TEMPERATURE CONVERTER APP DEVELOPMENT IN ANDRIOD STUDIO

SUBMITTED BY:

RAJU YERIPILLI-20GK1A0424

DEPARTMENT OF ELECTRONICS AND

COMMUNICATION ENGINEERING

PRIYADARSHINI INSTITUTE OF TECHNOLOGY & MANAGEMENT

5thMile,Pulladigunta,Vatticherukuru,Guntur-522017

**TEMPERATURE CONVERTER APP DEVELOPMENT:**

Building coding projects is one of the best ways to learn coding and build your portfolio. However, sometimes a simple project spec may not be enough to help you build your project. This tutorial will walk you through how to build a Temperature converter website, one of the most popular projects on DevProjects.

You’ll build a simple tool to convert between different temperature units. To follow along, check out the project spec on DevProjects!

Overview of this tutorial:

Introduction

Project Overview

Setting up the project environment

Let’s Start Building

Result

Time to deploy

Recap

Introduction

While Kelvin is the SI Unit of temperature, people generally prefer Fahrenheit or Celsius unit type to measure temperature. We’re going to build a temperature converter that will convert Fahrenheit, Celsius, and Kelvin units to each other, using the most popular CSS framework called Bootstrap 4.6 and JavaScript library – jQuery.

Here is the live demo of Temperature converter website.

Project Overview

In this tutorial, we will create a temperature converter and walk through how the website works. We will deploy this project on GitHub using GitHub pages, a hosting service by GitHub that allows you to publish static websites online directly from repositories. The tutorial will guide you through the step-by-step process of setting up the GitHub repositories before publishing the website. The suggested text editor is VScode, but you can choose any other text editor you like.

Why Bootstrap

We are using this library because of its variety of components. Additionally, the bootstrap grid system is based on Flex, which provides us with full responsive support for any website. You can read more about it on its official website.

Why jQuery

While other powerful libraries and frameworks have emerged, jQuery is very beginner friendly and is the best library for practicing and getting started in JavaScript libraries.

jQuery is a small, fast, and feature-rich library. It saves developers a lot of time by completing complicated tasks with just a few lines of code.

What you need

VSCode or any other text editor

Any browser of your choice

Basics knowledge of jQuery and Bootstrap

A GitHub Account for deployment

Recreate this project on DevProjects. Try it now!

Find Mentors

Find Freelancers

How It Works

Events

BECOME A MENTOR

SIGN UP

LOG IN

Codementor Blog

Last updated Sep 22, 2021

A Step-by-step Tutorial for Building a Temperature Converter Website

You’ll build a simple tool to convert between different temperature units.

A Step-by-step Tutorial for Building a Temperature Converter Website

Building coding projects is one of the best ways to learn coding and build your portfolio. However, sometimes a simple project spec may not be enough to help you build your project. This tutorial will walk you through how to build a Temperature converter website, one of the most popular projects on DevProjects.

You’ll build a simple tool to convert between different temperature units. To follow along, check out the project spec on DevProjects!

Overview of this tutorial:

Introduction

Project Overview

Setting up the project environment

Let’s Start Building

Result

Time to deploy

Recap

Introduction

While Kelvin is the SI Unit of temperature, people generally prefer Fahrenheit or Celsius unit type to measure temperature. We’re going to build a temperature converter that will convert Fahrenheit, Celsius, and Kelvin units to each other, using the most popular CSS framework called Bootstrap 4.6 and JavaScript library – jQuery.

Here is the live demo of Temperature converter website.

Project Overview

In this tutorial, we will create a temperature converter and walk through how the website works. We will deploy this project on GitHub using GitHub pages, a hosting service by GitHub that allows you to publish static websites online directly from repositories. The tutorial will guide you through the step-by-step process of setting up the GitHub repositories before publishing the website. The suggested text editor is VScode, but you can choose any other text editor you like.

⭐ Why Bootstrap

We are using this library because of its variety of components. Additionally, the bootstrap grid system is based on Flex, which provides us with full responsive support for any website. You can read more about it on its official website.

⭐ Why jQuery

While other powerful libraries and frameworks have emerged, jQuery is very beginner friendly and is the best library for practicing and getting started in JavaScript libraries.

jQuery is a small, fast, and feature-rich library. It saves developers a lot of time by completing complicated tasks with just a few lines of code.

⭐ What you need

VSCode or any other text editor

Any browser of your choice

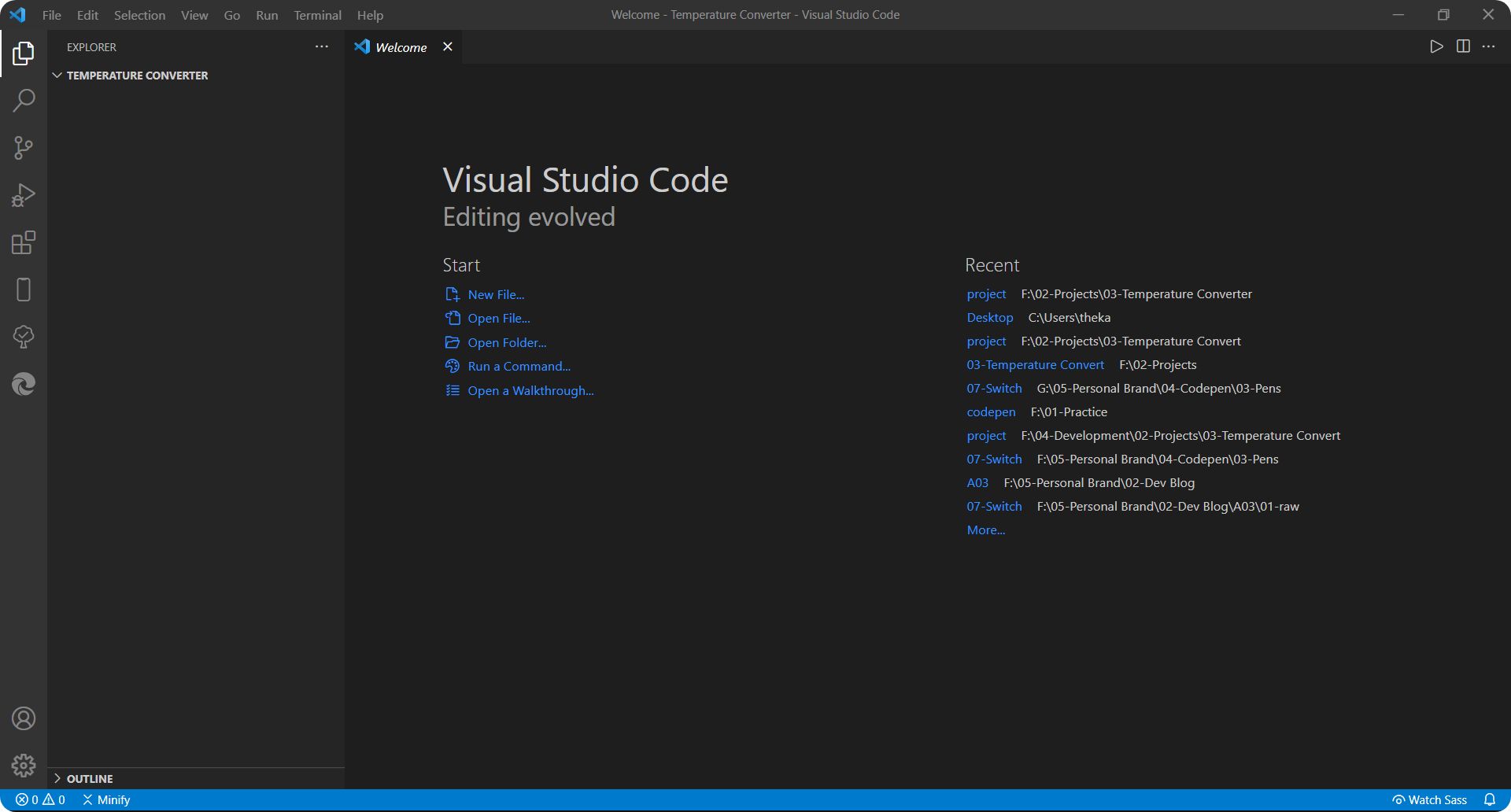
Basics knowledge of jQuery and Bootstrap

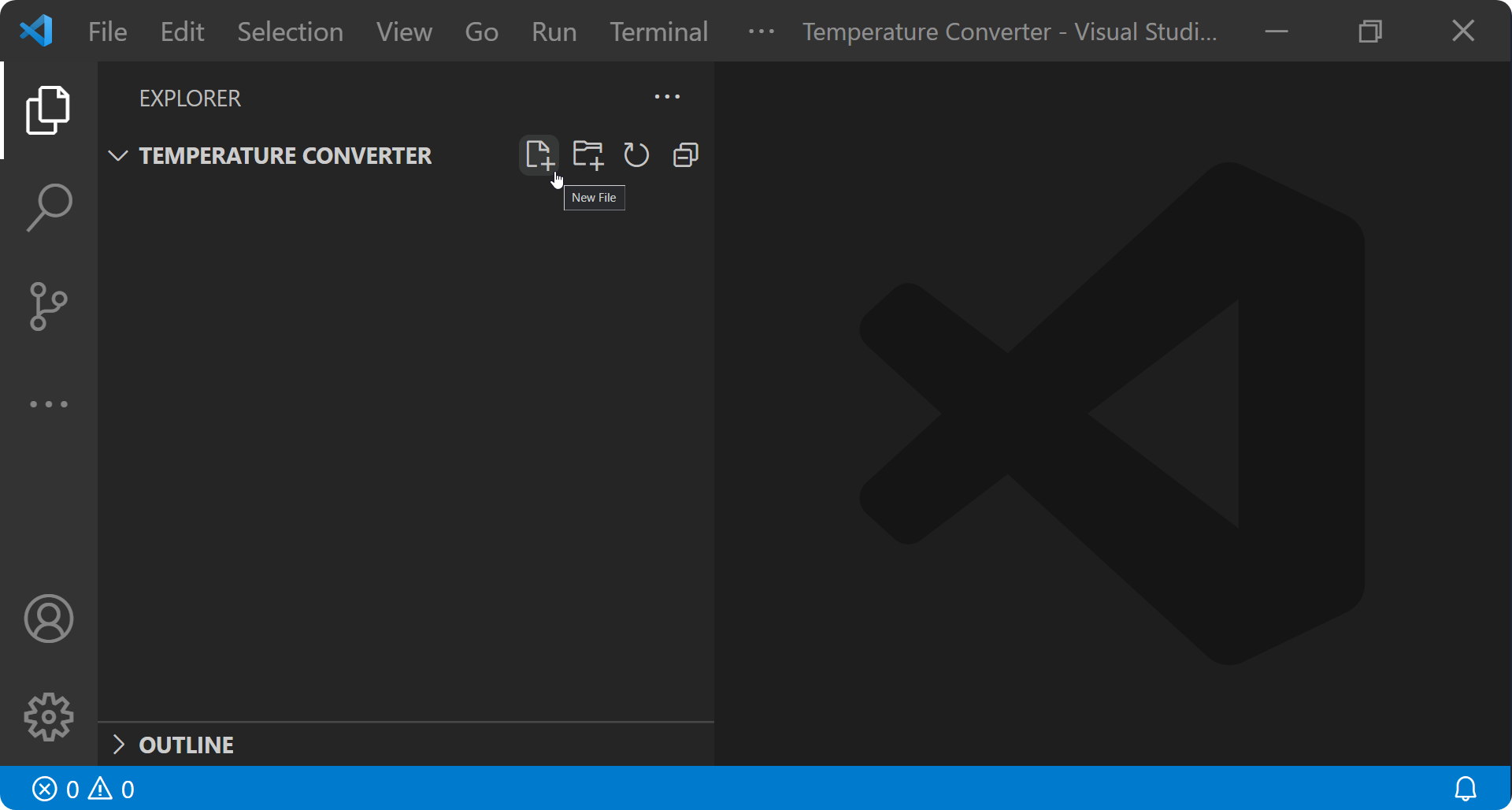
A GitHub Account for deployment

Recreate this project on DevProjects. Try it now!

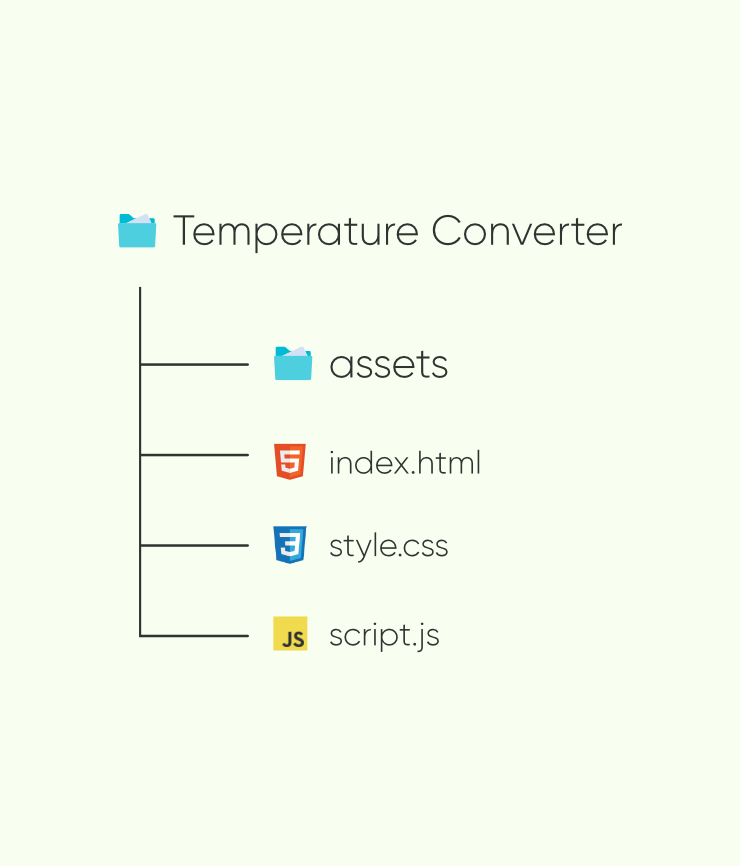
Setting up the project environment

Create a new folder named Temperature Converter and open the folder in VScode. Initially, our workspace will look like this:



Hover the mouse on the sidebar to find the icon for creating new files or folders

Create a sub-folder named assets and the following files: index.html, style.css, and script.js.



Assets

In this folder, we’ll store all icons and any other media used in our project.

Index.html

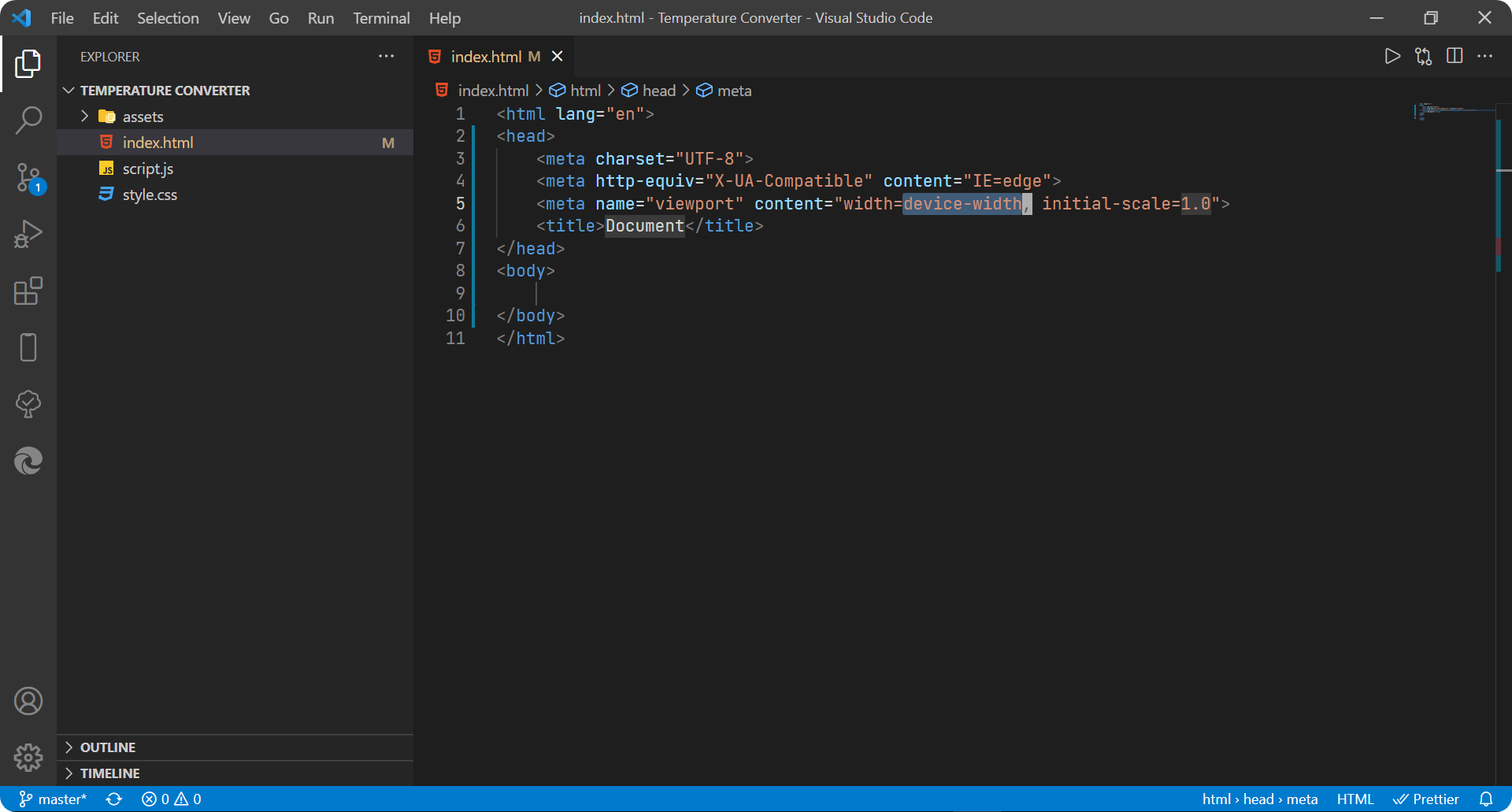
To create the structure of the website.

Style.css

To add custom CSS styling in our project.

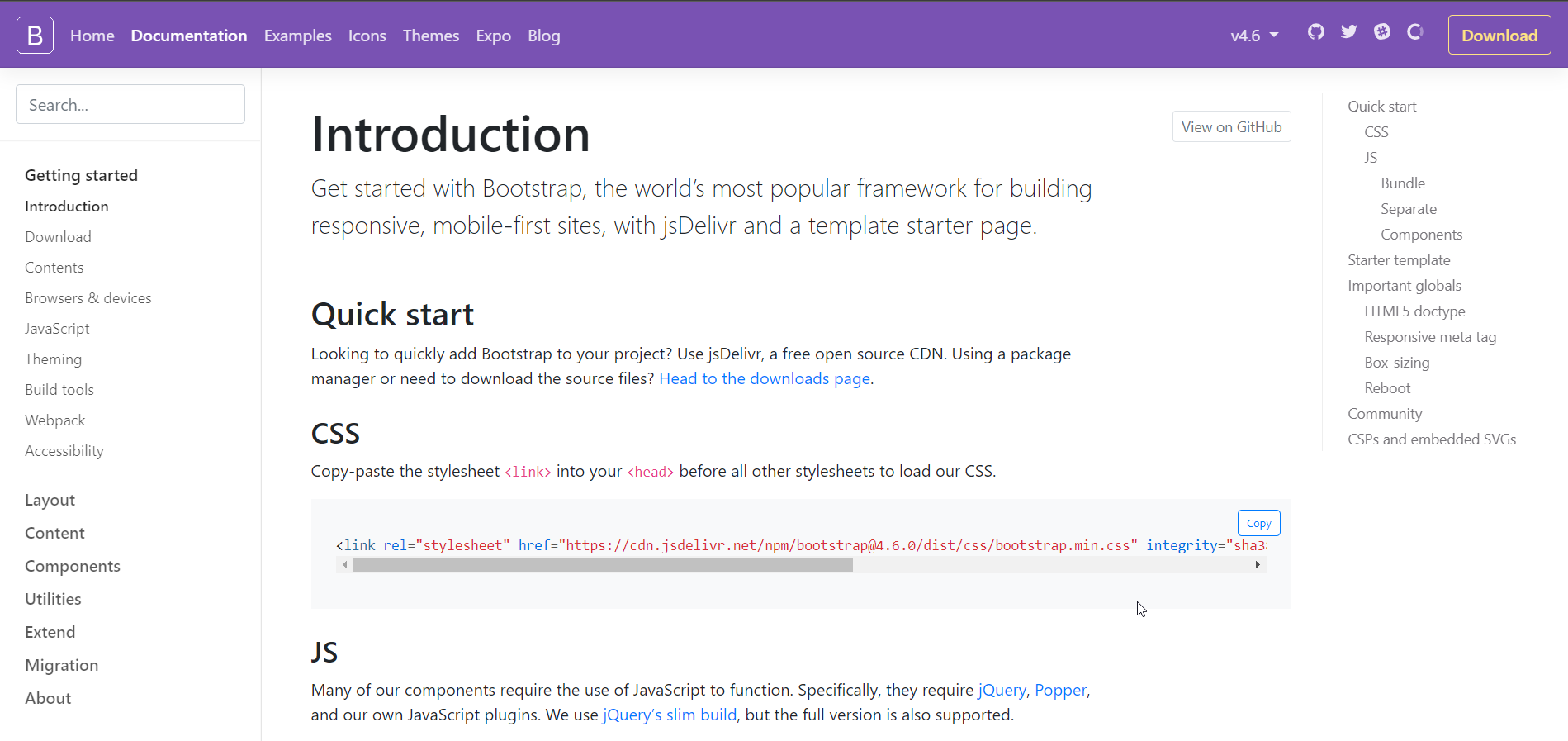
Script.js

This is like the brain of our project, In which we write JavaScript code to make our website work.

Open the index.html type doc then press enter, A boilerplate will appear as shown below:

Open the index.html type doc then press enter, A boilerplate will appear as shown below:

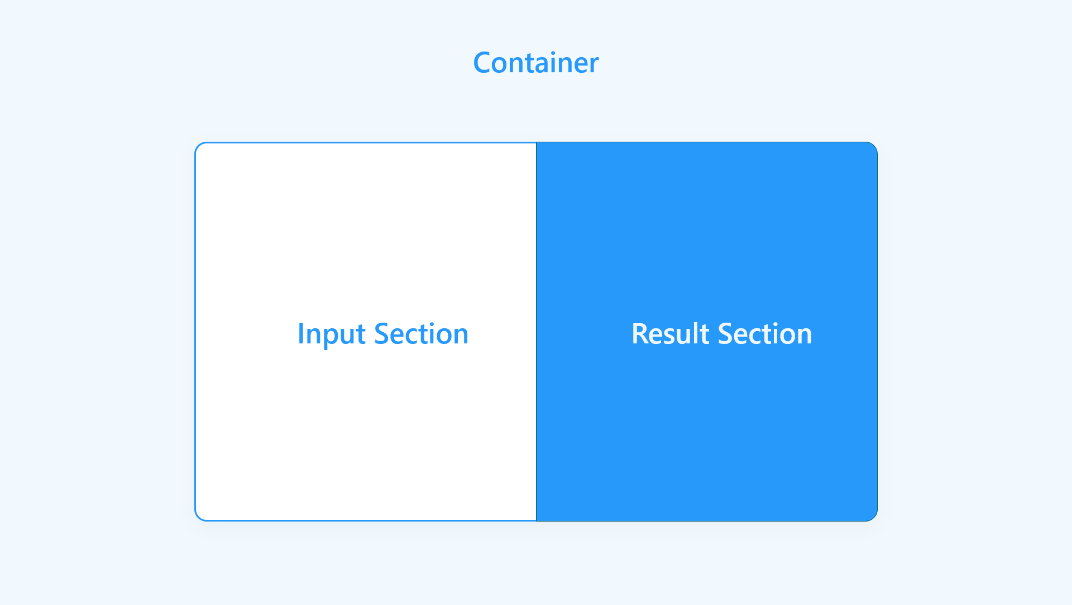
a. Set up Bootstrap 4.6 CSS



## Let's Start Building

The project will be divided into 5 steps:  


### Step 0 : Create HTML structure with Bootstrap classes



Create the container, then make our card-group. The card-group will contain both input-card and result-card. The structure code will look something like this:

<div class=”container”>

<div class=”row”>

<div class=”card-group col-12 col-md-10 offset-md-1 my-md-auto”>

<!—Input and Result Section will come here🡪

</div>

</div>

</div>

Add the input & result card section with custom CSS class inputSection to enter input values, and resultSection to display the result value.

<div class=”card inputSection col-12 col-md-6”>

<div class=”card-body”>

<form>

<div class=”row px-3”>

<div class=”col-12 col-md-11 px-4”>

<span class=”card-title d-block”>Enter Temperature</span>

<label for=”degreeInput” class=”py-sm-2”>Degree</label>

<div class=”input-group”>

<input type=”number” class=”form-control” id=”inputDegree” name=”inputDegree” placeholder=”Enter Degree” value=”0” />

<div class=”input-group-append”>

<select class=”form-control” id=”selectInputDegreeType”>

<option value=”C” selected>&deg;C</option>

<option value=”F”>&deg;F</option>

<option value=”K”>K</option>

</select>

</div>

</div>

<label for=”selectConversionType” class=”py-sm-2”>Convert In</label>

<div class=”input-group d-inline-block”>

<div class=”input-group-append”>

<select class=”form-control” id=”selectConversionType”>

<option value=”F” selected> Fahrenheit (&deg;F) </option>

<option value=”C”>Celcius (&deg;C)</option>

<option value=”K”>Kelvin (K)</option>

</select>

</div>

</div>

<button type=”submit” class=”convertBtn

Btn btn-lg

Col-12 col-md-6

Offset-md-3

Mt-4

Rounded-pill

d-flex

justify-content-center

align-items-center

text-white”> Convert&emsp; <svg class=”mt-1” xmlns=<http://www.w3.org/2000/svg> width=”15px” height=”15px” viewBox=”0 0 21.367 20.826”>

<path id=”Icon\_awesome-arrow-right” data-name=”Icon awesome-arrow-right” d=”M9.085,4.042l1.059-1.059a1.14,1.14,0,0,1,1.617,0l9.271,9.266a1.14,1.14,0,0,1,0,1.617L11.76,23.137a1.14,1.14,0,0,1-1.617,0L9.085,22.078A1.146,1.146,0,0,1,9.1,20.443l5.747-5.475H1.145A1.142,1.142,0,0,1,0,13.823V12.3a1.142,1.142,0,0,1,1.145-1.145H14.85L9.1,5.678A1.138,1.138,0,0,1,9.085,4.042Z” transform=”translate(0 -2.647)” fill=”#fff” />

</svg>

</button>

</div>

</div>

</form>

</div>

</div>

<div class=”card resultSection col-12 col-md-6”>

<div class=”card-body d-flex justify-content-center

Align-items-center”>

<div id=”resultValueSection”>

<div id=”convertedDegree”>32</div>

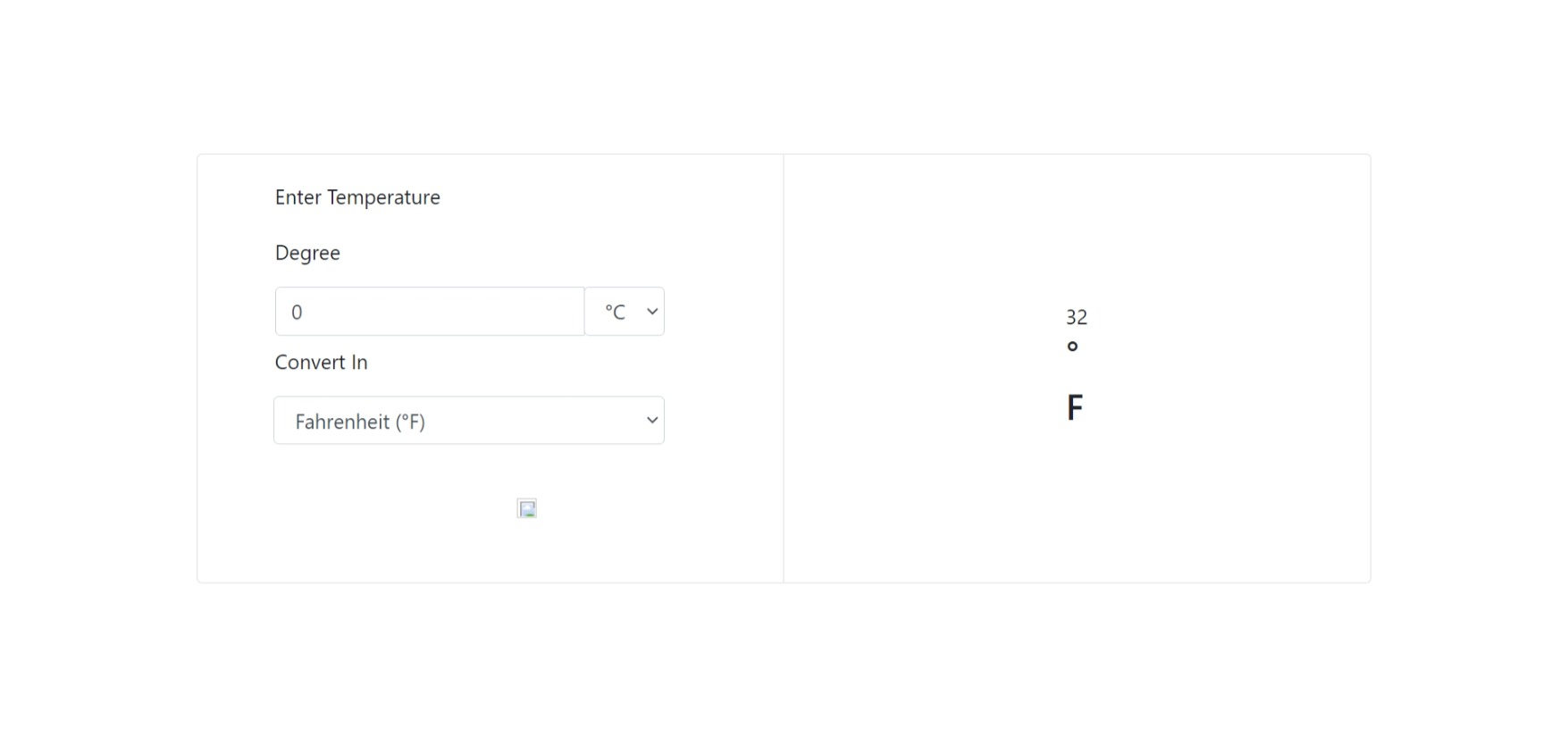
<h3 id=”degree”>&deg;</h3>

<h3 id=”convertedUnit”>F</h3>

</div>

</div>

</div>

By finishing the two steps above, we’ve completed the structure part.. Your output will look something like this:  


Doesn’t look like our final design? Don’t worry. In the next step, we use custom CSS to style and design our website.

Useful references

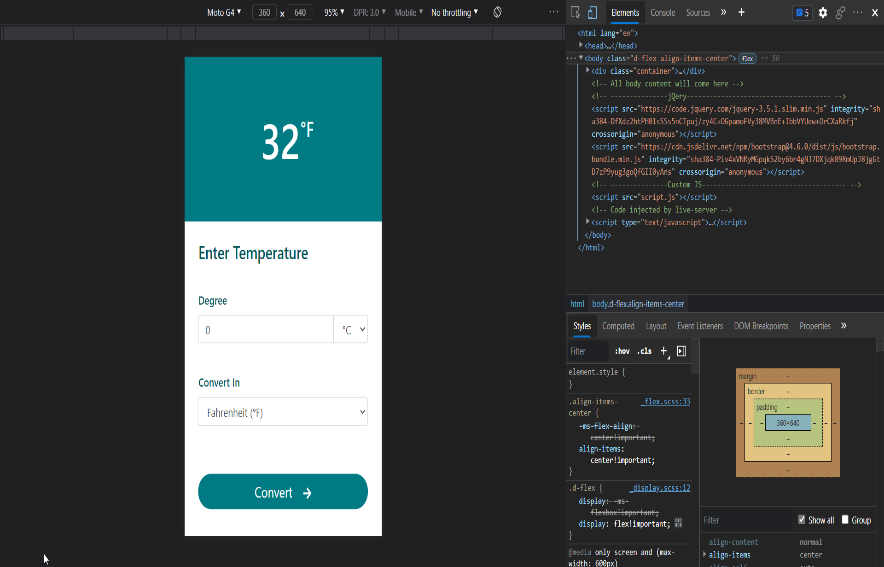
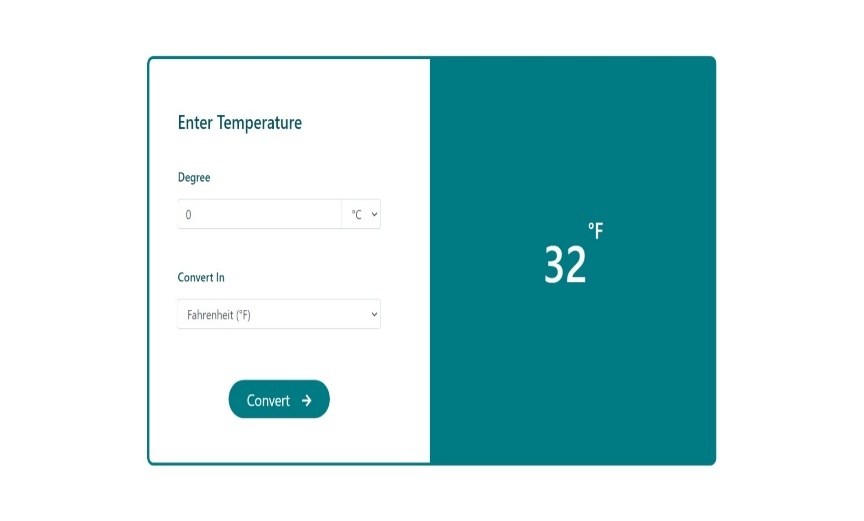
1. [Bootstrap Layout](https://getbootstrap.com/docs/4.6/layout/overview/)
2. [Bootstrap Cards](https://getbootstrap.com/docs/4.6/components/card/)
3. [Bootstrap Input-Groups](https://getbootstrap.com/docs/4.6/components/input-group/)
4. [Bootstrap Forms](https://getbootstrap.com/docs/4.6/components/forms/)

### Step 1: Add custom CSS styling

1. Use the [reference code](https://www.notion.so/472681e481af54f417febc5fc8bbee65) to add styling or add your own custom styles.

Your result will look something like this:

* 1. On small screens or mobile.On medium to large screen or PC:

 On medium to large screen or PC:  


Now that we've completed all our styling work at this stage, simply add JS functionality and then the project is ready to deploy.

[You can build this project yourself! Start the project now.](https://www.codementor.io/projects/web/temperature-converter-website-atx32dy7mf?ref=temperatureconverter)

### Step 2: Create convert functions for all three units.It's time to create the functions that will convert our input degree to the desired unit. First,let's.understand how this works.

Our function will take two arguments: input temperature value and output temperature unit type. The function will check the output unit type, apply the formula according to the output temperature unit type, and return the converted value.

* Fahrenheit Converter ( °F ⇒ °C or K )

This function will only convert the Fahrenheit temperature value into Celsius or Kelvin. So whenever the user enters a °F value, we have to call this function. The reference code is shown below:

// Fahrenheit Converter

Function fTo(inputDegreeValue, conversionDegreeType) {

Let temperature = ‘’;

Switch (conversionDegreeType) {

Case ‘F’:

Temperature = inputDegreeValue;

Break;

Case ‘C’:

Temperature = eval((inputDegreeValue – 32) \* (5 / 9));

Break;

Case ‘K’:

Temperature = eval((inputDegreeValue + 459.67) \* (5 / 9));

Break;

}

Return temperature;

}

Ind Mentors

Find Freelancers

How It Works

Events

BECOME A MENTOR

SIGN UP

LOG IN

Codementor Blog

Last updated Sep 22, 2021

A Step-by-step Tutorial for Building a Temperature Converter Website

You’ll build a simple tool to convert between different temperature units.

A Step-by-step Tutorial for Building a Temperature Converter Website

Building coding projects is one of the best ways to learn coding and build your portfolio. However, sometimes a simple project spec may not be enough to help you build your project. This tutorial will walk you through how to build a Temperature converter website, one of the most popular projects on DevProjects.

You’ll build a simple tool to convert between different temperature units. To follow along, check out the project spec on DevProjects!

Overview of this tutorial:

Introduction

Project Overview

Setting up the project environment

Let’s Start Building

Result

Time to deploy

Recap

Introduction

While Kelvin is the SI Unit of temperature, people generally prefer Fahrenheit or Celsius unit type to measure temperature. We’re going to build a temperature converter that will convert Fahrenheit, Celsius, and Kelvin units to each other, using the most popular CSS framework called Bootstrap 4.6 and JavaScript library – jQuery.

Here is the live demo of Temperature converter website.

Project Overview

In this tutorial, we will create a temperature converter and walk through how the website works. We will deploy this project on GitHub using GitHub pages, a hosting service by GitHub that allows you to publish static websites online directly from repositories. The tutorial will guide you through the step-by-step process of setting up the GitHub repositories before publishing the website. The suggested text editor is VScode, but you can choose any other text editor you like.

⭐ Why Bootstrap

We are using this library because of its variety of components. Additionally, the bootstrap grid system is based on Flex, which provides us with full responsive support for any website. You can read more about it on its official website.

⭐ Why jQuery

While other powerful libraries and frameworks have emerged, jQuery is very beginner friendly and is the best library for practicing and getting started in JavaScript libraries.

jQuery is a small, fast, and feature-rich library. It saves developers a lot of time by completing complicated tasks with just a few lines of code.

⭐ What you need

VSCode or any other text editor

Any browser of your choice

Basics knowledge of jQuery and Bootstrap

A GitHub Account for deployment

Recreate this project on DevProjects. Try it now!

Setting up the project environment

Create a new folder named Temperature Converter and open the folder in VScode. Initially, our workspace will look like this:

Image1.png

Hover the mouse on the sidebar to find the icon for creating new files or folders

Image2.png

Create a sub-folder named assets and the following files: index.html, style.css, and script.js.

fileDirectory.png

Assets

In this folder, we’ll store all icons and any other media used in our project.

Index.html

To create the structure of the website.

Style.css

To add custom CSS styling in our project.

Script.js

This is like the brain of our project, In which we write JavaScript code to make our website work.

Open the index.html type doc then press enter, A boilerplate will appear as shown below:

Image3.png

Open the index.html type doc then press enter, A boilerplate will appear as shown below:

1. Set up Bootstrap 4.6 CSS

Image4.png

Visit Official Bootstrap 4.6.0 Documentation and copy the CDN link of the CSS and JS Bundle. Alternatively, you can also use this Boilerplate in your index.html

<html lang=”en”>

<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”>

<title>Tempreature Unit Converter</title>

<!-- ---------------Bootstrap 4.6-------------------------------------- 🡪

<link rel=”stylesheet”

Href=<https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css>

Integrity=”sha384-B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l”

Crossorigin=”anonymous”>

<!-- ---------------Custom CSS-------------------------------------- 🡪

<link rel=”stylesheet” href=”style.css”>

</head>

<body class=”d-flex align-items-center”>

<!—All body content will come here 🡪

<!-- ---------------jQery-------------------------------------- 🡪

<script src=<https://code.jquery.com/jquery-3.5.1.slim.min.js>

Integrity=”sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj”

Crossorigin=”anonymous”></script>

<script

Src=<https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js>

Integrity=”sha384-Piv4xVNRyMGpqkS2by6br4gNJ7DXjqk09RmUpJ8jgGtD7zP9yug3goQfGII0yAns”

Crossorigin=”anonymous”></script>

<!-- ---------------Custom JS-------------------------------------- 🡪

<script src=”script.js”></script>

</body>

</html>

Why not build this project yourself? Build this project for free.

Let’s Start Building

The project will be divided into 5 steps:

Steps.png

Step 0 : Create HTML structure with Bootstrap classes

Image5.png

Create the container, then make our card-group. The card-group will contain both input-card and result-card. The structure code will look something like this:

<div class=”container”>

<div class=”row”>

<div class=”card-group col-12 col-md-10 offset-md-1 my-md-auto”>

<!—Input and Result Section will come here🡪

</div>

</div>

</div>

Add the input & result card section with custom CSS class inputSection to enter input values, and resultSection to display the result value.

<div class=”card inputSection col-12 col-md-6”>

<div class=”card-body”>

<form>

<div class=”row px-3”>

<div class=”col-12 col-md-11 px-4”>

<span class=”card-title d-block”>Enter Temperature</span>

<label for=”degreeInput” class=”py-sm-2”>Degree</label>

<div class=”input-group”>

<input type=”number” class=”form-control” id=”inputDegree” name=”inputDegree” placeholder=”Enter Degree” value=”0” />

<div class=”input-group-append”>

<select class=”form-control” id=”selectInputDegreeType”>

<option value=”C” selected>&deg;C</option>

<option value=”F”>&deg;F</option>

<option value=”K”>K</option>

</select>

</div>

</div>

<label for=”selectConversionType” class=”py-sm-2”>Convert In</label>

<div class=”input-group d-inline-block”>

<div class=”input-group-append”>

<select class=”form-control” id=”selectConversionType”>

<option value=”F” selected> Fahrenheit (&deg;F) </option>

<option value=”C”>Celcius (&deg;C)</option>

<option value=”K”>Kelvin (K)</option>

</select>

</div>

</div>

<button type=”submit” class=”convertBtn

Btn btn-lg

Col-12 col-md-6

Offset-md-3

Mt-4

Rounded-pill

d-flex

justify-content-center

align-items-center

text-white”> Convert&emsp; <svg class=”mt-1” xmlns=<http://www.w3.org/2000/svg> width=”15px” height=”15px” viewBox=”0 0 21.367 20.826”>

<path id=”Icon\_awesome-arrow-right” data-name=”Icon awesome-arrow-right” d=”M9.085,4.042l1.059-1.059a1.14,1.14,0,0,1,1.617,0l9.271,9.266a1.14,1.14,0,0,1,0,1.617L11.76,23.137a1.14,1.14,0,0,1-1.617,0L9.085,22.078A1.146,1.146,0,0,1,9.1,20.443l5.747-5.475H1.145A1.142,1.142,0,0,1,0,13.823V12.3a1.142,1.142,0,0,1,1.145-1.145H14.85L9.1,5.678A1.138,1.138,0,0,1,9.085,4.042Z” transform=”translate(0 -2.647)” fill=”#fff” />

</svg>

</button>

</div>

</div>

</form>

</div>

</div>

<div class=”card resultSection col-12 col-md-6”>

<div class=”card-body d-flex justify-content-center

Align-items-center”>

<div id=”resultValueSection”>

<div id=”convertedDegree”>32</div>

<h3 id=”degree”>&deg;</h3>

<h3 id=”convertedUnit”>F</h3>

</div>

</div>

</div>

By finishing the two steps above, we’ve completed the structure part.. Your output will look something like this:

Demo1.jpeg

Doesn’t look like our final design? Don’t worry. In the next step, we use custom CSS to style and design our website.

Useful references

Bootstrap Layout

Bootstrap Cards

Bootstrap Input-Groups

Bootstrap Forms

Step 1: Add custom CSS styling

Use the reference code to add styling or add your own custom styles.

Your result will look something like this:

On small screens or mobile:

Image6.png

On medium to large screen or PC:

Image7.jpeg

Now that we’ve completed all our styling work at this stage, simply add JS functionality and then the project is ready to deploy.

You can build this project yourself! Start the project now.

Step 2: Create convert functions for all three units

It’s time to create the functions that will convert our input degree to the desired unit. First, let’s understand how this works.

Diagram.png

Our function will take two arguments: input temperature value and output temperature unit type. The function will check the output unit type, apply the formula according to the output temperature unit type, and return the converted value.

Fahrenheit Converter ( °F ⇒ °C or K )

This function will only convert the Fahrenheit temperature value into Celsius or Kelvin. So whenever the user enters a °F value, we have to call this function. The reference code is shown below:

// Fahrenheit Converter

Function fTo(inputDegreeValue, conversionDegreeType) {

Let temperature = ‘’;

Switch (conversionDegreeType) {

Case ‘F’:

Temperature = inputDegreeValue;

Break;

Case ‘C’:

Temperature = eval((inputDegreeValue – 32) \* (5 / 9));

Break;

Case ‘K’:

Temperature = eval((inputDegreeValue + 459.67) \* (5 / 9));

Break;

}

Return temperature;

}

Celsius Converter ( °C ⇒ °F or K )

This function will only convert the Celsius temperature value into Fahrenheit or Kelvin. So whenever the user enters a °C value, we have to call this function. The reference code is shown below:

// Celcius Converter

Function cTo(inputDegreeValue, conversionDegreeType) {

Let temperature = ‘’;

Switch (conversionDegreeType) {

Case ‘C’:

Temperature = inputDegreeValue;

Break;

Case ‘F’:

Temperature = eval((inputDegreeValue \* (9 / 5)) + 32);

Break;

Case ‘K’:

Temperature = eval(inputDegreeValue + 273.15);

Break;

}

Return temperature;

}

Kelvin Converter ( K ⇒ °F or °C )

This function will only convert Kelvin temperature value into Fahrenheit or Celsius. When a user enters a K value, we have to call this function. The reference code is shown below:

// Kelvin Converter

Function kTo(inputDegreeValue, conversionDegreeType) {

Let temperature = ‘’;

Switch (conversionDegreeType) {

Case ‘K’:

Temperature = inputDegreeValue;

Break;

Case ‘F’:

Temperature = eval((inputDegreeValue – 273.15) \* (9 / 5) + 32);

Break;

Case ‘C’:

Temperature = eval((inputDegreeValue – 273.15));

Break;

}

Return temperature;

}

Function convertInputDegree() {

Let inputDegree = parseInt($(‘#inputDegree’).val());

Let selectInputDegreeType = $(‘#selectInputDegreeType’).val();

Let conversionType = $(‘#selectConversionType’).val();

Let resultValue = “”;

Switch (selectInputDegreeType) {

Case “C”:

resultValue = cTo(inputDegree, conversionType);

break;

case “F”:

resultValue = fTo(inputDegree, conversionType);

break;

case “K”:

resultValue = kTo(inputDegree, conversionType);

break;

}

// To prevent NaN

If (isNaN(inputDegree)) {

$(‘#convertedDegree’).text(‘’);

Return;

}

// To update the Degree Unit

$(‘#convertedUnit’).text(conversionType)

// To update the Degree Value

If (conversionType === selectInputDegreeType) {

$(‘#convertedDegree’).text(inputDegree);

} else {

Return $(‘#convertedDegree’).text(resultValue.toFixed(2));

}

}

Step 4: Update results in real-time

We’re almost finished building our temperature converter! However, there’s one but if the users have to click the submit button again and again, it will provide a bad user experience. We can fix this by updating the output value in real-time. To do this, we can call the function convertInputDegree() when users select a different input or output unit. The code should look like this:

// Realtime Update

$(‘#inputDegree’).on(‘input’, () => convertInputDegree());

$(‘#selectInputDegreeType’).change(() => convertInputDegree());

$(‘#selectConversionType’).change(() => convertInputDegree());

Result

Once we’ve completed all of the above steps, this is what the final preview will look like:

