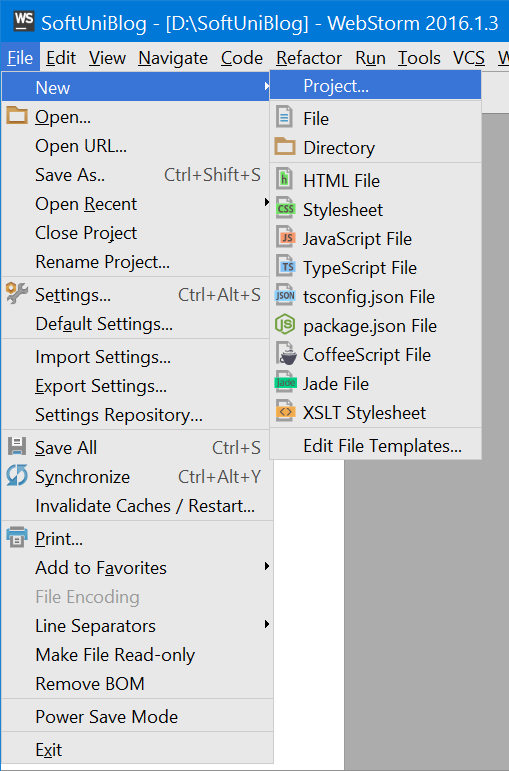
# Exercises: HTML5 and CSS

Problems for exercises and homework for the [“Software Technologies” course @ SoftUni](https://softuni.bg/courses/software-technologies).

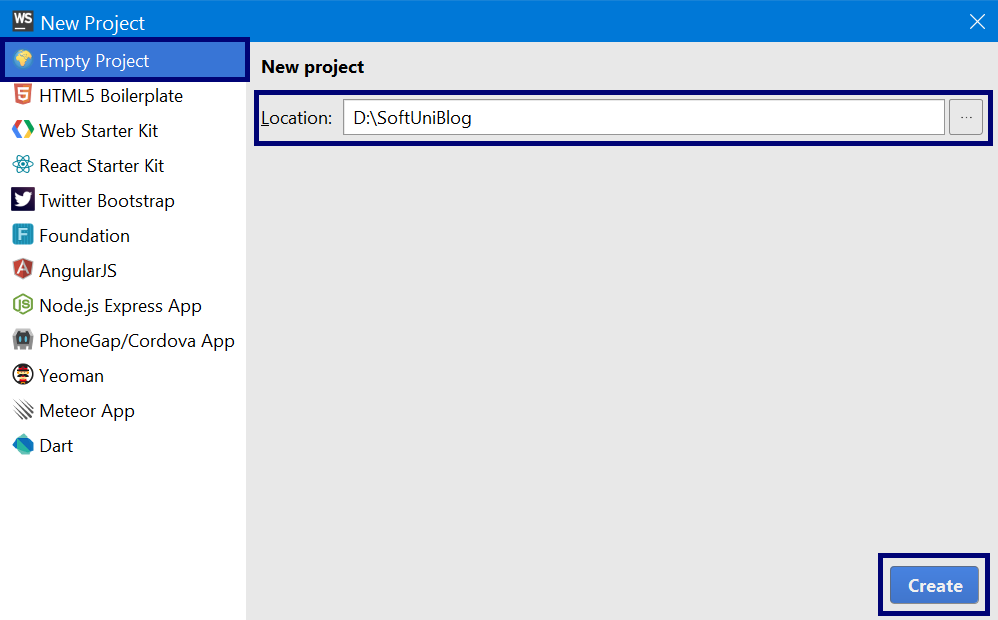
After following the steps for this exercise, you will receive the base for our blog. Similar to this you will also receive other labs, that will help you finish the blog.

## Create New Project

Open WebStorm and create new project from **File -> New… -> Project**

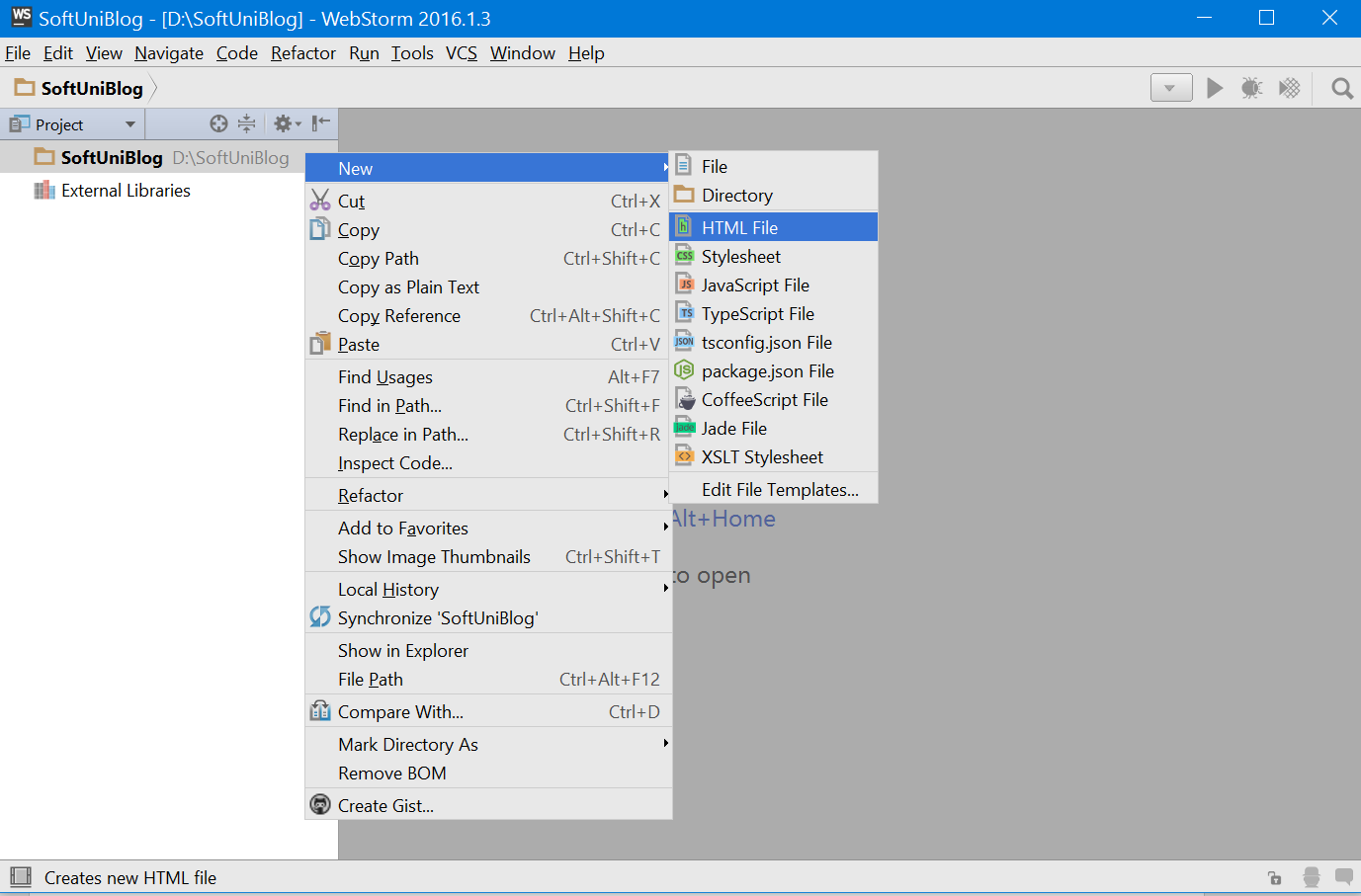


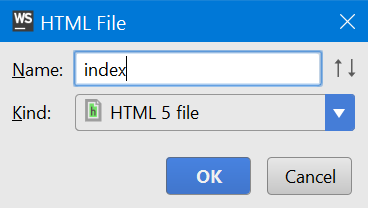
Now select **Empty Project** and choose a **destination folder** of your project



## Add Blog Home Page

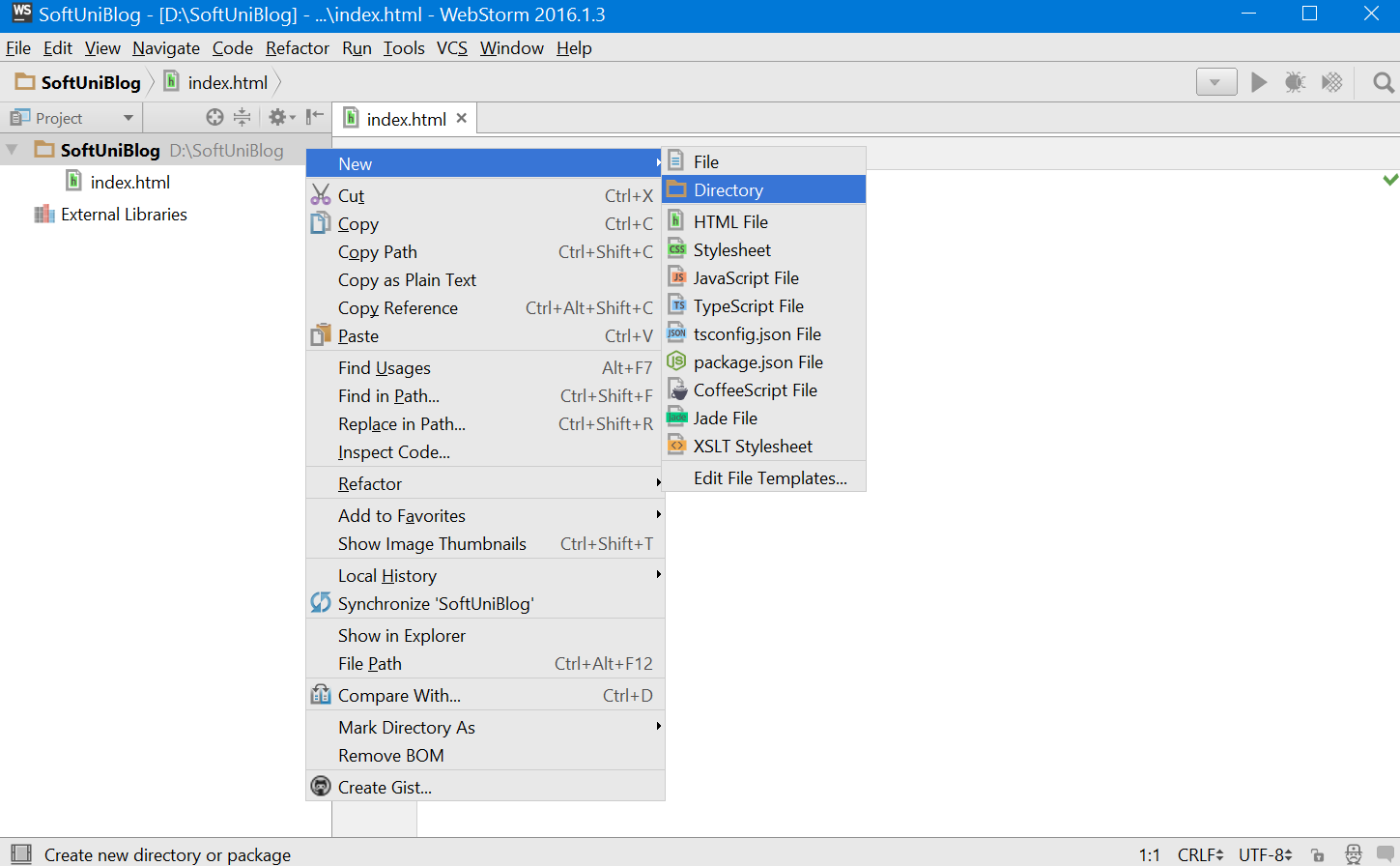
Create new HTML file called **index.html** that will be our **home page.**

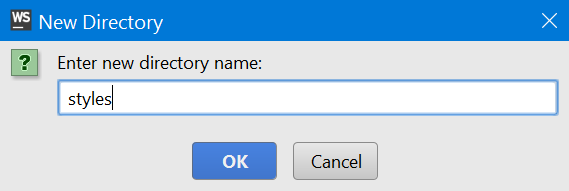




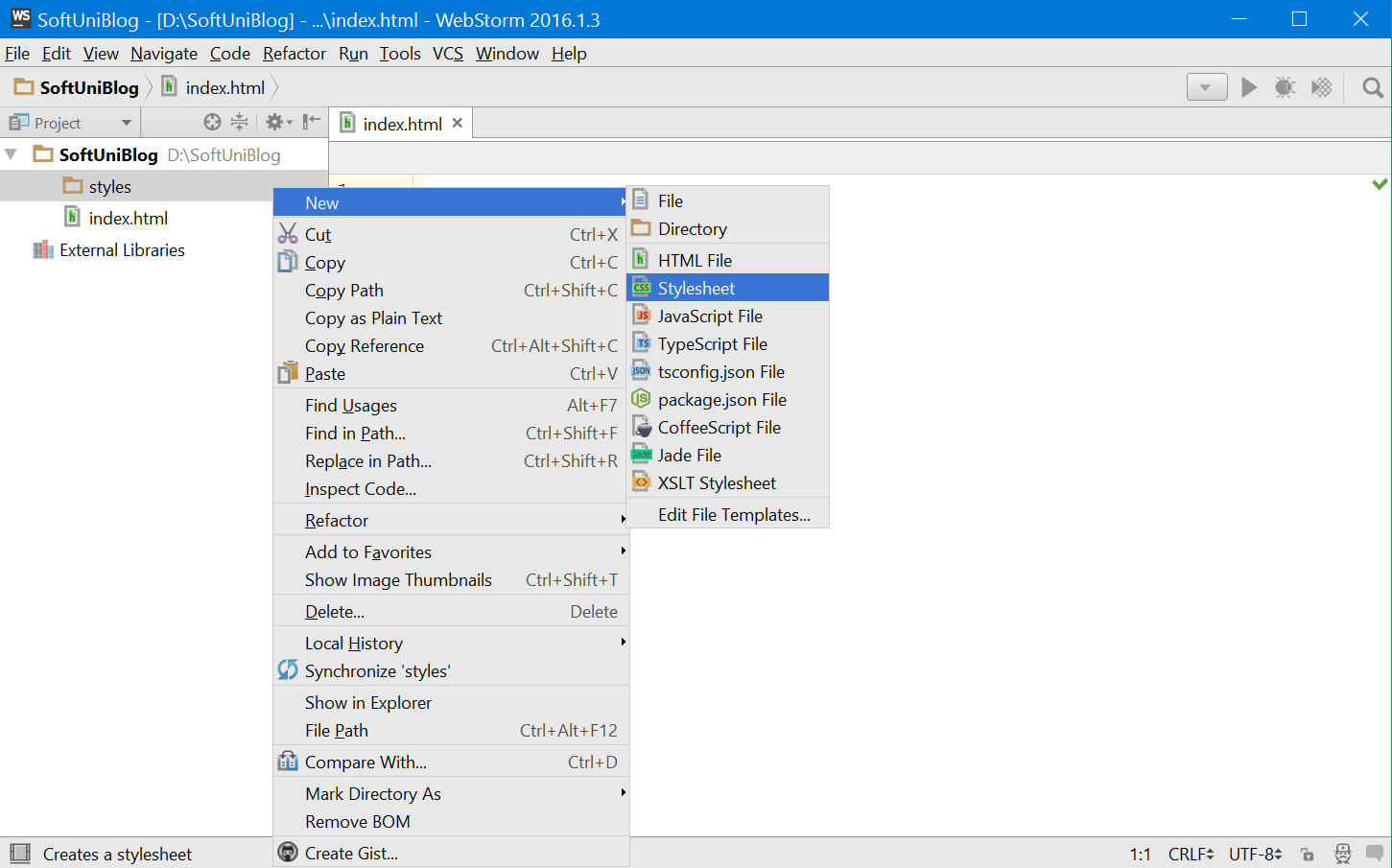
## Add Blog Styles File

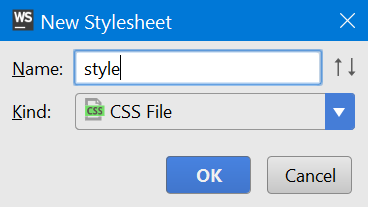
Create **new directory** in our project that will keep our **CSS files** and name it **styles**





Create a new **CSS file** with **styles for the blog** and **add** it **to the project**

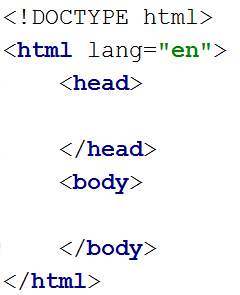
****



## Create Basic Structure

Now it’s time to start writing some code. Create the **basic structure of a html** document:

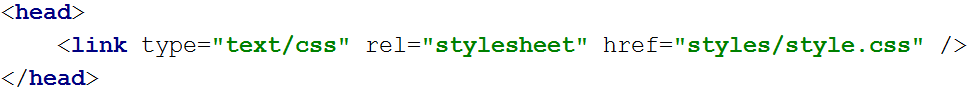
* **<!DOCTYPE>** – it’s an instruction to the web browser what version of HTML our page is written in. For HTML5 we should declare to be **<!DOCTYPE html>**
* **<html> –** this tag is the container of all other HTML elements in the document
  + **<head>** - this tag will contain the title of the document, styles scripts, meta information and more.
  + **<body>** - here we can put everything that we want to displayed in the browser



## Create Connection between HTML and CSS Files

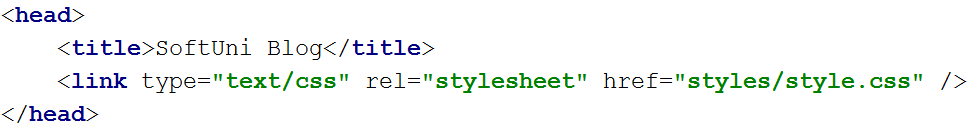
Now we have the **HTML file** where will be the **structure of the blog** and the **CSS file** that will **make the blog look pretty**. But the **HTML file** **does not know** that the **CSS file exists**. So let’s link these files. This can be done in the <head> part of the HTML file using <link> tag with **appropriate attributes**.

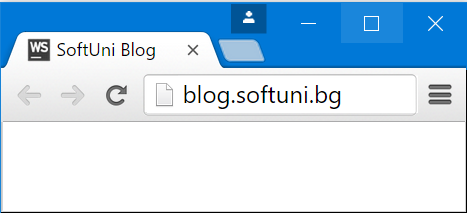
* **type** = text/css
* **rel** = stylesheet
* **href** = styles/styles.css

****

## Change Blog Title

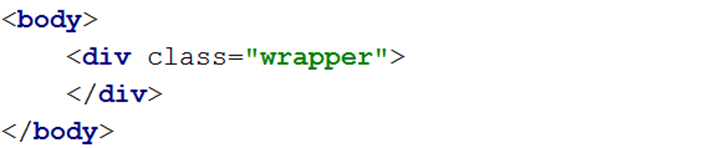
Let’s give our page more **appropriate title** that will be displayed in the browser’s tab.





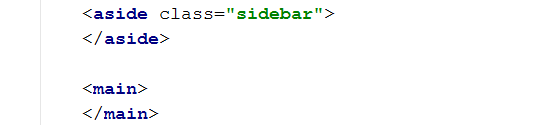
## Create a wrapper for the main content of the page

The page content is usually located in the **body** part. However, just putting the content in there plainly is not very appropriate. This is why we will wrap the content in a single block, which we will call – a **wrapper**. Create a **<div>** tag and set its class attribute to – “wrapper”.

****

## Create the necessary structure in the content wrapper

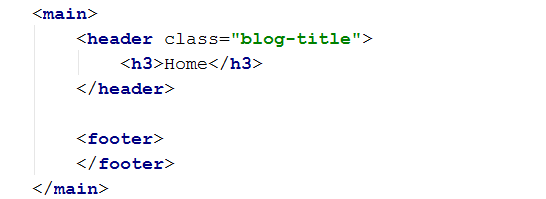
Now that we have a content wrapper, we can start forming the overall structure of the content. We are going to have a sidebar and main content sections on the main page. That is why we create an **<aside>** tag and a **<main>** tag in the wrapper, to divide the main page into the two needed sections.



Set aside’s class argument to “**sidebar**” to clarify that that is our sidebar element.

## Create the necessary structure in the main tag

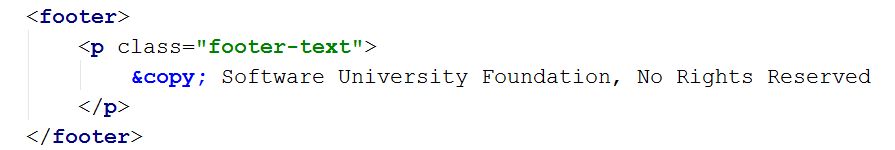
When we have the main content structure we can start building up the content. Usually pages consist of 3 main components – header, content and footer. Let’s create the header and footer sections for our main content.



After we’ve written the **<header>** and **<footer>** tags, we set the header’s class argument to **“blog-title”**. Then we add a heading **<h3>** tag with “Home” text in it. This will be our home page’s header.

## Creating the footer content

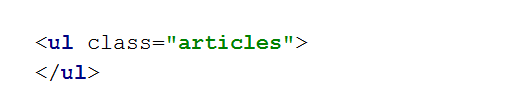
We should create an appropriate footer content too. Usually the footer consists of copyright specifications, so let’s put the Software University copyright specification.



Set the paragraph tag (**<p**>) class to “footer-text”, so that we know that that is the footer content. “&copy” is the code for the copyright symbol according to the URL encoding.

## Creating the main content of the page

Now that our home page has a header and footer, what is left is to create the content of the page. Our page will hold information about several posts, which is why we need to create a list. The order of the elements in that list is irrelevant to us, that is why we can use unordered list (**<ul>**).



Set the ul’s class to “articles”.

## Create a single post’s content

Let’s add one post to our list of posts. Write the following structure in the ul:

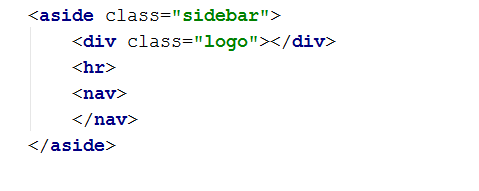


We have created a list element (**<li>**) and we have put an **article tag** in it, which clarifies that the current element is an article. Set the li’s class to “**single-post**”. The article consists of several elements. The **div element** with class “**dot**” is what will be the list element bullet. We will make a custom list style that is why we need it. The **heading** with class “**title**” is the title of the current article. Then we have a secondary title – the **p element** with class “**subtitle**”. And lastly – the **p element** with class “**content**” which points to the article’s content (the content in the picture above is flattened).

Now, create the other posts in the ul list, by following the same logic, as the one in the picture, shown above.

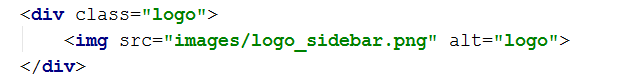
## Creating the side-bar structure

Now that our main content is ready, we can concentrate on the aside section. Usually the side-bars of the web pages hold a navigation bar with links to other pages. Our side bar will contain a logo too.



We create a div with class logo to hold our page’s logo, a **<nav>** tag to hold our navigation bar, and a **<hr>** (horizontal line) to separate them.

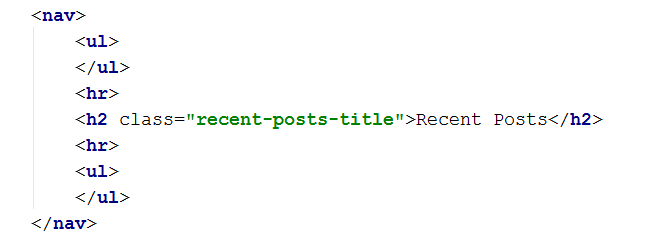
## Loading the logo image



Put an **<img>** tag to hold the image for the logo, and set its “**src**” attribute to point to the “**logo\_sidebar.png**” picture in the “**images**” you received. Set the “**alt**” attribute of the img to “**logo**”. The “**alt**” attribute is an **alternate text** which will be rendered on the webpage, in case the picture does not load correctly.

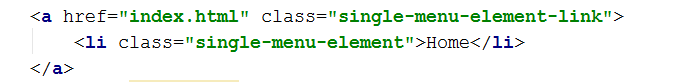
## Filling the navigation bar

The navigation bar will hold multiple links which will navigate the user through the web application. Let us create 2 lists for the navigation structure – one for the main navigation links and 1 for navigation between the posts on the page.



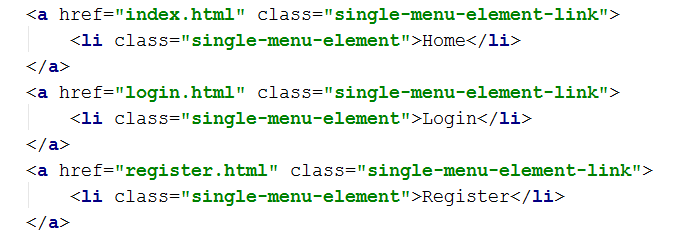
Separate the lists with horizontal lines, and put a heading between them. That heading’s function will be to divide the main navigation from the recent posts navigation. Set the attributes and content of the heading exactly as specified above.

Let’s create a single link for the main navigation ( the first list ):



The **<a>** tag is used to create hyperlinks in an HTML code. The **href** attribute is used to specify the URL to which the Web Browser will go after the click on the link. Follow the structure, specified above, exactly as it is written, with the same classes and content, for maximal accuracy.

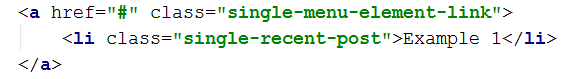
Let’s add the other two links of the main navigation, and finalize it.



And with this we finish our main navigation between the web pages.

Now let’s fill the navigation between the posts on our page.

Use the same logic as shown in the picture above for the links – with <a> and <li> tags, and make 5 post links in the secondary list to control the navigation between the posts. Set the <a> tags’ href attribute to **“#”** which is basically a link to the current page. The posts don’t need to change the page. Also, set the inside <li> element like this:



Follow exactly the same pattern as specified above for all 5 links you make. Set the <li> element’s content of the different elements to: “Example 1” “Example 2”… and so on.

And with that we have finished the design of the HTML code on our page. In other words, our architecture is ready. Now for the hard part – the styling.

# Creating Styles with CSS

## Resetting the CSS

The first thing you should do when styling web-pages, is to reset the CSS. Web Browsers have their own default styles for some tags and elements and they tend to override the written CSS sometimes. Go to <http://meyerweb.com/eric/tools/css/reset/> and copy the css code there and put it in “style.css” above everything else. Then you can be sure you are writing on zero styling and nothing can override your own styles.

## Adding the custom fonts

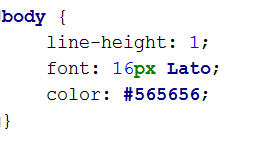
We will need some fancy custom fonts for our page, so we will import them. Write the following **above** the reset css code.



This will give you access to the two fonts **Lato** and **Oswald**.

## Styling the page’s body

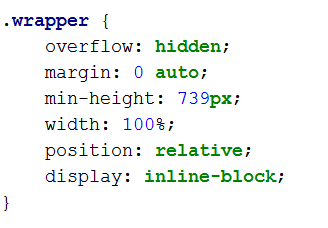
Now that we have reset the styling we can start. First we will create the style for the page’s body.



As you know CSS works like key-value pairs. Here, for example, we set the **line-height** to **1**. **Multiple values** to one key is allowed in CSS, like the font is set in this example. The color on the other side is a **hex** code. Follow the instructions specified above exactly, in order to achieve maximum accuracy.

## Styling the content wrapper

Besides the body, we must apply styling to the content wrapper itself too.



First we see the overflow property. It specifies what should happen with an element if content overflows another element’s box.

The margin property is used to create space between elements. It is mainly used for layout manipulation. The margin property specifies the space **outside** the border.

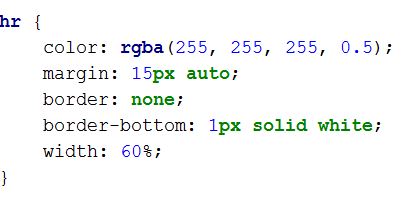
There is also the **min-height** property. Along with the **max-height**, **min-width**, **max-width** it specifies exactly what its name means.

Then we have width – along with the height, these properties specify the two dimensions of a certain element. Most blocks in HTML are rectangles and own width and height properties. It can be set with pixels, and percentages. The percentage specifies how much percent of the maximum width / height for the current element, will the current element’s width / height be.

After that we see the **position** property which specifies the position of the current element on the screen. The **relative** value specifies that the current element will be positioned relative to its original position.

Lastly we see the display option which specifies the way an element must be rendered on the web page. Currently its set to an “**inline block**”.

## Styling the horizontal line



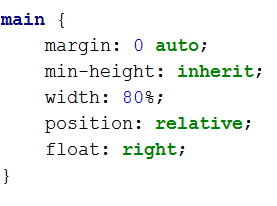
Here we have a different way of setting the color – using the **rgba** function. We are also specifying settings such as **margin**, **border** and **width**.

The margin property is seen again.

The border property specifies the current element’s main border style. In the current example we also have border-bottom which specifies only the bottom side of the border of a certain element. There are also ways to stylize the top, left, and right side of the border.

Then we set the width of the element and we end the section with the styling for this element.

## Styling the main content section

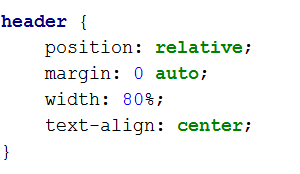


Now let’s put an abstract style for the main content. As we see the margin property exists in this context too. The min-height property is seen again here. In this current case we have an interesting value – **inherit**. This specifies that the current property will **inherit** its style from the parent element of the current element. In the current case the main element is in the **wrapper** element so it will inherit from it. Then we have a width specification.

Then we see the float property. It specifies whether a current element should float, and if so, to what side.

## Styling the header element

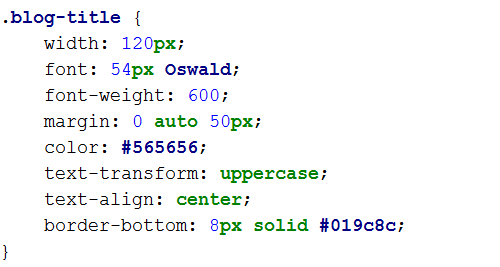
After introducing the abstract main content design, it is time to start with the insides. First, of course, is the header.



As we see most of the properties we’ve been introduced to, by now, are frequently met in the styling of different elements. Here we have a **text-align** property which specifies the position of the text in a certain element.

## Styling the home page’s header element

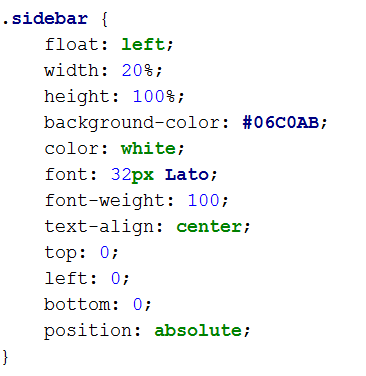
After we’ve applied an abstract design for the header element, we need to concretize the style a bit more for the case we need. Let us start using the classes and ids, we’ve put on the HTML elements, as selectors. The home page’s header’s class was “blog-title”. So, in the CSS, we select all elements with that class in the following way:



Classes in CSS are selected with a **dot** in front of the class name. As we see the width of the home page’s header’s width is hard set. Then we use one of the custom fonts we imported in the beginning of the styling. After that we set the font-weight. As the property’s name itself speaks, it sets the text weight. Most of the other properties are already familiar. Let’s turn our attention at the **text-transform** property. The text-transform property provides the ability to transform the text in the content of the selected tag. In our case, it’s “uppercase”, which capitalizes all letters.

## Applying abstract styles to the side-bar

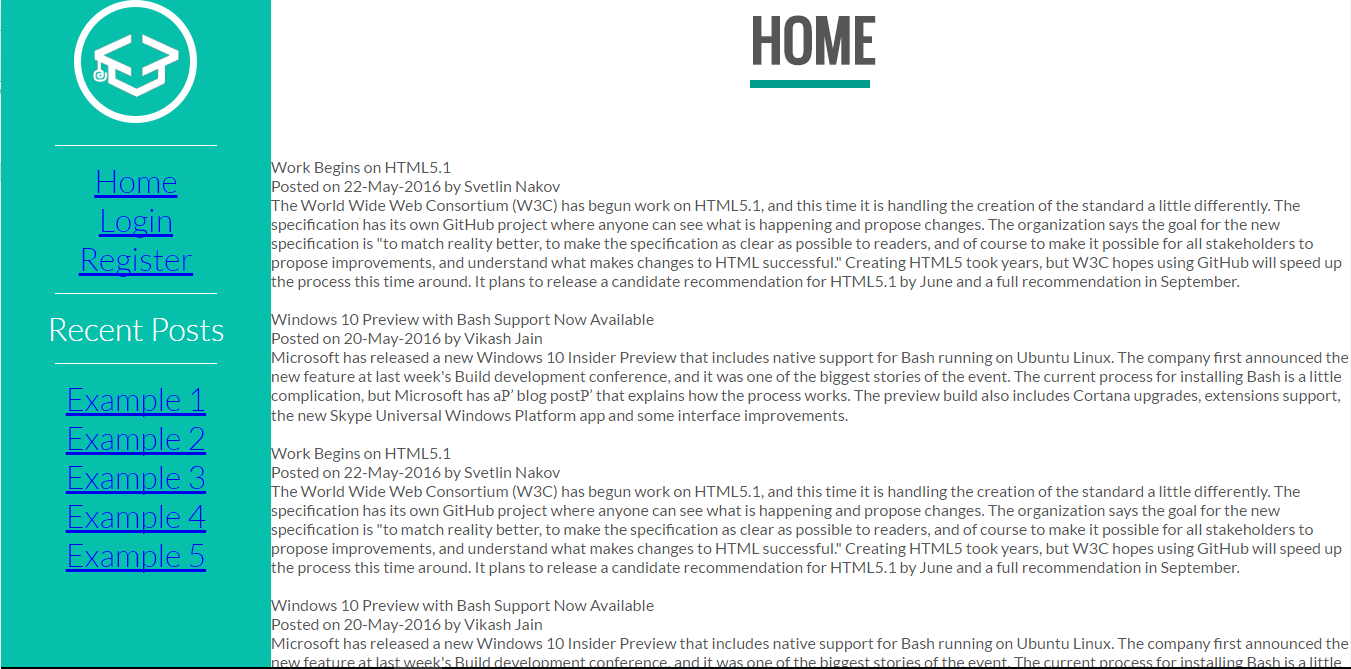
Now that we have stylized some of the things in the main content’s structure, we can put an abstract design on the sidebar so that we bring some layout to the page. Let’s select the sidebar section by its class.



As we see the sidebar’s styles aren’t that different from the other elements’, except it’s a bit complex, but that is necessary. After all it’s a big part of the page. The only new things we see here are the properties **top**, **left**,and **bottom**, there is also one we don’t see – **right**. Those 4 are positioning properties, which are used for manual position specification. They are used to set the margins from the 4 sides.

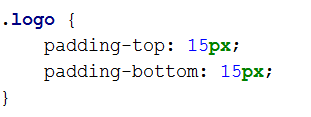
## Revision

Now that we have applied some abstract styles on the structure we should be able to see the following layout:



## Styling the sidebar logo

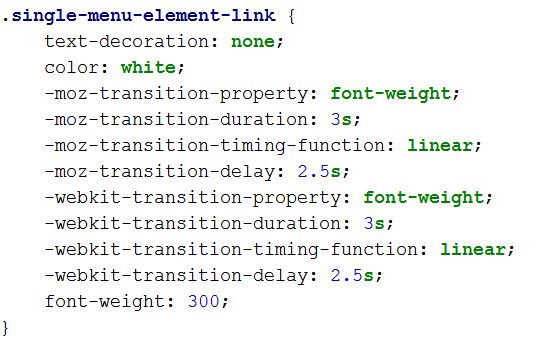
Now let’s start styling the sidebar. We should start with the logo, because it works as the sidebar’s header.



Here we see the padding property, along with its concretizing properties for sides. The padding properties are used to generate space around the content.

## Styling the links of the navigation bar

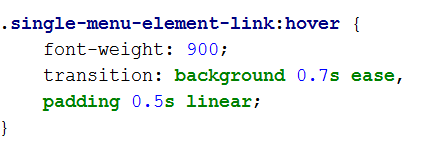
Here it gets interesting. The links should be a little more dynamic. They should have the so called **hover** effects, which are normally done with transitions. Transitions are special effects that affect certain properties of certain elements for certain amounts of time, thus creating creative animations. Let’s see how that’s done.



We have nullified the text-decoration of the <a> tags, because by default the <a> tags put a specific decoration on their content to clarify that they are hyperlinks. We change the color to white. Then we start creating the transitions. As you can see a transition has a **property** it affects, a **duration** for its effect, a type of its **timing function and** a **delay**. Notice that there are strange things before the properties. Those are special prefixes that specify for which Web Browser is the current styling, because different browsers define different styles.

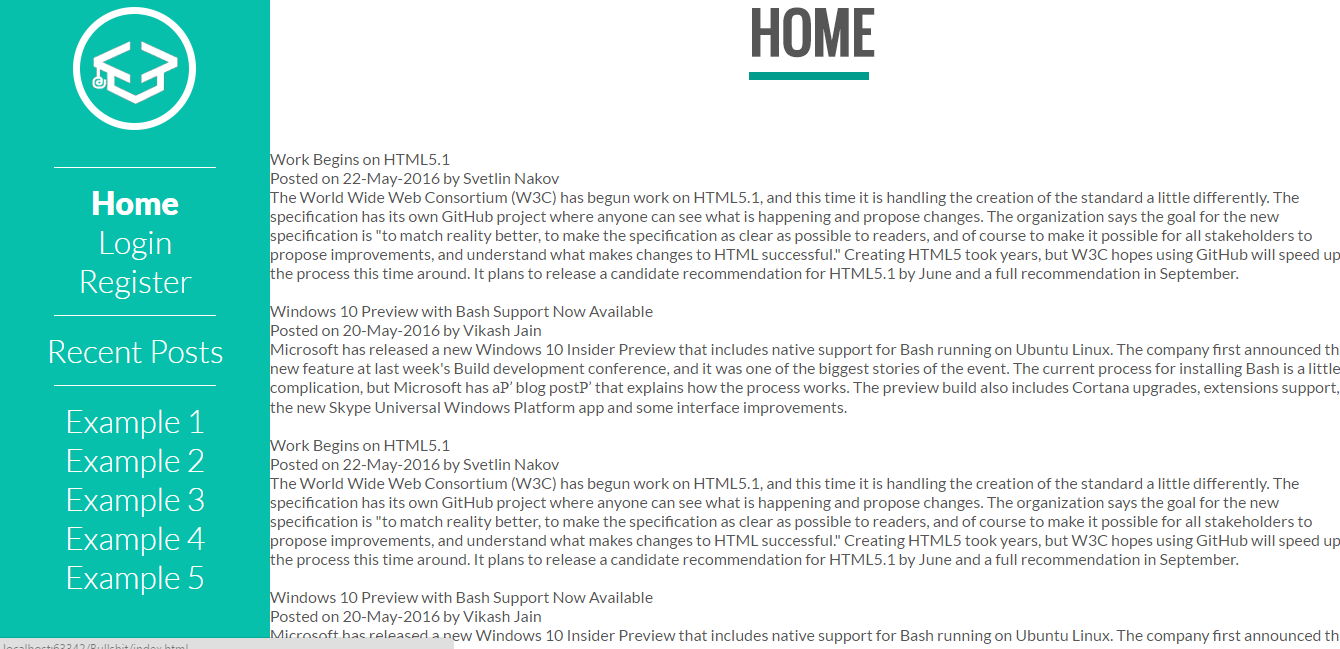
Notice how we set the default **font-weight** to 300.

With that we are ready with the static information about the link. Now let’s add the hover effect itself.



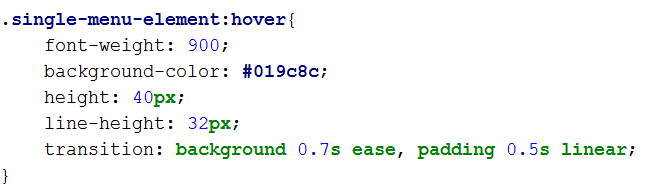
The effects on a certain tag are defined like, on the picture above. See how we define a new font-weight. That font-weight will be set on hover of the element. However the transitions we initialized back then will affect this change. They will make it smooth. There is always a back-transition on the hover effect, which gives us double transition –forwards and backwards, which gives even more design to the transitions.

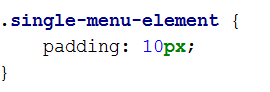
If you’ve done everything in the steps specified above you should see the results as follows:



On hover the text changes and gets **bolder** than before, which was the main idea of the transition. Notice how the back transition works, overtime.

Let’s add some more effects to the current links.

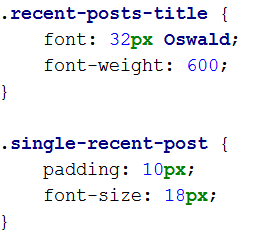




Pay attention to the selectors, they are different.

## Style powder to the Recent Posts

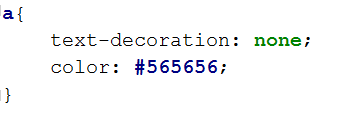
Now that we’ve styled most of the links lets add some styling to the Recent Posts title and some more paddings and fonts styles to concretize the recent posts links.



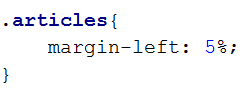
## Styling the main section’s content

The time has come for the main section to be styled. Create a new file main-styles.css and import it to the header of the index.html the same way as you added the style.css. Extracting particular styles into multiple files is good for the overall structure.

Let’s add some styles to this new file. First we will give the abstract styles to the hyperlinks:



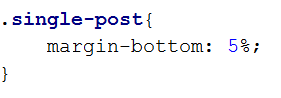
Followed by a simple styling for the articles list that holds the posts in the content of the page:



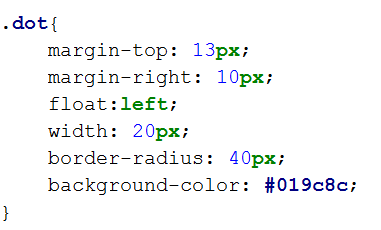
## Styling the posts

Styling the posts, and their sub-elements is the next step in the styling of this page:

Some general styling for the posts:

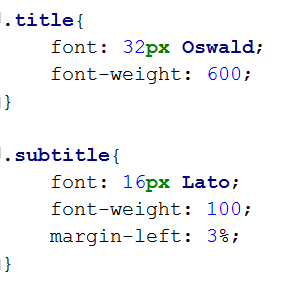


A while ago we said that we are going to create a custom style for our list of posts. This is where it happens. Those **div**s that were supposed to be our list item bullets are finally getting their styling.

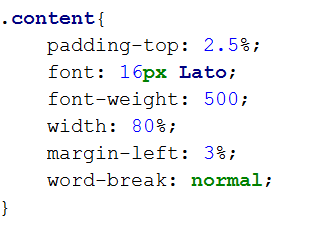


We have spotted a new property – the border-radius. The border-radius basically specifies the rounding level of a certain element. In other words, how round will it be? In the current case the dot will be almost perfectly round, like a circle.

Next, come the title and subtitle of the post. Let’s stylize them too with some simple fonts and margins:



Last but not least, comes the content of the post:



Here we see an interesting property – the **word-break** property. This property specifies line breaking rules – whether a word should be broken into two pieces and divided into two lines or not.

With that we finish the posts styling.

## Styling the footer of the main page

Last for this page, are the footer styles, which are very simple.

