5 & CSS3**

**1.1 We’ll use HTML, CSS & jQuery**

* ***Hands-on web development*** with HTML5, CSS3 and some jQuery.
* The proven 7 real-world steps from complete scratch to a fully ***functional*** and ***optimized*** ***website***
* *Simple-To-Use Web-Design* guidelines and tips :: *Standout from the crowd.*
* How to get and use amazing images, fonts and icons :: For free.
* *RESPONSIVE* web design : For all screen size.
* How to use jQuery for super cool effects like animations. :: Effects .
* How to work with Brackets :: Brackets editor.
* How to optimize your website :: Good speed performance, Search engine.

**1.2 Structure of the course:**

|  |  |
| --- | --- |
| Part 1:::  Dive into HTML   1. Our main tool: Brackets text editor 2. What is HTML? 3. The structure of an HTML document 4. Starting to fill the structure 5. Images and attributes 6. One more thing: links 7. HTML quiz   Formatting with CSS   1. Getting started with ***CSS*** 2. Starting to make our webpage pretty: ***text*** 3. ***Colors*** 4. Classes and ID'S 5. The CSS BOX MODEL 6. Building a ***simple*** ***layout*** 7. Polishing our blog post 8. ***Relative*** vs. ***Absolute*** 9. Getting started with the CHROME DEVELOPER TOOLS 10. Design challenge 1 11. CSS quiz   Part 2:::  Web-design basics   1. Introduction to webdesign 2. ***Typography*** 3. ***Colors*** 4. Working with ***images*** 5. Working with ***icons*** 6. ***Spacing*** and ***layout*** 7. ***User*** ***experience*** 8. Getting inspired 9. Summary | Part 3 :::  The killer website project   1. The 7 real-world steps to a fully functional website 2. Starting to put the 7 steps into action 3. First development steps 4. Setting up the fluid grid for responsive webdesign 5. Building the header - Part 1 | 6. Building the header - Pan 2 7. Building the header - Part 3 8. Building the features section - Part 1 9. Building the features section - Part 2 10. Building the favorite meals section - Part 1 11. Building the favorite meals section - Part 2 12. Building the how-lt-works section - Part 1 13. Building the how-lt-works section - Part 2 14. Building the cities section - Part 1 15. Building the cities section - Part 2 16. Building the *customer testimonials* section - Part 1 17. Building the *customer testimonials* section - Part 2 18. Building the *sign-up* section - Part 1 19. Building the *sign-up* section - Part 2 | 20. Building the *contact form* form - Part 1 21. Building the *contact form* form - Part 2 22. Building the footer - Part 1 23. Building the footer - Part 2 24. Making the webpage responsive - Part 1 25. Making the webpage responsive - Part 2 26. A note about web browsers   Let's add some cool effects   1. Introduction to jQuery 2. Building a "sticky" navigation - Part 1 3. Building a "sticky" navigation - Part 2 4. Scrolling to elements 5. Adding animations on scroll 6. Making the navigation responsive |
| Part 4 :::  Optimizing and launching our website   1. Final touch: creating a favicon 2. Performance optimization: site speed 3. Basic search engine optimization (SEO) 4. Let's ***launch*** our webpage! 5. ***Google Analytics*** | |

**1.3 Dive into HTML**

* HTML: HyperText Markup Language.
* HTML documents are described by HTML tags

***<h1>*** headings

***<p>*** for paragraph

***<a>*** links

* Tags MARK UP the beginning and the end of an element

**<tagname>**content**</tagname>**

* Web browsers translate HTML and CSS
* ***index.html*** is a standard name for the main HTML file of any project. HTML documents are described by elements. All HTML documents have a common structure.
* common structure:

<!DOCTYPE html>

<html>

<head>

<title></title>

</head>

<body>

</body>

</html>

* The ***<head>*** element can include a title for your web site. CSS styles, some information for the browser or for search engines and more.
* The ***<body>*** element is where all the ***visible stuff*** of your web page goes, like all the content such as ***text***, ***links***, ***images***, lists, and many more elements.
* The ***<!DOCTYPE html>*** declaration must be the very first thing in your HTML document, even before the HTML tag.
* This is technically not an HTML element, but an instruction to the browser about what version of HTML the page is written in. So with this declaration, I just told the browser that this file will be an HTML5.
* A demo: Fill the basic structure: There are 6 heading s in html. <h1> to <h6> as biggest to smallest.

<!DOCTYPE html>

<html>

<head>

<title>My First BLog post</title>

</head>

<body>

<h1>My First BLog post</h1>

<h2>This is My First BLog post</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non

proident, sunt in culpa qui officia deserunt mollit anim id est laborum. </p>

<h2>HTML is amazing</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non

proident, sunt in culpa qui officia deserunt mollit anim id est laborum. </p>

</body>

</html>

* Some default formatting:
* Comment in HTML: <!--This is a comment : -->

<**strong**> for bold text </**strong**>

<**em**> for italic text </**em**>

<**u**>underlined text</u>

* For line break use <**br**>
* Combine two tags: <**strong**>

for bold text <**em**> for italic text. <**u**>underlined text<**/u**> <**/em**>

</**strong**>

* Images and attributes:
* ***<img>:*** To put images on our webpage, we use the ***img*** tag, which actually has *no closing tag*, so it's just ***img***. There are some ***HTML elements which have only the opening tag and no closing tag***, and the ***img*** element is one of them.
* Element attributes: Element attributes provide *additional information* about an element. Because now, to indicate which image we want to display, we use the source attribute, which is ***src***. For example:

<**img** **src**="author.jpg" **alt**="Author">

* ***src*** for file location and
* ***alt*** for an alt attribute is an alternative text for the image, if the image cannot be displayed in the browser for some reason.
* LINKS: links are defined with the ***<a>*** tag, which stands for ***anchor***. And we used the ***href*** attribute to tell the browser where the link should take us. Link wont active until you use “***href***” attribute.

<!DOCTYPE html>

<html>

<head>

<title>My First BLog post</title>

</head>

<body>

    <h1>My First BLog post</h1>

    <h2>This is My First BLog post</h2>

      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

  </p>

    <h2>HTML is amazing</h2>

    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

</p>

    <strong> for bold text </strong>

    <img src="wallpaper\_blog.png" alt="Wallpaper">

    <p>Wallpaper</p> <br>

    <a href = "http://www.youtube.com">Youtube</a> <br>

    <em> for italic text </em>

<u>underlined text</u>

<p> For line break use <br> Next line appeared</p>

​

    <!--Combine two tags : -->

    <p><strong> for bold text <em> for italic text. <u>underlined text</u>  </em>

</strong> </p>

<img src="author.jpg" alt="Author">

    <p> Author : Gizkard Reventolv</p>

    </body>

</html>

* To open the link in a new tab, we need to define the target attribute and then we choose the blank options.

<a href = "http://www.youtube.com" target="\_blank">Youtube</a> <br>

* Internal links: We can also link to any document inside our current project. For example, other webpages, or even images, or other documents, and those will be internal links. So let's just add a link to the HTML image we used before.

<a href = " wallpaper\_blog.png" target="\_blank">WallPaper</a> <br>

<!DOCTYPE html>

<html>

<head>

<title>My First BLog post</title>

</head>

<body>

    <h1>My First BLog post</h1>

    <h2>This is My First BLog post</h2>

      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

  </p>

    <h2>HTML is amazing</h2>

    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

</p>

    <strong> for bold text </strong>

    <img src="wallpaper\_blog.png" alt="Wallpaper">

    <p>Wallpaper</p> <br>

    <a href = "http://www.youtube.com" target="\_blank">Youtube</a> <br>

    <em> for italic text </em>

<u>underlined text</u>

<p> For line break use <br> Next line appeared</p>

​

    <!--Combine two tags : -->

    <p><strong> for bold text <em> for italic text. <u>underlined text</u>  </em> </strong> </p>

<img src="author.jpg" alt="Author">

    <p> Author : Gizkard Reventolv</p><br>

    <a href = " wallpaper\_blog.png" target="\_blank">WallPaper</a> <br>

    </body>

</html>

**1.4 Formatting with CSS**

* CSS: Cascading Style Sheets
* CSS defines exactly how HTML looks like
* HTML is content
* CSS is style
* There are three ways to use CSS.

|  |  |
| --- | --- |
| 1. CSS code inside an HTML tag | <p style="font-size: 120%"> |
| 1. CSS code inside an HTML document | <style>  P { font-size: 120% }  </style> |
| 1. CSS code in external file | P {  font-size: 120%  } |

* ***<link> tag:*** To use an ***external*** file, we have to tell the ***HTML*** file in which ***CSS*** file the new styles will be. So to link these two ***.html*** and ***.css*** documents we use the <link> tag in the HTML head.

<head>

<title>My First BLog post</title>

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

</head>

* CSS for text formatting: CSS is written in rules. Each rule consists of a selector, and a declaration block. For example, to format the main heading, ***h1*** is the selector because it will select all ***h1*** elements.
* This means that all h1 elements will be formatted with the code we will write in the declaration block, inside ***curly brackets*** in our ***style.css*** file.

h1 {    color: green;

        font-size: 40px;

        font-family: Helvetica Neue, Arial; }

h2 {    color: green;

        font-size: 20px;

        font-family: Helvetica Neue, Arial; }

* Text color is green in this example. The property here is ***color***, and the value of the property is the name, ***green***.
* Group the common rules: We repeated some rules here. The ***h1*** and the ***h2*** have the same ***color*** and the same ***font family***. And we should actually avoid that, because repeating code is bad practice. Instead of repeating, we can just ***group the common rules*** together.

h1,h2 { color: green;

        font-family: Helvetica Neue, Arial; }

h1 {  font-size: 40px;}

h2 {  font-size: 20px;}

* Quick formatting in BRACKETS editor: Instead of switching to the ***CSS*** file, you can ***edit CSS right in the HTML file***. Just click on the P tag (or any tag), and then hit ***Command+E***, or ***Control+E***. And this opens a ***CSS editing section*** right here in the ***.html*** file. This is called Quick Edit. Now hit new rule. Now it automatically puts the ***p*** selector and the declaration block.

p { font-family: adelle; }

* Global rule: Since everything is inside the ***body***, this rule will apply to everything ***visible on the web page*** (all elements that have no defined styles). It is like a global rule. If we copy the font family right into the body, everything can have the ***Helvetica New*** font family.

body { font-family: Helvetica Neue, Arial;

    font-size: 18px; }

* Colors: In ***CSS***, ***colors*** are displayed ***combining*** ***red***, ***green***, and ***blue*** light. This is the RGB model. In the **RGB model**, each color is defined by a combination of ***red***, ***green***, and ***blue*** using a hexadecimal notation.
* The lowest value for each color is zero, which is ***0*** in hexadecimal, and the highest value is 255, which is ***ff*** in hexadecimal.
* For example, the complete red has ***ff***, which is the ***highest*** value in the ***red channel***, and ***zero*** in the ***green*** and the ***blue*** channel.
* ***Yellow*** is as mixture of ***red*** and ***green***, so it has ***ff*** in the ***red*** and in the ***green*** ***channel***, but ***zero*** in the ***blue*** channel.

|  |
| --- |
| C:\Users\RTTI\Pictures\color2.JPG |

* So by mixing red, green, and blue together using different values between zero and 255, we can obtain over 16 million colors.
* For instance, darker green is represented with this hexadecimal code ***1da717***. So we have ***1d*** for ***red***, ***a7*** for ***green***, and ***17*** for ***blue***. Mixing these together makes this nice green.
* For ***gray*** colors, just make ***red***, ***green***, and ***blue*** take the same value. For instance, ***8b8b8b*** gives us ***a shade of gray***. If all six values in a ***hex code*** are the same, like ***555555***, we simply write ***555***.

|  |  |
| --- | --- |
| * Alpha: CSS colors can also have ***transparencies***. If we want colors for transparencies, we do not use the ***hexadecimal notation***. Instead, we write them like you can see on the ***green square***. This gives us the same green we saw before, but with ***75% transparency***. * Color picker in BRACKETS editor: let's try to change that ***green color*** of the headings we defined before, because we will actually never use color names like ***green***, as we did previously. So put the cursor here on this ***color***, and then hit ***Ctrl + E***, just as we did before in the HTML file. | C:\Users\RTTI\Pictures\gretr.JPG |

* This time as nice color picker will appear. We can use ***sliders*** to change the color hue and transparency.
* With this slider, we can adjust the transparency. As we saw before, this now has this RGBA notation, and here we have the transparency value. Now I will just put this back up to 100% and change to the hexadecimal notation as before.
* CLASS and ID selectors in html tag: Now we discuss about two new selectors: ***classes*** and ***IDs***. We used ***h1***, ***h2***, ***p***, and ***body*** selectors, ***to select the corresponding HTML elements***. But now, suppose you wanted to style only one of the paragraphs, not all of them. That's why we have ***classes*** and ***IDs***. We can ATTRIBUTE ***class*** or ***ID*** ***names*** to elements and then use these ***classes*** or ***IDs*** to select them in the ***CSS*** code.
* Same class can be ***attributed*** to as many elements as you like, while an ***ID*** can be used ***only once*** inside each ***HTML*** document. For example, we give class name “text” to our two paragraphs tag ***<p class="text1">*** and an id “author” to author name ***<p id="author">***

<!DOCTYPE html>

<html>

<head>

<title>My First BLog post</title>

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

</head>

​

    <body>

    <h1>My first BLog post</h1>

    <h2>This is My First BLog post</h2>

        <p class="text1">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

  </p>

    <h2>HTML is amazing</h2>

        <p class="text1">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod </p>

    <img src="wallpaper\_blog.png" alt="Wallpaper">

​

        <p>Wallpaper</p> <br>

    <a href = "http://www.youtube.com" target="\_blank">Youtube</a> <br>

​

<img src="author.jpg" alt="Author">

    <p id="author"> Author : Gizkard Reventolv</p><br>

    <a href = " wallpaper\_blog.png" target="\_blank">WallPaper</a> <br>

    </body>

</html>

* Let's go back to our CSS file. And here, down here, we can now add the new selectors. So, style the paragraphs with the ***class="text1"*** class. We'll add dot, ***.text1{ }***

.text1{text-align: justify}

* To select that ID, we use the hash ***#*** symbol. For example, to select ***id="author"*** use ***#author{}***

#author{font-size: 22px;

        font-family: Fira Code;}

* Using IDs is actually not a good practice, because, they can only be used once.

|  |  |
| --- | --- |
| **1.5 Box Model**  All HTML elements can be seen as boxes. What the box model does is to allow us to define space between elements and to add a border around elements.   * The box model consists of ***margins***, ***padding***, ***borders*** and actual ***content*** of the box. For instance, ***text***. * This image shows very well how the box model works. | C:\Users\RTTI\Pictures\Box_model.JPG |

1. ***Content*** is where text and images appear.
2. ***Padding*** is like a ***transparent area*** around the ***content***, but *inside of the box*.
3. The ***border*** of the box goes around the padding and the content. It ***may be transparent, or not***.
4. And finally, the ***margin*** is the space between boxes.

* ***Padding***, ***margin*** and ***border*** are ***properties*** in ***CSS***, and can be specified for the ENTIRE BOX or for INDIVIDUAL SIDES. ***Top***, ***bottom***, ***left*** or ***right***. We'll use all of them later.

The box model also allows us to set the height and the width of an element. ***The standard box model let you just set the*** height ***and the*** width ***of the*** content***. Not of the*** entire box itself. This means that ***padding***, ***margin*** and ***border*** will be added to the ***height*** and ***width*** we specify.

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
|  |  |