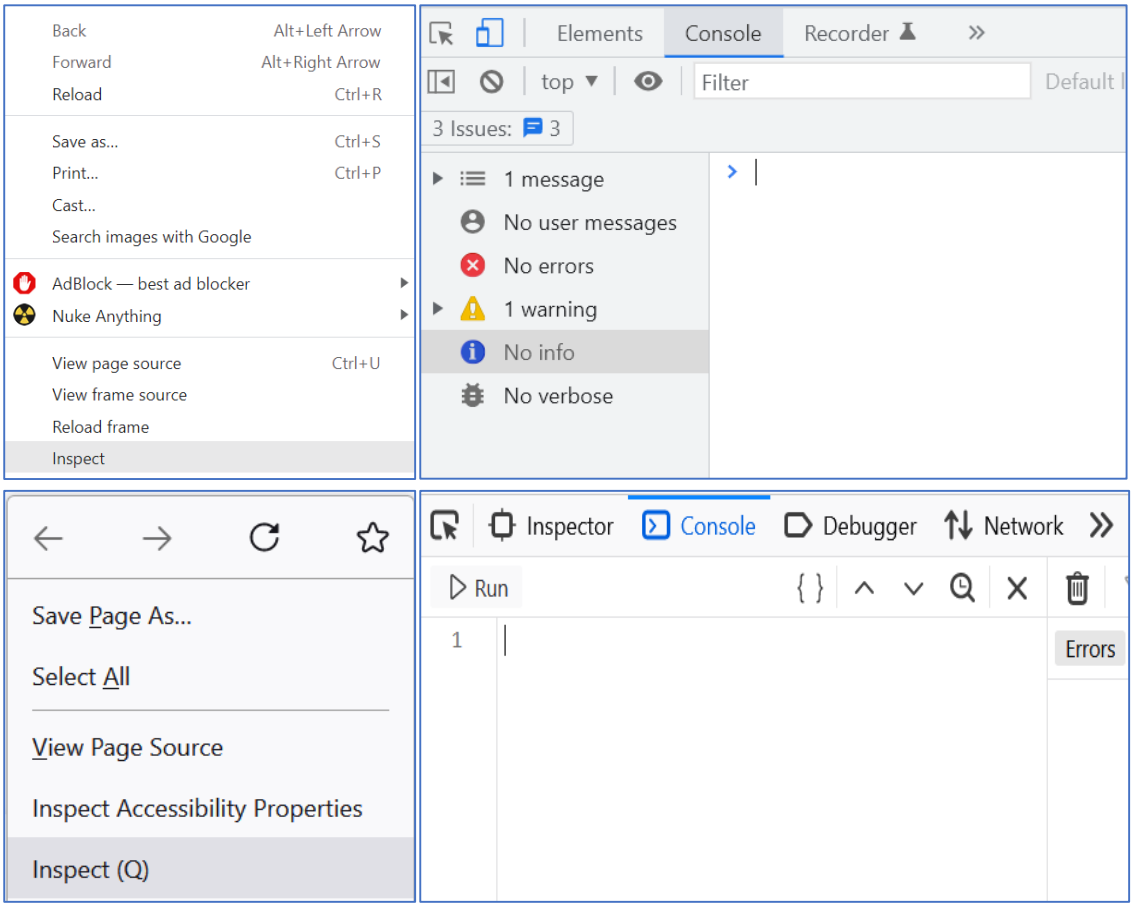
Exercise: Programming Languages

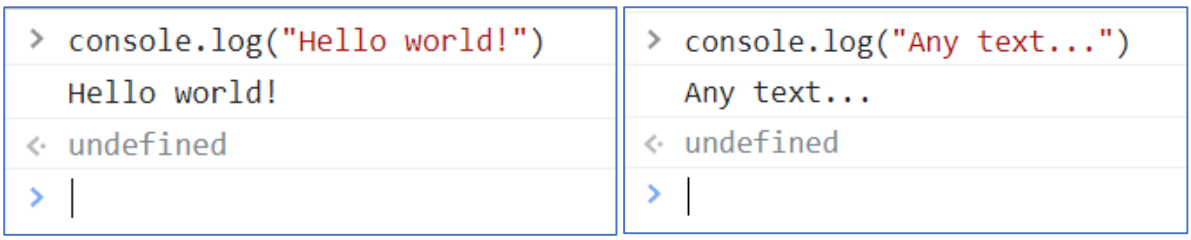
Problems for exercises and homework for the "Software Technologies" course @ Software University.

1. Run Simple JavaScript Commands from Browser Console:

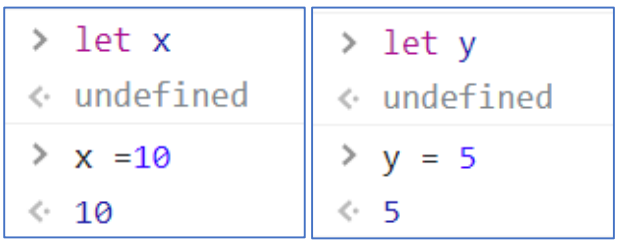
1. **Output a message to the Console:**
   - Open Browser and press "**F12**", or right-click the mouse and press "**Inspect**":



   - Use the "**console.log**" command:



2. **Create a variable and assign a value to it:**
   - Use the "**let**" command:



3. **Output the value of a variable to the console:**

```
> console.log(x)
  10
<· undefined
>
```

4. **Perform a mathematical operation:**

```
> console.log(x+y)
  15
<· undefined
>
```

```
> let z = 3
<· undefined
> console.log((x + y) * z)
  45
<· undefined
>
```

5. **Create a function and call it:**

```
> function greet(name) {console.log("Hello, " + name
  + "!");}
<· undefined
> greet("John")
  Hello, John!                                  VM1111:1
<· undefined
>
```

6. **Use a conditional statement:**

```
> let age = 25;
<· undefined
> if (age >= 18) {console.log("You are an adult.")}
  else {console.log("You are not an adult yet.")}
  You are an adult.                             VM1422:1
<· undefined
>
```

```
> age = 18
<· 18
> if (age >= 18) {console.log("You are an adult.")}
  else {console.log("You are not an adult yet.")}
  You are an adult.                             VM1447:1
<· undefined
>
```

```
> age = 17
<· 17
> if (age >= 18) {console.log("You are an
  adult.")} else {console.log("You are not an
  adult yet.")}
  You are not an adult yet.                     VM1479:1
<· undefined
>
```

7. **Use JavaScript functions for calculations:**

- Create a function that calculates a triangle area by given **height** and adjoining **side**:

```
> function calculateTriangleAreaByHeightAndSide(height, side) {
    const area = (height * side) / 2;
    return area;
  }
< undefined
```

- Now assign values for height and side of the triangle:

```
> const triangleHeight = 6;
  const triangleSide = 4;
< undefined
>
```

- Call the **`calculateTriangleAreaByHeightAndSide`** function with the assigned values:

```
> const area =
  calculateTriangleAreaByHeightAndSide(triangleHeight,
  triangleSide);
  console.log("The area of the triangle is: " + area);
  The area of the triangle is: 12                    VM86:2
< undefined
```

- Create a similar function for finding the area using the **Heron's formula** for calculating the triangle area, by given three sides of the triangle:

```
> function calculateTriangleArea(sideA, sideB, sideC) {
    const semiPerimeter = (sideA + sideB + sideC) / 2;
    const area = Math.sqrt(
      semiPerimeter *
        (semiPerimeter - sideA) *
        (semiPerimeter - sideB) *
        (semiPerimeter - sideC)
    );
    return area;
  }
< undefined
>
```

- Assign values for the sides:

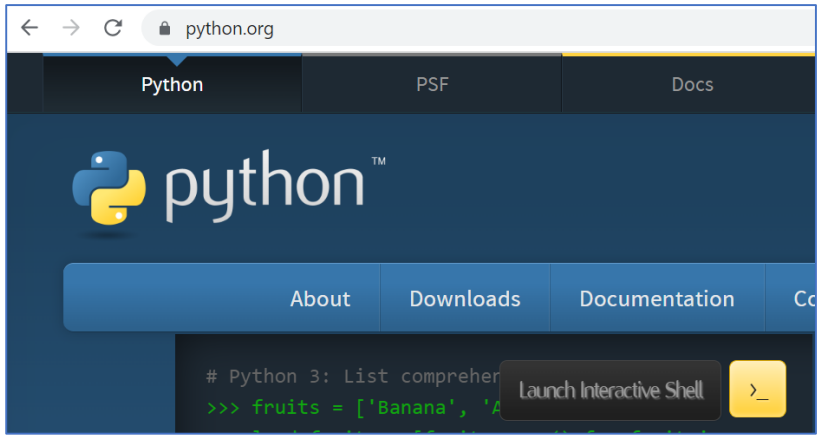
```
> const side1 = 3;
  const side2 = 4;
  const side3 = 5;
```

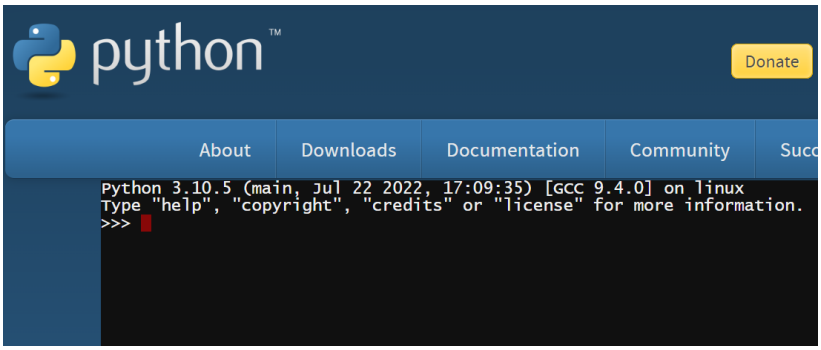
- Call the **`calculateTriangleArea`** function with the assigned values:

```
  const area = calculateTriangleArea(side1, side2, side3);
  console.log("The area of the triangle is: " + area);
  The area of the triangle is: 6                     VM47:6
< undefined
```

2. Play with Python Online:

1. **Launch an Online Interactive Shell:**
   - Open python.org and click on the yellow button ">_" to launch interactive shell.





2. **Arithmetic Operators:**
   - Python supports arithmetic operators like addition (+), subtraction (-), multiplication (*), division (/), modulo or remainder (%), and exponentiation (**).

```
>>> # Addition
>>> print(2 + 3)
5
>>> # Subtraction
>>> print(7 - 4)
3
>>> # Multiplication
>>> print(5 * 6)
30
>>> # Division
>>> print(8 / 2)  # Output: 4.0 (division always returns a float)
4.0
```

```
>>> #Modulo
>>> print(9 % 2) # Output: 1 (remainder when 9 is divided by 2)
1
>>> print(15 % 4)
3
>>> print(17 % 12)
5
>>>
```

```
>>> # Exponentiation
>>> print(2 ** 3) # Output: 8 (2 raised to the power of 3)
8
>>>
```

3. **Comparison Operators:**
    - Python supports comparison operators like **equal to (==)**, **not equal to (!=)**, **greater than (>)**, **less than (<)**, **greater than or equal to (>=)**, and **less than or equal to (<=)**.

```
>>> x = int(input())
5
>>> y = int(input())
10
```

```
>>> # Equal to
>>> print(x == y) # Output: False
False
>>>
>>> # Not equal to
>>> print(x != y) # Output: True
True
```

```
>>> # Greater than
>>> print(x > y)  # Output: False
False
>>>
>>> # Less than
>>> print(x < y)  # Output: True
True
```

```
>>> # Greater than or equal to
>>> print(x >= y) # Output: False
False
>>>
>>> # Less than or equal to
>>> print(x <= y) # Output: True
```

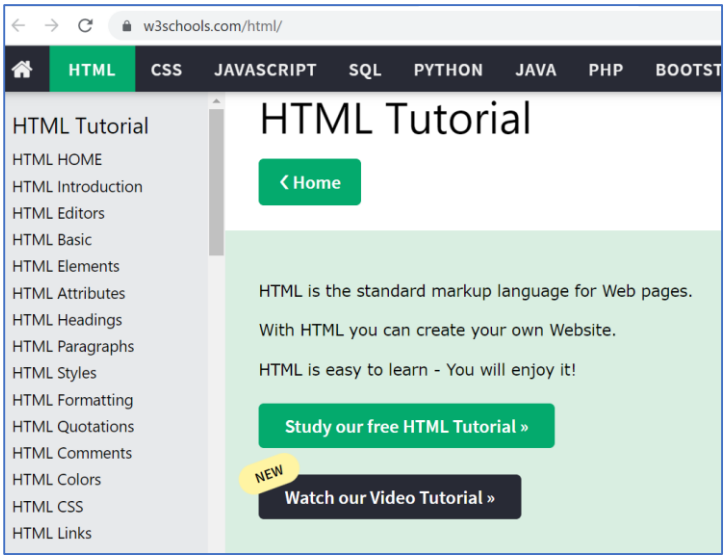
4. **String Operations:**
    - Python supports many string operations like **concatenation (+)**, **repetition (*)**, **indexing ([])**, and **slicing ([start:end:step])**.

```
>>> a = "Hello"
>>> b = "World"
>>>
```

```
>>> a = "Hello"
>>> b = "World"
>>> # Concatenation
>>> print(a + " " + b) # Output: "Hello World"
Hello World
>>> # Repetition
>>> print(a * 3)       # Output: "HelloHelloHello"
HelloHelloHello
>>> # Indexing
>>> print(a[1])        # Output: "e" (the second character of "Hello")
e
>>> # Slicing
>>> print(b[1:4])      # Output: "orl" (the second to fourth characters of "World")
orl
```

3. HTML – W3Schools Tutorials:

1. [HTML Tutorial](#) – [HTML Introduction](#):
   - Follow the link and start the HTML Tutorial:



   - Carefully read the introduction paragraph and remind yourself what is HTML:

HTML is the standard markup language for creating Web pages.

What is HTML?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

   - See the example given and try some basic manipulations by yourself:

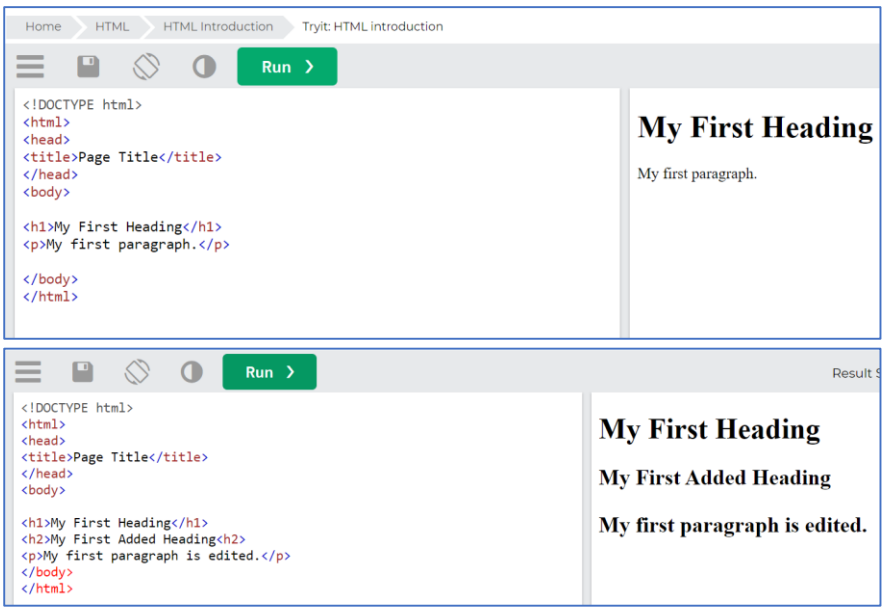
Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

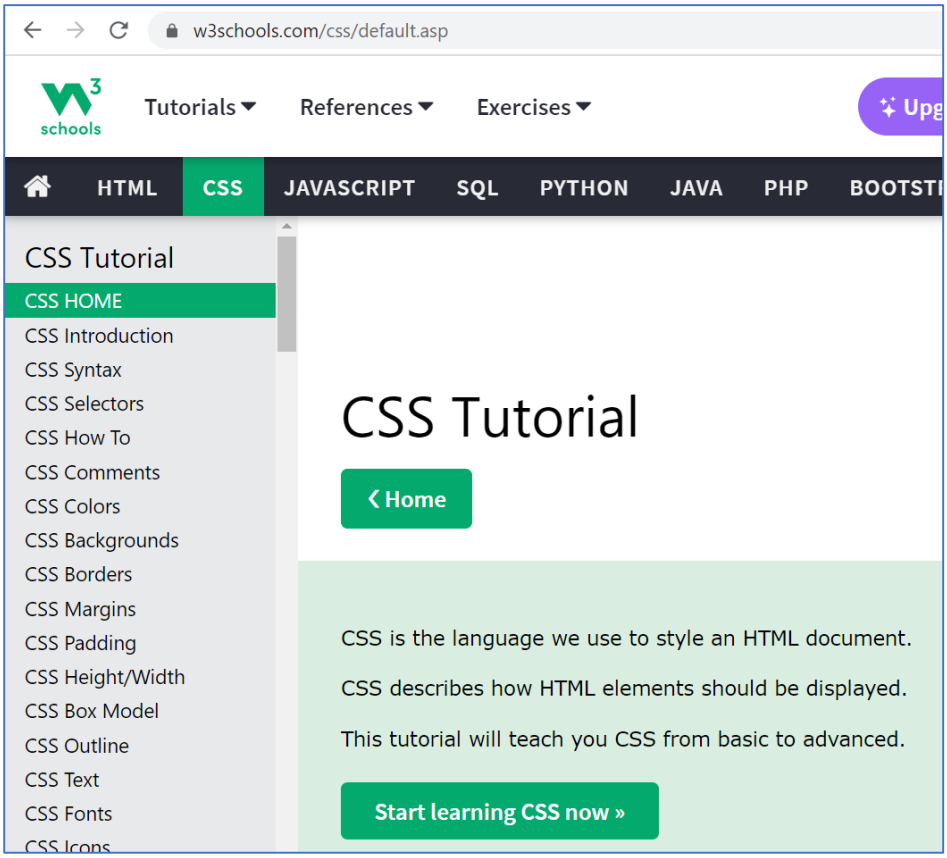
Try it Yourself »



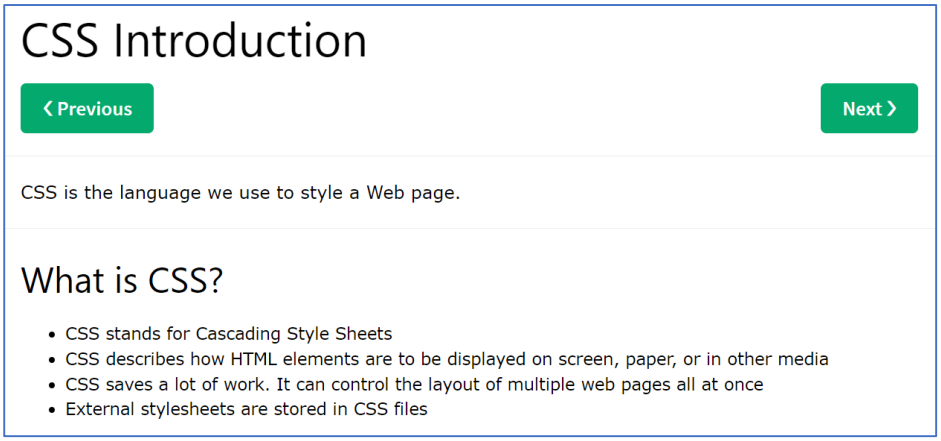
2. **HTML Tutorial – [HTML Editors]:**
    - Learn HTML Using Notepad or TextEdit – Follow the steps shown to create your first web page with Notepad or TextEdit.
3. **HTML Tutorial – [HTML Basic]:**
    - On the third page of the Tutorial, there will be shown some basic HTML examples.
    - You will be able to understand the structure of a simple HTML page.
    - In the end you will know how to view the HTML source code and inspect HTML elements.

4. CSS – W3Schools Tutorials:

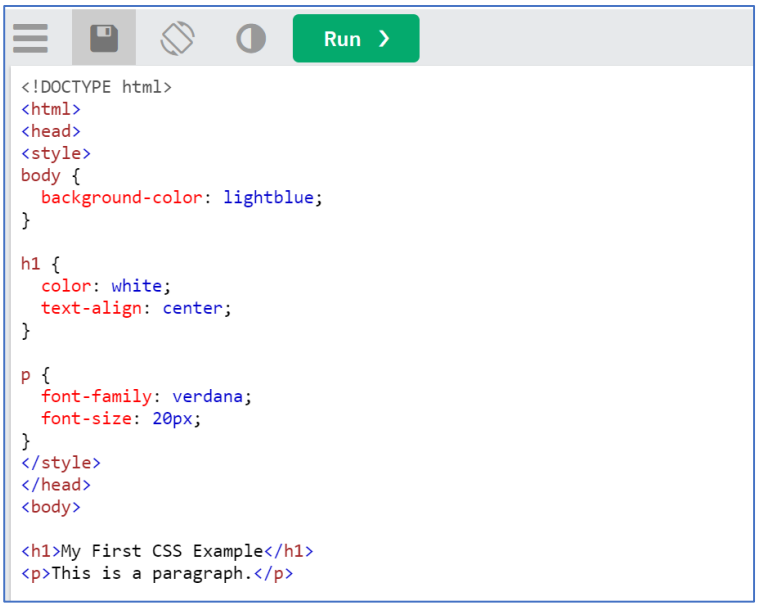
1. **[CSS Tutorial] – [CSS Introduction]:**
    - Follow the [link] and start the CSS Tutorial:

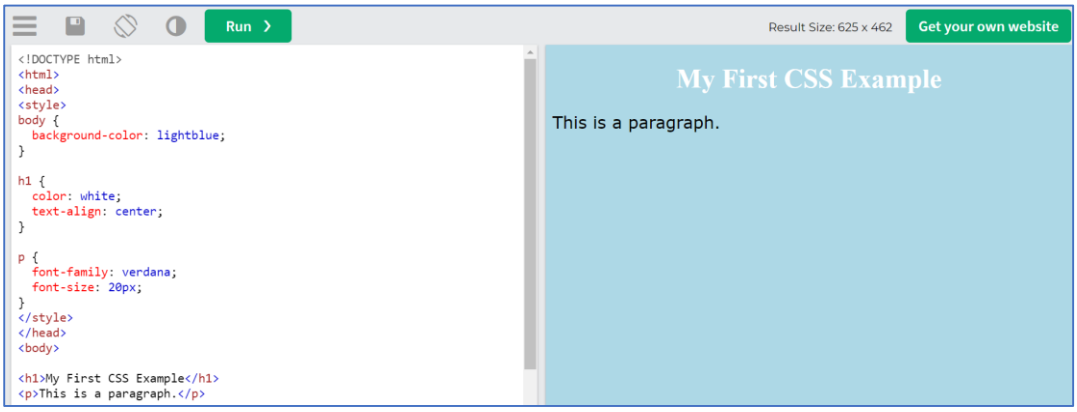


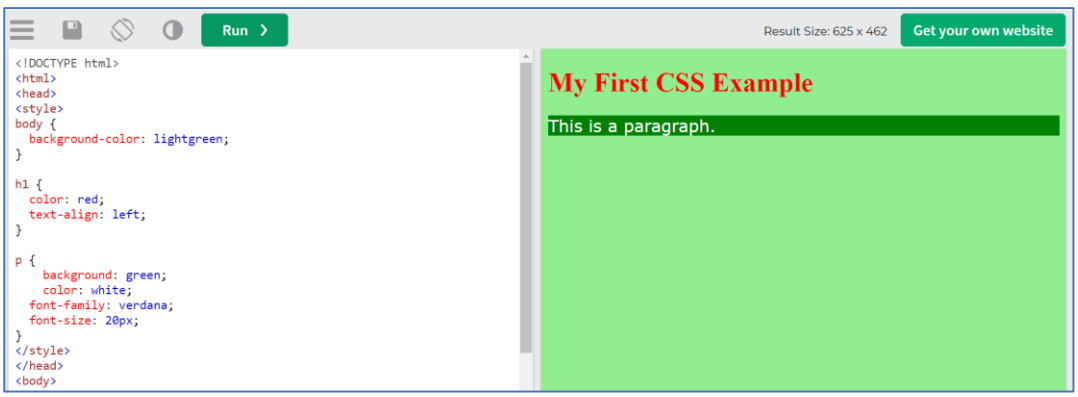
- Carefully read the introduction paragraph and remind yourself what is CSS:



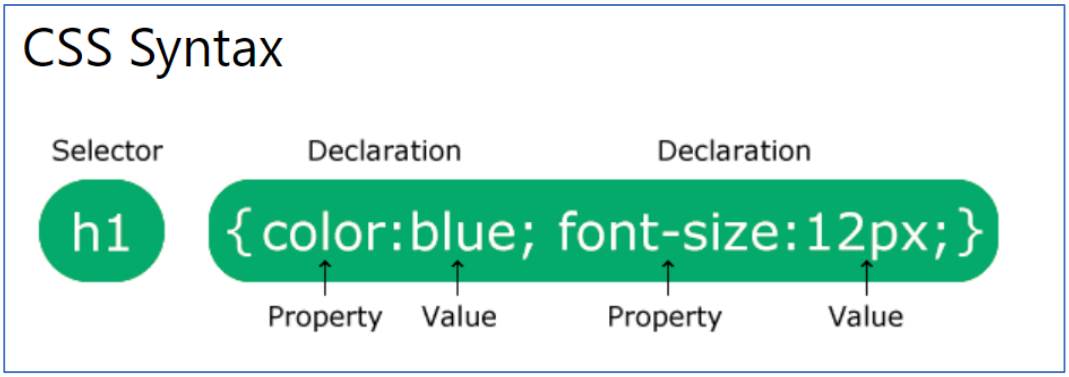
- See the example given and try some basic manipulations by yourself:







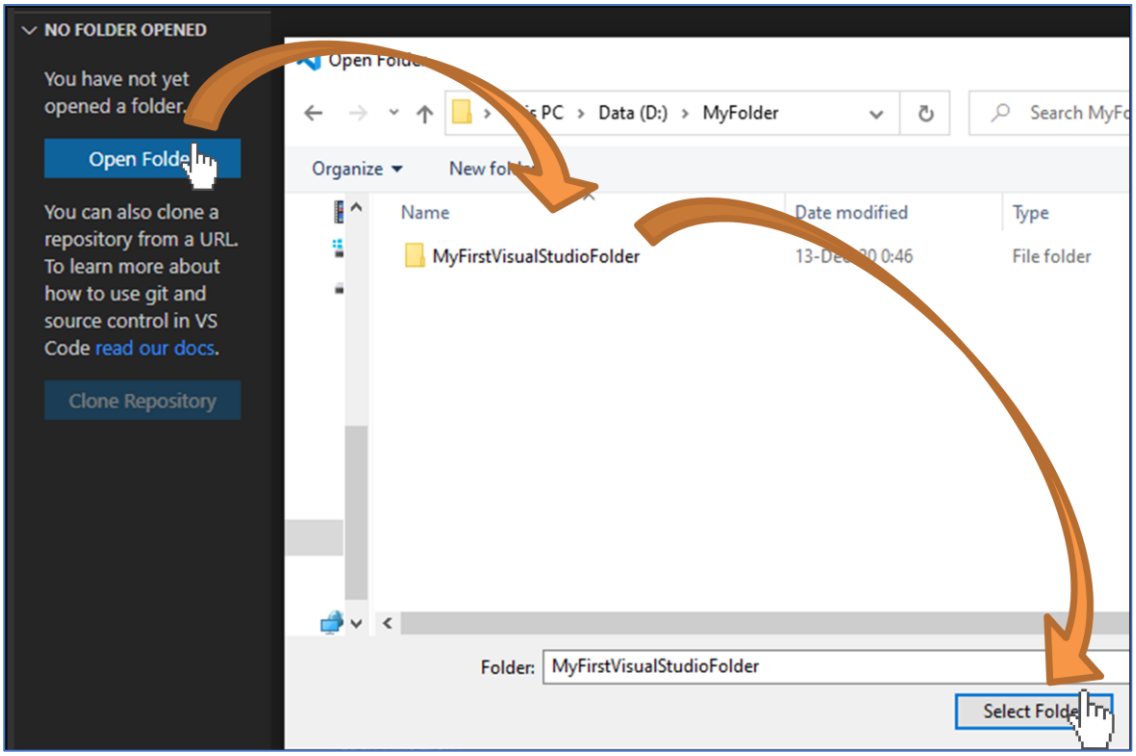
2. **CSS Tutorial – CSS Syntax:**
    - Learn the structure of the CSS Syntax:



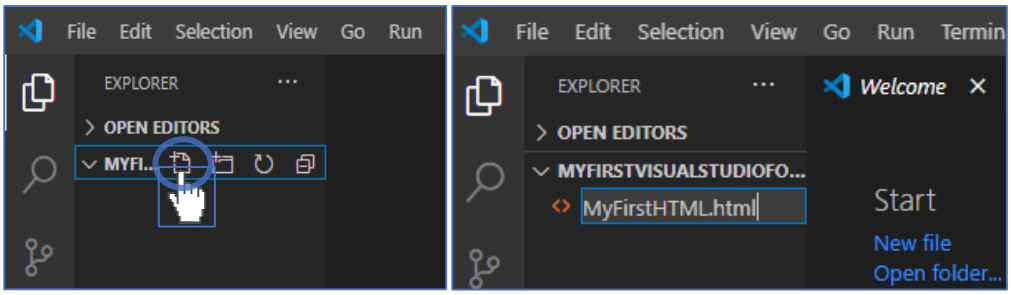
3. **CSS Tutorial – CSS Selectors:**
    - CSS selectors are used to "find" (or select) the HTML elements you want to style.
    - This page will explain the most basic CSS selectors.

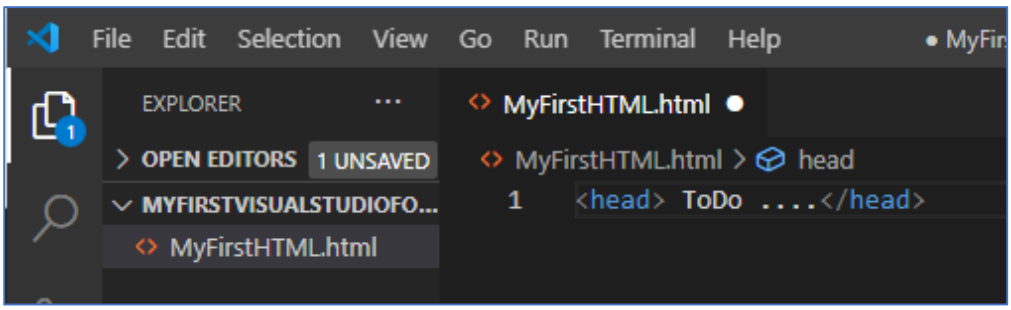
5. Create a Simple Webpage using HTML & CSS:

1. **Install VS Code by using the guide in the resources.**
2. **Start with Visual Studio Code.**
    - **Create and select the working folder in which we will create our first HTML page:**

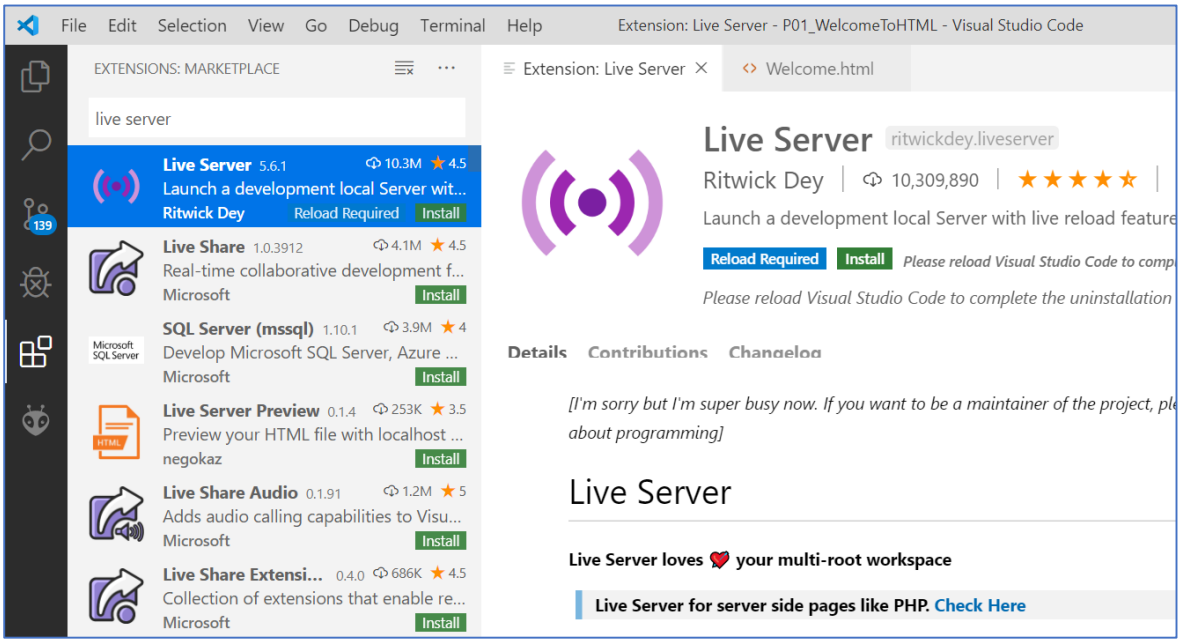


    - Create e new **HTML file**:

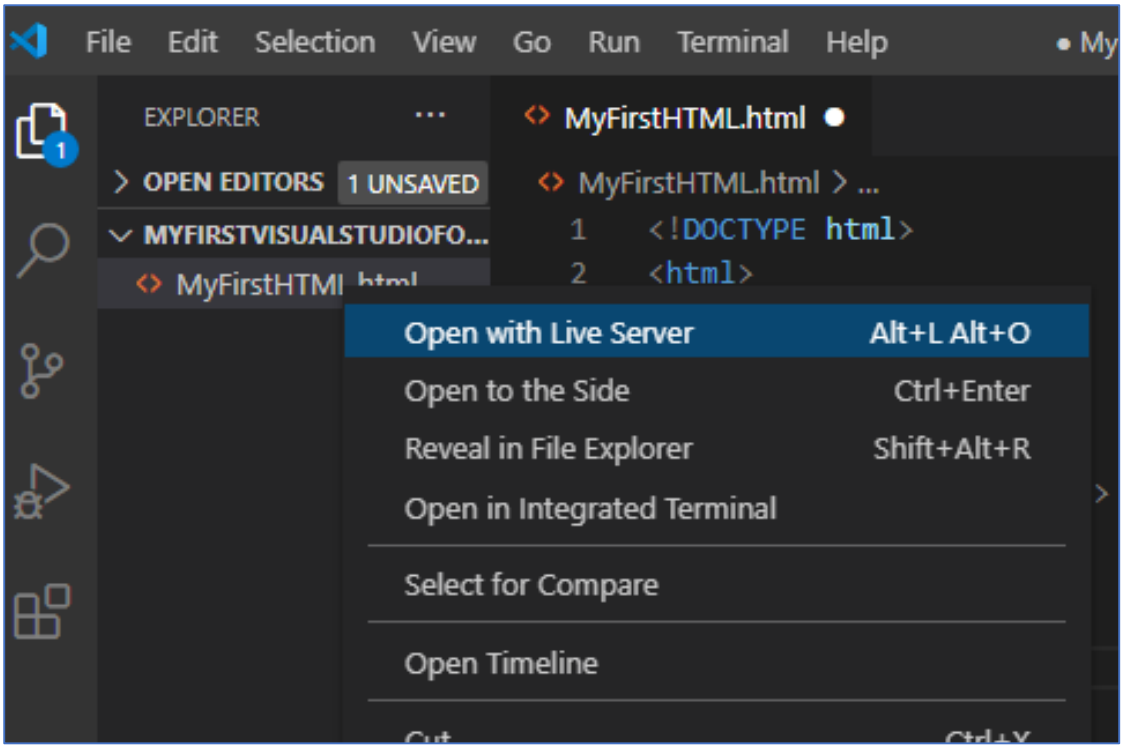




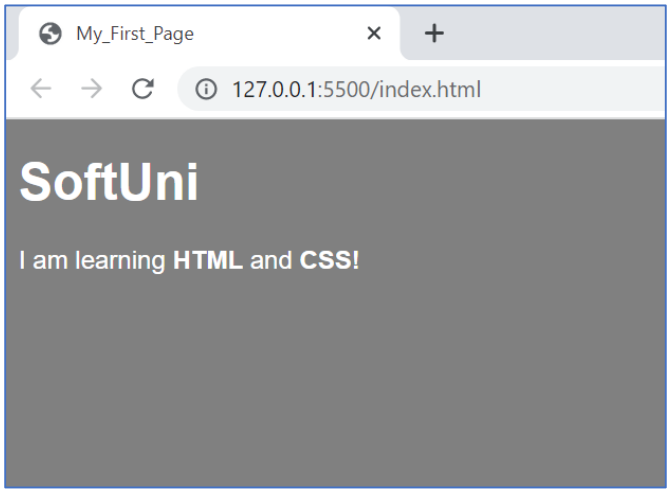
- Install the **extension "`Live Server`"**:



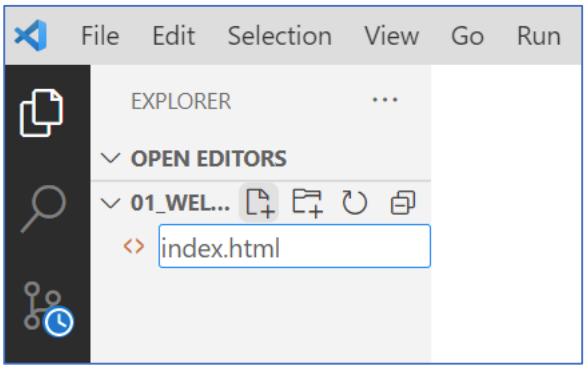
- Seeing the result of the written code in a browser, **right-click** on the project and select [**Open with Live Server**].



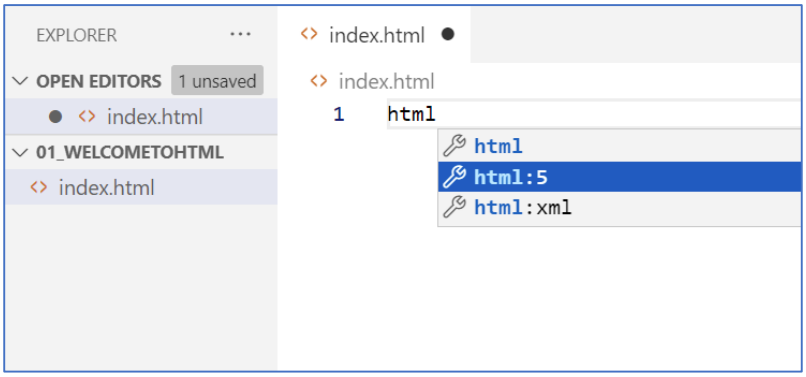
3. Create a **Web Page** like the following:



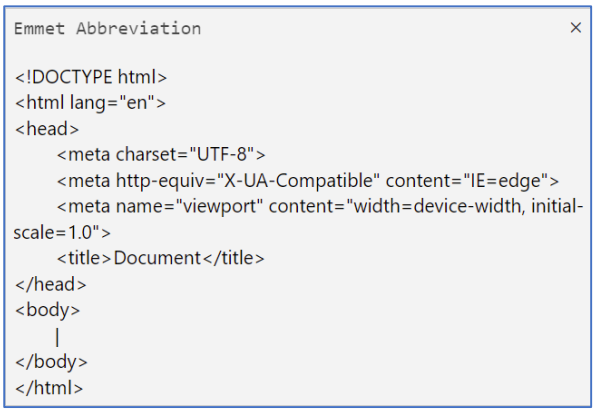
- Create **`index.html`** file:



- Start writing "**`html`**" and select the "**`html : 5`**" option:



- For example, you can type **`html:5`** and then press the "Tab" key, and emmet will expand it to the HTML5 doctype declaration and basic structure:



```
index.html
index.html > html > head > meta
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Document</title>
8   </head>
9   <body>
10      
11  </body>
12  </html>
```

- Change the title of the **Document** to **My_First_Page**:

```
index.html > html
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>My_First_Page</title>
8   </head>
9   <body>
10  </body>
11  </html>
```

- In the body part, create a **h1** tag, with text "SoftUni" :

```
9   <body>
10      h1
11  </body
12  </html>
```

h1

Emmet Abbreviation

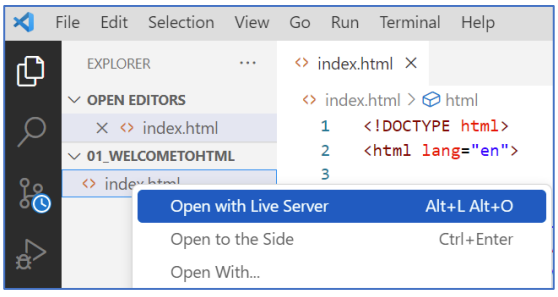
<h1>|</h1>

```
11  <body>
12      <h1>SoftUni</h1>
13  </body>
```

- Add a **p** tag with text I am learning HTML & CSS in the body part:

```
11  <body>
12      <h1>SoftUni</h1>
13      <p>I am learning HTML and CSS!</p>
14  </body>
```

- Open the **index.html** file with Live Server:



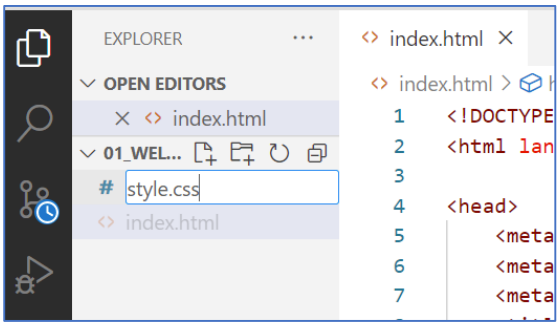
- Your page should look like this:

- Use strong tag for bolding selected parts of the text:

```
11    <body>
12        <h1>SoftUni</h1>
13        <p>I am learning <strong>HTML</strong> and <strong>CSS!</strong></p>
14    </body>
```

- Your page should look like this:

- It is time to create a **CSS file** and use it to control the visual appearance of web page:



- In order the **CSS file** to work properly, **`index.html`** and **`style.css`** files should know about each other. This should be made in the **head part** of the **`index.html`** file, creating **link to `style.css`**:

```html
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>My_First_Page</title>
</head>
```

- Start working on the **body part** of the document:

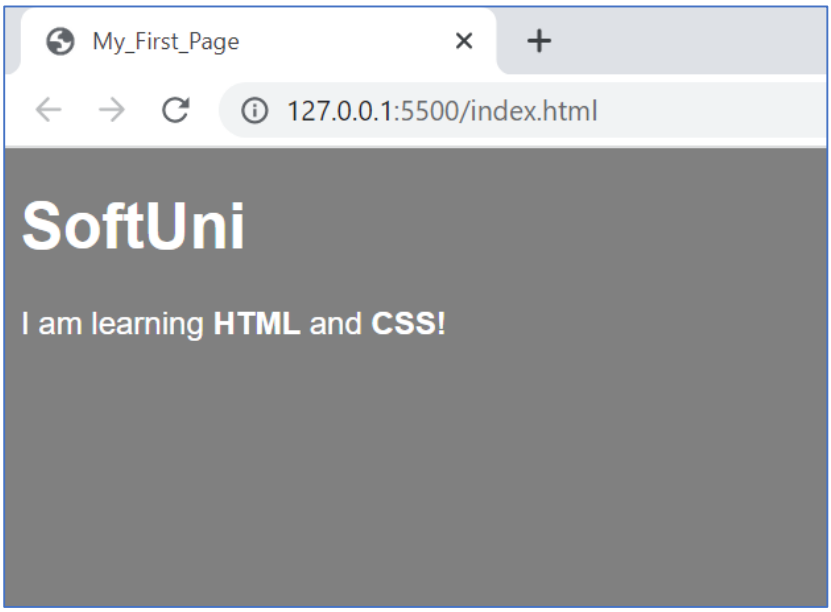
```
index.html   style.css 1
style.css > body
1  body{
2
3  }
```

```
index.html   style.css
style.css > body
1  body{
2      background-color: grey;
3  }
```

```
index.html   style.css
style.css > body
1  body {
2      background-color: grey;
3      font-family: Arial, Helvetica, sans-serif;
4  }
```

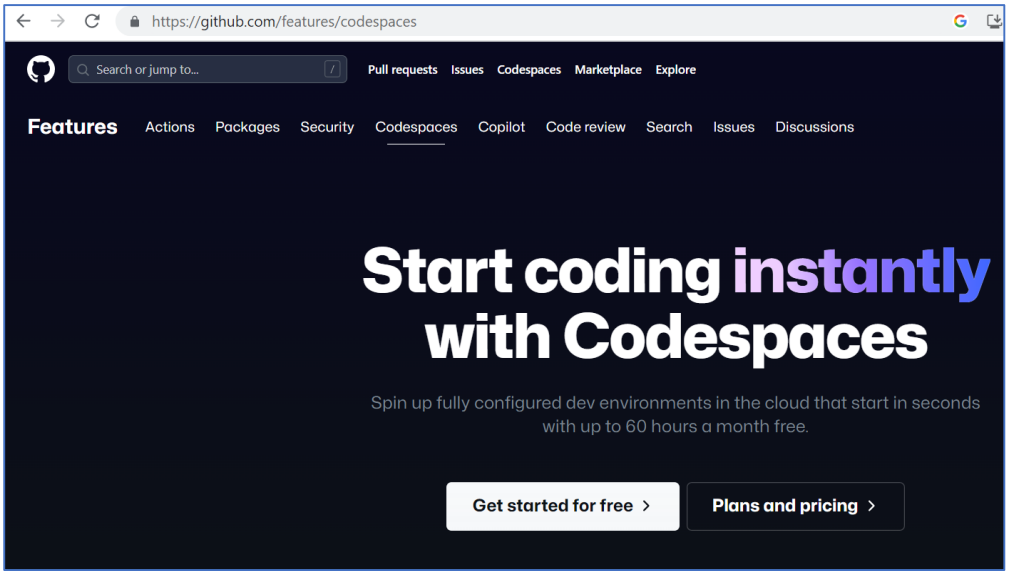
```
index.html   style.css
style.css > body
1  body {
2      background-color: grey;
3      font-family: Arial, Helvetica, sans-serif;
4      color: white;
5  }
```

- Your page should look like this:

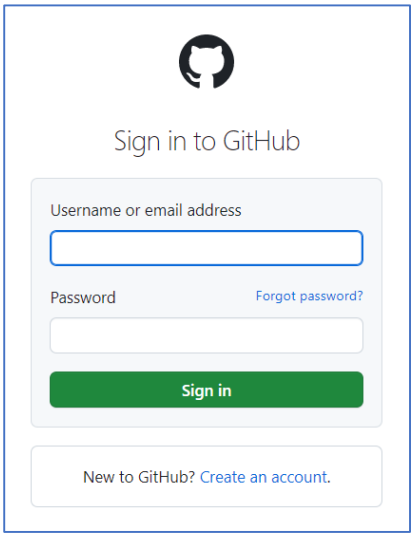


6. Run a Software Project in GitHub Codespaces:

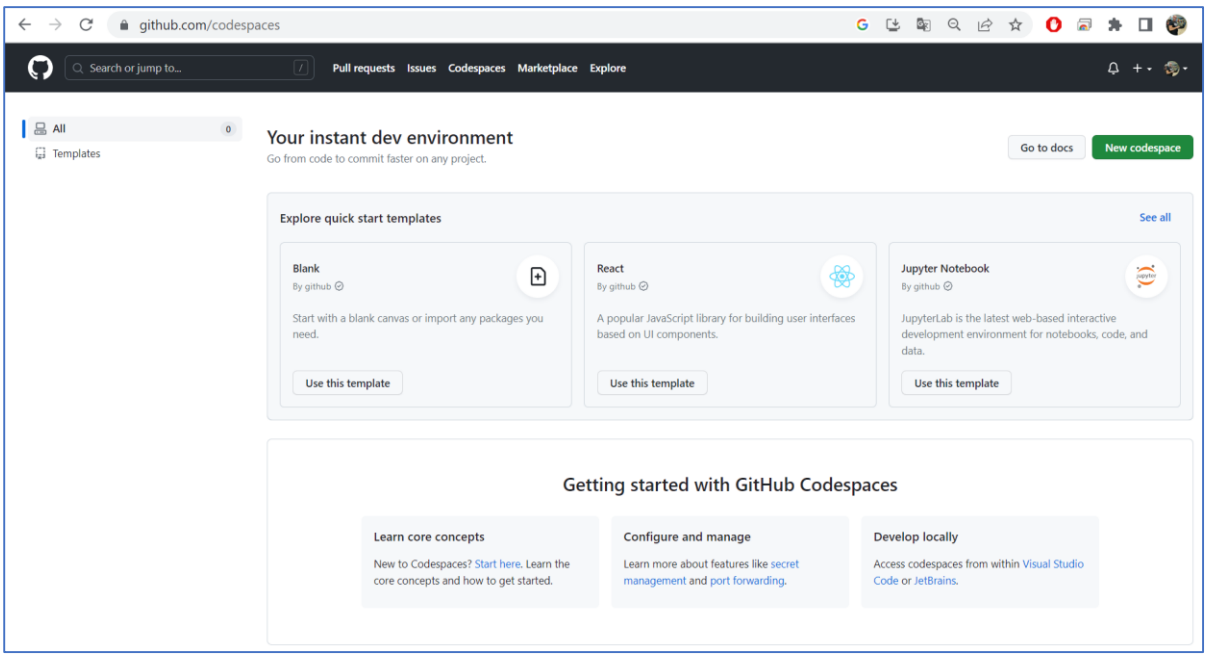
1. **Codespaces**:
   - Open the following link to open the GitHub Codespaces:



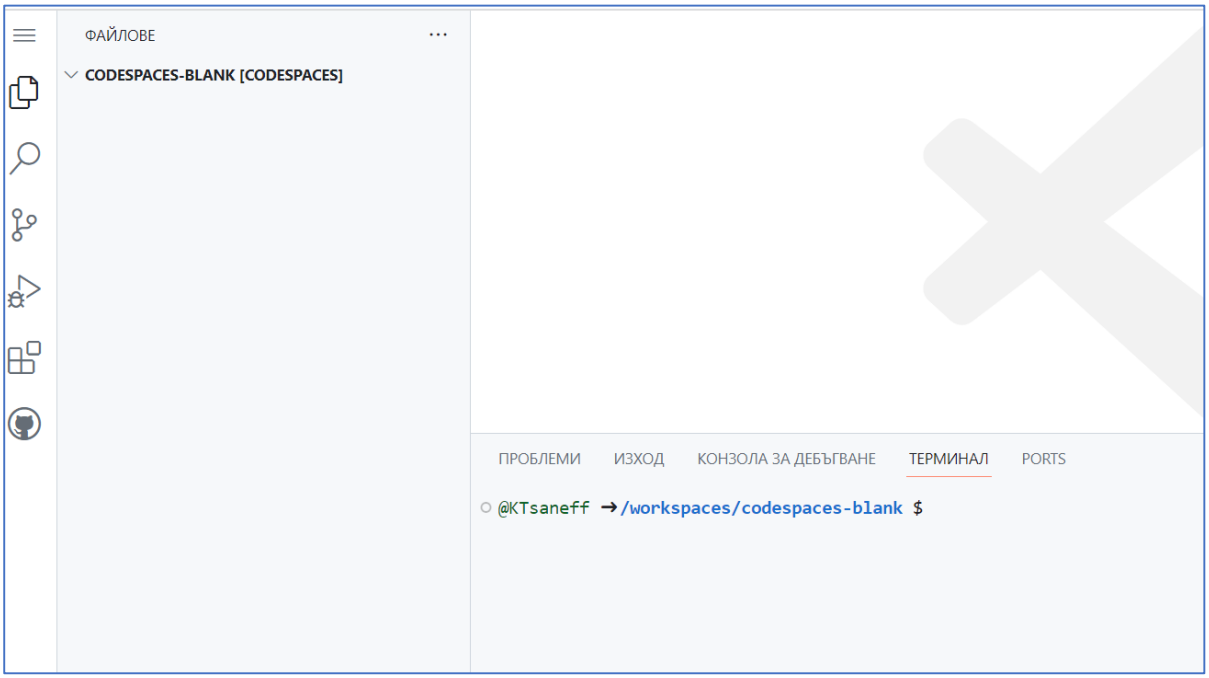
   - Click on the "**Get started for free**" button:



   - Sign in to GitHub or create a new account. You will need that account through your training.
   - Once you are signed in, the following page will appear. There you will be able to clone a real software project and explore it:



- Each codespace you create is hosted by GitHub in a Docker container, running on a virtual machine. You can choose from a selection of virtual machine types, from 2 cores, 8 GB RAM, and 32 GB storage, up to 32 cores, 64 GB RAM, and 128 GB storage.
- Use Blank template:



- Now the environment is ready for work and you can clone the repository of the project, using "**git clone**" command:

```
@KTsaneff →/workspaces/codespaces-blank $ git clone https://github.com/nakov/MVC-app-integration-tests-example-mocha
Cloning into 'MVC-app-integration-tests-example-mocha'...
remote: Enumerating objects: 235, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 235 (delta 1), reused 0 (delta 0), pack-reused 228
Receiving objects: 100% (235/235), 57.71 KiB | 3.61 MiB/s, done.
Resolving deltas: 100% (101/101), done.
@KTsaneff →/workspaces/codespaces-blank $
```

2. **Project is cloned:**
   - Edit and modify the code:
     - You can open the code files in the code editor provided by Github code spaces and make changes to the code. You can write new code, modify existing code, or delete code files.

- Run and test the code:
  - You can use the integrated terminal in Github code spaces to run and test your code. You can execute commands to build, run, and test your project.
- Manage project dependencies:
  - You can manage the dependencies of your project by installing new dependencies, updating existing dependencies, or removing dependencies. You can also manage package versions and package configurations.
- Use Git to manage code changes:
  - You can use Git commands in the terminal to manage code changes. You can commit your changes, create new branches, merge branches, and push changes to the remote repository.
- Collaborate with other developers:
  - If you are working with other developers, you can invite them to the code space to collaborate on the project. You can work together in real-time, share code, and communicate through the integrated chat feature.
- **Overall, with a cloned project in Github code spaces, you have all the tools you need to develop and manage your project. You can write code, test it, manage dependencies, and collaborate with others, all in one convenient environment.**