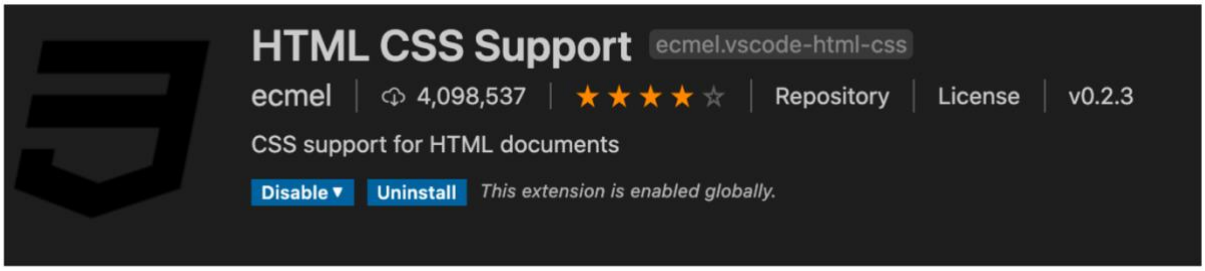
F28WP Web programming
Lab Sheet 3

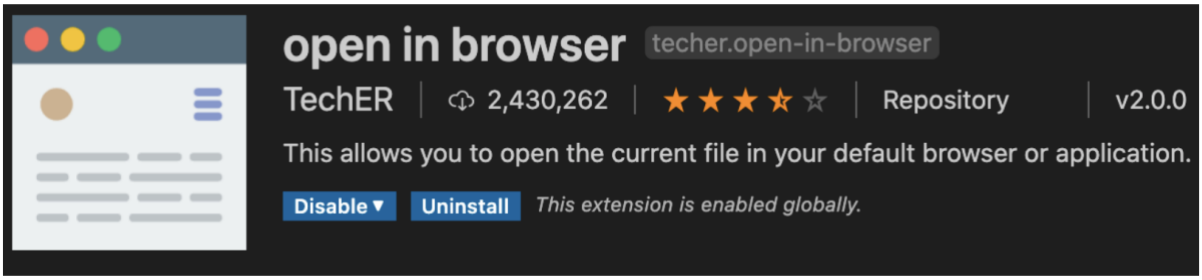
HTML and CSS

Activities in this lab session will help you understand the core of HTML and CSS. The lab is structure into two activities. First, you create a personal web page and store it under GitHub. The content and structure described in this lab sheet are given as examples. You can change the content and the structure to suit your needs.

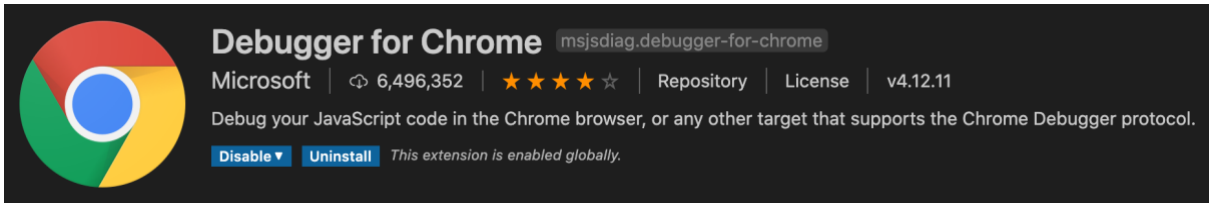
We are going to use Visual Studio code (VSC) to edit code (but you can any other editor you are used to). To get auto-completion and other functionalities while typing HTML/CSS (in VCS), install the extension HTML CSS Support



To visualize HTML pages from VSC, you should install the extensions **Open in browser**. To open a html page in your preferred browser, click Option-b on mac (or alt-b on Windows).

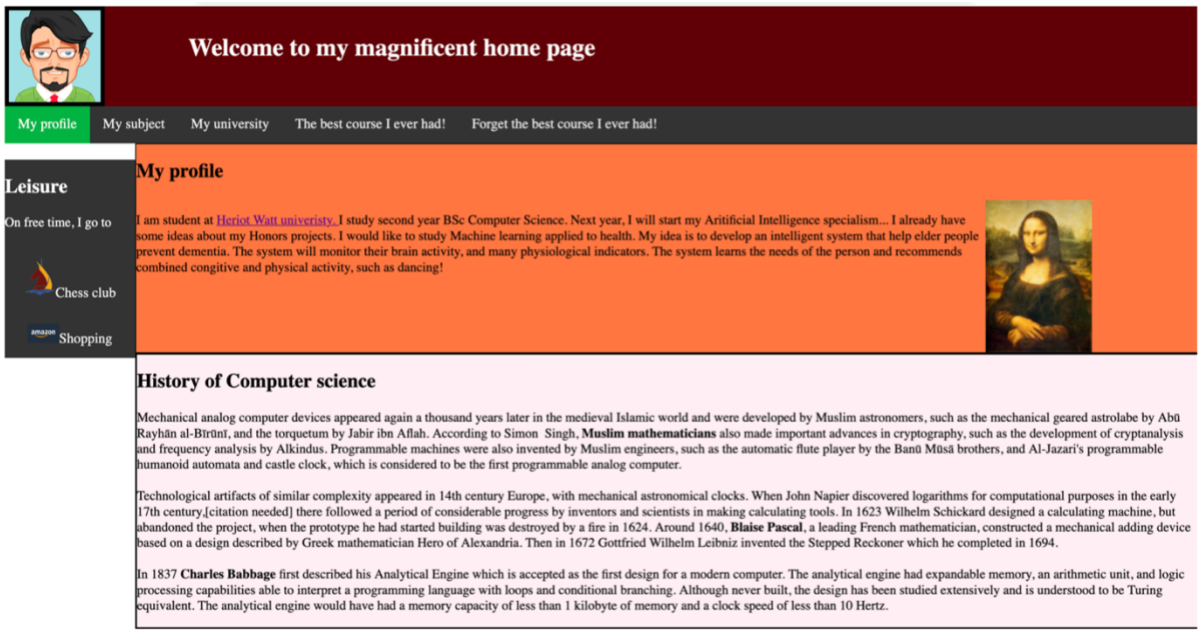


You can also install Debugger for Chrome to debug (find errors in) your JavaScript

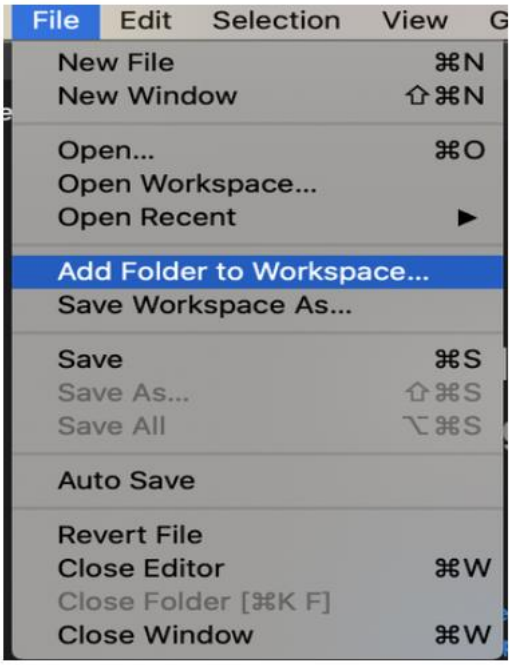


2.1 Design a web page

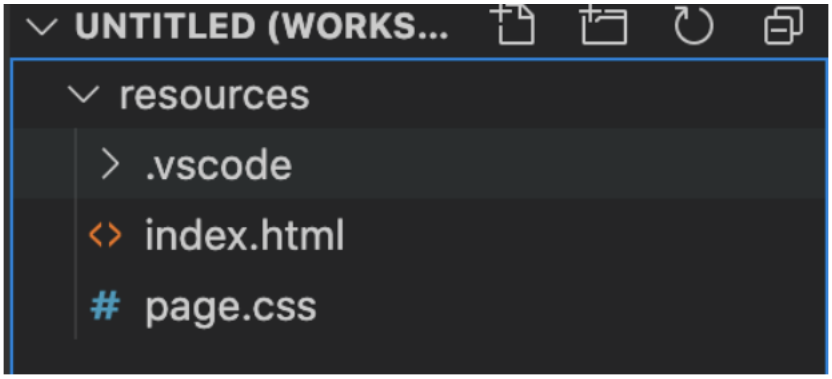
This first activity allows you to develop a web page using HTML/CSS, by taking example from a simple existing example web page.
The page should have the following structure, and example can be viewed at https://hbatatia.github.io/examplewebpage/. The HTML/CSS/JS code of this web page is accessible from GitHub at https://github.com/hbatatia/examplewebpage. You can clone this repository and open code to study it. Note that only index.html and page.css are the actual files of the web page. Files with names pagex.html and pagex.css are the versions that you work with at different stages of this lab.
However, you should work iteratively on creating your own page to avoid confusion and understand the different elements.



1. Try the functionalities of this page by moving the cursor around and clicking the different locations and links. You are going to create a similar page, using your own content and layout.
2. Create a folder named “Home page” in your home directory
3. Start Visual studio code
4. Select “Add folder to workspace”

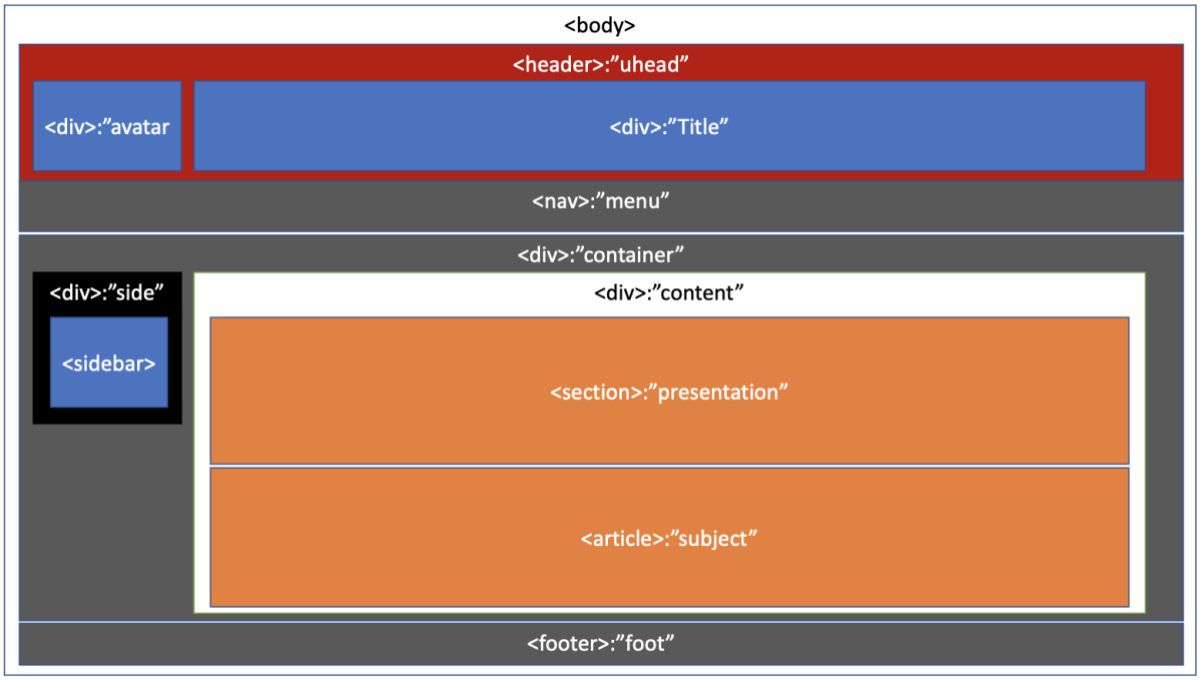


5. Select the folder “Home page” that you just created
6. Add two new files: index.html and page.css



Structure of the web page

The web page that you are recreating has the following structure. Note that different types of tags have been used: body, header, nav, div, section, article, sidebar, footer



Hint: To create this structure (without the presentation), you edit index.html and type in the following HTML code (file page1.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>First Webpage</title>
</head>
<body>
    <header id="uhead">
        <div>
        </div>
        <div id="Title">
        </div>
    </header>
    <nav class="menu">
    </nav>
    <div id="container">
        <div id="content">
            <section id="presentation">
                <div id="profile">
                </div>
                <div id="picture">
                </div>
            </section>
            <article id="subject">
            </article>
        </div>
        <div id="side">
            <aside class="menu">
                <div id="links">
                </div>
            </aside>
        </div>
    </div>
    <footer id="foot">
    </footer>
</body>
</html>
```

Open the page in the browser (alt-b). What do you notice? Display the source code and compare it to the content of index.html. Notice that the page is empty, source code contains only the structure tags, there is no content!

Content of the web page with HTML

In this activity, we will add content (text, images) to the web page.

1. Edit index.html
2. Type in the **content** of your page using HTML. You can use another content than the following (page2.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>First Webpage</title>
</head>
<body>
    <header id="uhead">
        <div>
            <img id="avatar" alt="my avatar" src="https://pickaface.net/gallery/avatar/unr_admin_171016_2225_zewhi.png">
        </div>
        <div id="Title">
            <h1 id="myh1">Welcome to my magnificent home page</h1>
        </div>
    </header>
    <nav class="menu">
        <a class="active">My profile</a>
        <a href="#subject">My subject</a>
        <a href="http://www.hw.ac.uk">My university</a>
        <a href="https://f28wp.github.io"> The best course I ever had!</a>
    </nav>
    <div id="container">
        <div id="content">
            <section id="presentation">
                <h2>My profile</h2>
                <div id="profile">
                    <p>I am student at <a href="http://www.hw.ac.uk">Heriot Watt univeristy. </a> I study second year BSc
Computer Science. Next year, I will start my Aritificial Intelligence specialism... I already have some ideas about my Honors
projects.
                        I would like to study Machine learning applied to health. My idea is to develop an intelligent system
that help elder people prevent dementia. The system will monitor their brain activity, and many physiological indicators. The
                        system learns the needs of the person and recommends combined congitive and physical activity, such as
dancing!</p>
                </div>
                <div id="picture">
                    <img id="photo" src="https://tineye.com/images/widgets/mona.jpg">
                </div>
            </section>
            <article id="subject">
                <h2>History of Computer science</h2>
                <p>Mechanical analog computer devices appeared again a thousand years later in the medieval Islamic world and
were developed by Muslim astronomers, such as the mechanical geared astrolabe by Abū Rayhān al-Bīrūnī, and the torquetum by Jabir
                    ibn Aflah. According to Simon  Singh, <span class="bold">Muslim mathematicians</span> also made important
advances in cryptography, such as the development of cryptanalysis and frequency analysis by Alkindus. Programmable machines
                    were also invented by Muslim engineers, such as the automatic flute player by the Banū Mūsā brothers, and Al-
Jazari's programmable humanoid automata and castle clock, which is considered to be the first programmable analog computer.
                    <BR>
                    <BR>Technological artifacts of similar complexity appeared in 14th century Europe, with mechanical
astronomical clocks. When John Napier discovered logarithms for computational purposes in the early 17th century,[citation
needed] there
                    followed a period of considerable progress by inventors and scientists in making calculating tools. In 1623
Wilhelm Schickard designed a calculating machine, but abandoned the project, when the prototype he had started building was
                    destroyed by a fire in 1624. Around 1640, <span class="bold">Blaise Pascal</span>, a leading French
mathematician, constructed a mechanical adding device based on a design described by Greek mathematician Hero of Alexandria. Then
in
                    1672 Gottfried Wilhelm Leibniz invented the Stepped Reckoner which he completed in 1694.
                    <BR>
                    <BR>In 1837 <span class="bold">Charles Babbage</span> first described his Analytical Engine which is accepted
as the first design for a modern computer. The analytical engine had expandable memory, an arithmetic unit, and logic processing
                    capabilities able to interpret a programming language with loops and conditional branching. Although never
built, the design has been studied extensively and is understood to be Turing equivalent. The analytical engine would have had
                    a memory capacity of less than 1 kilobyte of memory and a clock speed of less than 10 Hertz. </p>
            </article>
        </div>
```

```
        <div id="side">
            <aside class="menu">
                <h2>Leisure</h2>
                <p>On free time, I go to </p>
                <div>
                    <a href="https://www.dubaichess.ae/"><img id="chess" src="http://www.dubaichess.ae/wp-
content/uploads/2015/05/DCC-LOGO-91x120.png">Chess club</a>
                    <a href="https://www.amazon.ae/"><img id="chess" src="https://images-na.ssl-images-
amazon.com/images/G/01/gc/designs/livepreview/amazon_dkblue_noto_email_v2016_us-main._CB468775337_.png">Shopping</a>
                </div>
            </aside>
        </div>
    </div>
    <footer id="foot">
        <table align="center">
            <tr>
                <td>
                    Author: Firstname, Last name.
                </td>
                <td>
                    Copyright: HWU, 2020.
                </td>
                <td>
                    <a href="mailto:me@here.there">Contact: me@here.there</a>
                </td>
            </tr>
        </table>
    </footer>
</body>
</html>
```

Display the page with the browser (alt-b). Notice that the content is there, but there is no specific structure. The content is displayed in sequence.

Presentation of the page with CSS

Now, we would like to add some presentation rules to the page using CSS.

Text and color formatting

We are adding text and background color formatting rules. For this, we are using CSS rules. We use a selector to target each element and set a declaration block consisting of pairs attribute:value rules.

1- Open the page.css file and add CSS rules to change font size and color and background color for different div elements (file page2.css).

```
/*------------------header formatting-------------------------------------------------------------*/
#uhead {
    background-color: #580c0c;
    width: 100%;
}
/*------------------formatting the avatar-------*/
#avatar {
    width: 50%;
    max-width: 100%;
    max-height: 100%;
    border: 5px solid black;
}
/*------------------heading for the title-------*/
#myh1 {
    font-size: 30px;
    color: white;
    text-align: center;
    padding-top: 15px;
}
/*------------------menu formatting---------------------------------------------------------------*/
.menu {
    background-color: #333;
}
/*--------heading within menu----*/
.menu h1 {
    text-align: center;
    color: #FFFFFF;
}
/*------------------content formatting-----------------------*/
/*------------------overall container-----------------------*/
#container {
    margin: 0 auto;
}
/*------------------presentation with profile and photo-----------------------*/
#presentation {
    background-color: #ff7f50;
    width: 100%;
    border: 1px solid black;
}
/*----------profile div-------*/
#profile {
    width: 80%;
}
```

```
/*----------photo image size-------*/
#photo {
    width: 10%;
}
/*-----------------article-----------------------*/
article {
    width: 100%;
    border: 2px solid black;
    background-color: #fff0f5;
    height: auto;
    /*float: left;*/
}
/*-----------------a rule for emphasising names in the article-----------------------*/
.bold {
    font-weight: bold;
}
/*-----------------sidebar-----------------------*/
#ide {
    z-index: 1;
    top: 0;
    left: 0;
    background-color: #111;
    color: white;
     margin-right: 50px;
     margin-top: 20px;
}

/*-----------------relative size of the chess icon-----------------------*/

#chess {
    width: 30%;
}

/*-----------------sidebar-----------------------*/

footer {
    left: 0;
    bottom: 0;
    width: 100%;
    background-color: rgb(54, 50, 50);
    color: white;
    text-align: center;
}

/*-----------------make footer white to avoid the standard blue color of the links-----------------------*/
```

```
footer a {
    color: white;
}
```

Open the index.html file and add the following line to the <head> block. This allows the browser to apply the css rules from the file page.css to page.html.

```
<link rel="stylesheet" type="text/css" href="page.css" media="all" />
```

Display the page and scroll down to view the page content.
Notice that when you display the page, all div and other layout elements are displayed sequentially

Layout

To organize the container elements and set a layout for the page, one should use the **display, float, overflow** properties of the div elements. In this activity, we are creating a layout like the example web page (page3.css).

1. Edit the page.css file
2. Add position, float, display and overflow rules as follows. Every time you add a rule, display your page and check the effect. The symbol (…) means that you should keep existing CSS rules and add the ones displayed underneath.

```
#uhead {
…
   display: flex;
}
```

```
.menu {
…
    overflow: hidden;
}
```

```
#container {
…
    position: relative;
}
```

```
#content {
    margin-left: 160px;
    float: left;
    overflow: hidden;
}
```

```
#presentation {
…
    overflow: hidden;
}
```

```
#profile {
…
    float: left;
    overflow: auto;
    display: block;
}
```

```
#photo {
…
    display: block;
}
```

```
article {
…
    float: left;
}
```

```
#side {
…
    width: 160px;
    overflow-x: hidden;
    position: absolute;
}
```

```
footer {
…
    position: fixed;
}
```

3- Display the page and check the layout.

4- Add the following line to the <head> block to make the page responsive (page3.html). This cause images to resize depending on the size of the screen.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Menu bar

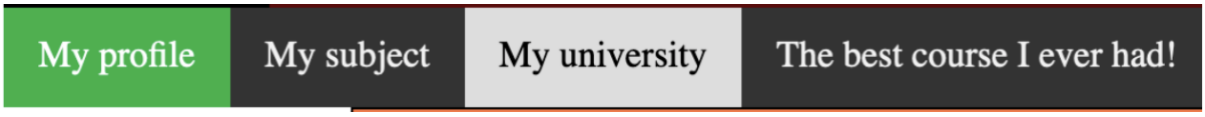
Notice that links in the menu bar and the sidebar are not clearly visible. Their blue color is not appropriate to the theme of the page.

1- To make them look like a menu, add the following CSS.
The selector ".menu a" will target the hyperlinks within the menu. The elements are given color, padding and font size.

The selector ".menu a hover" will the target hyperlink that is under the mouse cursor . It will be highlighted by changing its background color and text color.

```
/*----------------links within the menu-------*
.menu a {
    float: left;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}
/*-----links within the menu when hovered-----*/
.menu a:hover {
    background-color: #ddd;
    color: black;
}
/*---------------the active/first link-------*/
.menu a.active {
    background-color: #4CAF50;
    color: white;
}
```

Display the page and notice the change of the background color of the links when your move the cursor over them.



Adding navigation with anchors

Clicking on the links "Profile" and "Subject" has no effect. We will add links to them in order to display the appropriate part of the page (page5.html).

1. Modify the two first menu elements by adding a hyperlink to them as follows

```
        <a class="active" href="#presentation">My profile</a>
        <a href="#subject">My subject</a>
```

Note that "My profile" links now to the <section> with ID "presentation"; and "My subject" links to the <article> with ID "subject" (reduce the browser's windows size to see the effect).

Behaviour of the page with basic javascript

As we have experienced, HTML defines the content of a page; CSS defines the presentation of the page. JavaScript is a language that allows developers to add behaviour to web pages. For this lab session, we are using very simple single instruction JavaScript scripts to add some behaviour to our page.

Going back after activating a hyperlink

The first application of JavaScript is going to allow us to implement a back functionality in the page. When you click on links “My profile” or “My subject”, the browser shows the associated part of the page; but you cannot see the top of the page anymore (reduce browser size to experience this behaviour). We are adding a JavaScript to allow the user to go back to the top of the page (page6.html).

1- Modify the tag to become

```
<section id="presentation" onclick="window.location.hash='uhead';">
```

Note that we have defined the onclick property with a single instruction script. The variable window.location.hash represent the current anchor of the page displayed by the browser. Changing it will make the browser replace the page at the specified anchor. Here, we are assigning the anchor (id of the header element) to this variable.
Whenever and wherever the user clicks within the div “presentation”, the browser will display the top of the page.

1- Modify also the element <article id=”subject”> tag to give it the same behaviour

```
<article id="subject" onclick="window.location.hash='uhead';">
```

Loading an external URL in a div

The hyperlinks to external web pages display in a new tab.
We would like to display an external web page into a element of our page, without changing the rest of the page. For this, we will write a small javascript code (only one instruction). It creates a GET request similar to the one created by clicking on the hyperlink. The returned HTML page is assigned to the (page7.html).

1- Delete the following line

```
<a href="https://f28wp.github.io"> The best course I ever had!</a>
```

2- Replace it with the line

```
<span onclick=" $('#content').load('https\://f28wp.github.io');"> The best course I ever had!</span>
```

Here instead of using the <a> tag, we are using the tag to associate a script with the text “The best course I ever had!”. The script executes when the user clicks the text, hence the **onclick** property. The script is made of only one instruction.

$('#content') refers to the element having the ID "content". This an object of the page that has a predefined method (or function) called **load**. It is responsible for setting the content of the element.
The parameter 'https://f28wp.github.io' will be replaced by the HTML returned by "GET https://f28wp.github.io". Note that \ is there to protect the colon symbol to avoid JavaScript to process it as a special character. This HTML is provided as parameter to the load function which assign it to the "content" .

3- The function **load** comes from a code library called **jQuery**. To get it to work, we must add the following line to the <head> block to load the library. (More on jQuery will come later in this course).

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

4- Reload the page and click on "The best course I ever had!". The external page should

display in the content div of your page!

Restoring the content of a div

If you click now on the "My profile" or "My subject" links, you notice no effect. This is because the content of the div "content" is lost. To be able to restore it when needed, we will save it to a variable using JavaScript. We set this save operation to be done when the page first loads (page8.html).

1. We first define a **global variable** in a <script> tag within the <head> block. Add the following to the <head> block. The variable is named **storeHTML** and is assigned an empty string.

```
<script>
      var storedHTML = "";
</script>
```

2. In the <body> tag, add the onload property as follows.

```
<body onload="storedHTML = $('#content').html();">
```

In this single instruction, $('#content') is the having the ID "#content". This html element has the method html(), which extracts the current HTML content of the . In this instruction, we are assigning this HTML to the global variable storedHTML that we defined earlier. To view this html, you can log it to the console by adding console.log(storedHTML) as follows:

```
<body onload="storedHTML = $('#content').html(); console.log(storedHTML);">
```

Now, our small script is made of two instructions and still executes when the page loads.

3. To restore the content of the "#content" div, we create a new element in the navigation bar, like the following

```
<span> Forget the best course I ever had!</span>
```

4. Now, we associate a script to this so that when the user clicks, it will restore the original HTML content of the div. This script is also made of a single instruction that puts the content of the variable storedHTML into the div

```
<span onclick="$('#content').html(storedHTML);"> Forget the best course I ever had!</span>
```

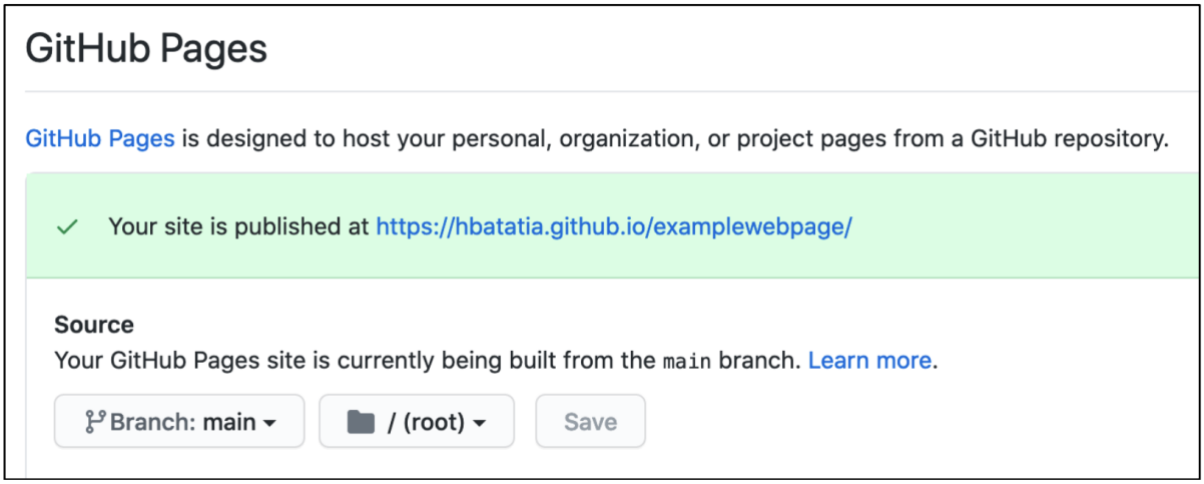
5. Display the page and try the new active span (link) after loading the course page.

2.2 Creating GitHub page

This activity allows you to expand your experience with GitHub. GitHub allows you to host a repository as a webpage (e.g., f28wp.github.io).

1. Connect to GitHub.com

In the **settings** of your repository, scroll down to **GitHub Pages**. Select **Branch: main** and click **save**.



2. Upload the web site your created in the first activity to GitHub

Remark: for more details on GitHub web pages, work through the exercise tutorial on https://pages.github.com/

3- Put a link to your page in the Readme file in your Repository