Topic	MAPS	
Class Description	**Students will learn to render interactive maps on the web using the Mapbox GL JS library.**	
Class	**C178**	
Class time	**45 mins**	
Goal	• Create a Mapbox account. • Learn how to render maps in web applications using Mapbox GL JS library.	
Resources Required	• Teacher Resources: ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ smartphone ○ earphones with mic ○ notebook and pen • Student Resources: ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ smartphone ○ earphones with mic ○ notebook and pen	
Class structure	**Warm-Up** **Teacher-led Activity** **Student-led Activity** **Wrap-Up**	**5 mins** **15 mins** **20 mins** **5 mins**
WARM-UP SESSION - 5 mins		
CONTEXT • **Understand the Mapbox GL JS library.** • **Learn to render maps using Mapbox API.**		

Teacher Starts Slideshow  **Slide 1 to 3** Refer to speaker notes and follow the instructions on each slide.	
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session  **Slide 4 to 13**	
Following are the session deliverables: • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	

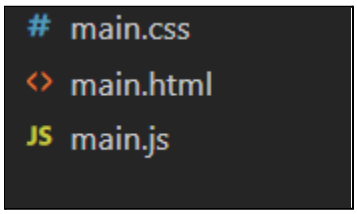
Class Steps	Teacher Action	Student Action
Step 1: Warm-Up (5 mins)	Hi, how are you? Great!	**ESR:** I am good!
	Well now we know how to write code in jQuery, we can keep practicing to get more and more comfortable with it. Today we are going to learn about maps.	

	Before that let's discuss a few things related to maps. ***Note**: Encourage the student to discuss what they know about the technology and help to be more involved.*	***Note**: The student discusses his/her views with the teacher.*
	Can you tell me what are maps?	**ESR**: Maps are a representation of a place.
	Yes! Maps are basically a 2D representation drawn to scale any 3D spaces around us.	
	Do you know when the first Map was drawn?	**ESR**: Varied.
	It was around 5000 years ago! We humans have always been fascinated by the world we are living in since prehistoric times.	
	Do you know cave people used to draw paintings on the cave walls?	**ESR**: Yes.
	When humans were evolving but still living in the cave, they learned the art to write and draw. They used symbols to draw something related to their religious beliefs. But recent findings tell that some of these paintings were maps of huntings	

	areas, or routes of traveling, or even maps of stars. So you can imagine humans have been making maps since long ago. Do you know what we call the art of making maps? Amazing! Cartographers study and design maps on the basis of geographical areas. A few 1000 years ago, people across the world started traveling a lot to discover new things. Along with this they also discovered new places and started making maps. And slowly the complete world was drawn on the paper. Can you tell me when the first world map was drawn? First world map was drawn around 1500 years ago! Until the digital era came, maps were drawn on paper. Now everyone is using the most popular digital maps application, Google Maps!	**ESR**: Yes. It is Cartography. **ESR**: Varied.

	It has made our lives so convenient, whenever we want to find places! Not only finding places but also navigating from one place to another. Today we will be learning how we use digital maps in our web application. Are you excited? Let's get started then.	**ESR**: Yes.
Teacher Ends Slideshow 		
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
CHALLENGE • **Render Maps using Mapbox GL JS library.**		
Step 2: Teacher-led Activity (15 mins)	*<The teacher clones the activity from the Teacher Activity 1 and shows the output>.* **[Teacher Activity 1]** First we will take a look at the boilerplate code. Here we have the 3 files: • **main.html**: The HTML content file	

- **main.css**: Styling file
- **main.js**: JavaScript file



Let's begin with the HTML file.

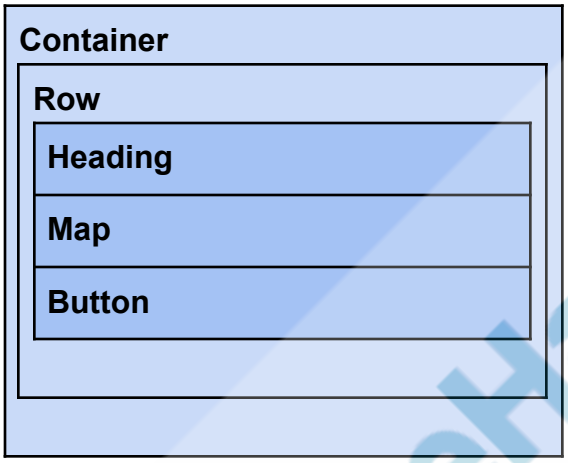
The jQuery & Bootstrap library have been included in the <head> tag.

We need Mapbox APIs to be included in the <head>, which we will discuss in a while.

```
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="initial-scale=1,maximum-scale=1,user-scalable=no">

    <title>Navigation</title>

    <!-- JQuery -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"
        integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>

    <!-- Bootstrap -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
        integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.bundle.min.js"
        integrity="sha384-LtrjvnR4Twt/qOuYxE721u19sVFLVSA4hf/rRt6PrZTmiPltdZcI7q7PXQBYTKyf"
        crossorigin="anonymous"></script>
</head>
```

Let's take a look at the <body> content design.

We have the following DOM structure:




```
<!-- Body HTML -->
<div class="container">
    <div class="row">
        <div class="col-sm-12 col-md-12 col-lg-12 text-center p-5">
            <h1>AR NAVIGATION</h1>
        </div>
        <div class="col-sm-12 col-md-12 col-lg-12 text-center">
            <div id="map"></div>
        </div>
        <div class="col-sm-12 col-md-12 col-lg-12 text-center p-5">
            <button id="navigate-button">Navigate</button>
        </div>
    </div>
</div>

<!-- CSS and JS -->
<link rel="stylesheet" href="./main.css" />
<script src="./main.js" type="text/javascript"></script>
```

Note 1: *Some styling has been added to these elements. Check the* ***main.css*** *file.*
Note 2*: CSS and JavaScript files are added in the <body> below the map container. This will make the map container load before any other actions are performed on it.*

Web Page Template

Once we have the HTML content structure of the page ready, let's see how we can render maps on the web page.

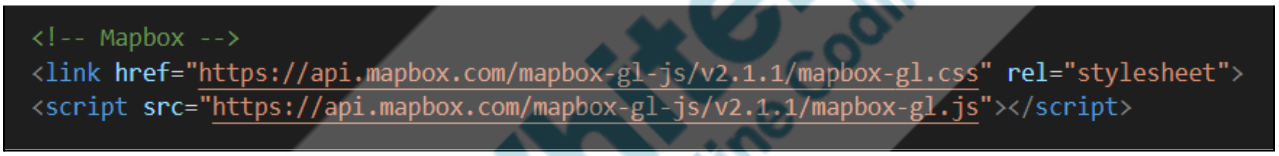
For this we are going to use **Mapbox GL JS library.**

Mapbox GL JS is a JavaScript library that is designed to render interactive maps in the web application.

This library uses **WebGL** API in the background.

WebGL stands for **W**eb **G**raphics **L**ibrary.

	We'll see what that means, but before that can you tell me what you understand by the "Graphics"? Perfect! Graphics are a pictorial representation of any object. Pictures help us understand better rather than just explaining in words. Maps are a good example of graphics. You can easily understand the way from one place to another by looking at a map, instead of someone explaining the way. **WebGL is a JavaScript API which helps us to render 2D and 3D graphics**! Remember we are using A-Frame for rendering 3D objects? A-Fame is built using Three.js library which uses WebGL behind the scenes to render 3D objects. Now since we want to use Mapbox library APIs, we should include these in the <head> tag.	**ESR:** Graphics are pictures and drawings used to represent something. **ESR**: Yes.

	<script src='https://api.mapbox.com/mapbox-gl-js/v2.2.0/mapbox-gl.js'></script> Mapbox also provides their own CSS to use for the maps. We will need to include that also in the <head> tag. <link href='https://api.mapbox.com/mapbox-gl-js/v2.2.0/mapbox-gl.css' rel='stylesheet' />	
	 	
	Now to access the properties and methods of these APIs we will need an access token. Do you know what an access token is? Access tokens are a way to know which user is accessing the APIs. Access tokens are entry points to call any of the APIs that we want to use. To get the access token, we will have to first create a Mapbox account.	**ESR:** Varied.

	<The teacher creates a Mapbox account by following the steps given in the reference document below.> **Create Mapbox Account** **Steps to Create Account Document**	
	Mapbox Account Dashboard 	
	We will use the "**Default public token**"" to access the API. We are going to use the **mapboxgl** global variable to use the **accessToken** property. *<The teacher copies the "**Default public token**" from the Mapbox account dashboard and assigns it to mapboxgl.accessToken property in **main.js** file.>*	

```
// Initializing Mapbox
mapboxgl.accessToken = 'pk.[redacted]A';
```

Now we are going to use the **Map Object.**

```
var map = new mapboxgl.Map({

})
```

This object is used to render the maps on a web page.

<The teacher adds the Map object.>

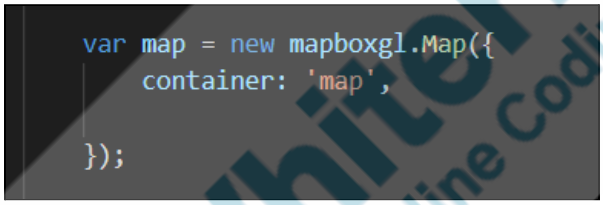
```
// Initializing Mapbox
mapboxgl.accessToken = 'pk.[redacted]';

var map = new mapboxgl.Map({

});
```

There are many parameter options that we can pass into the Map object to set up the map.

We will learn about a few of them today.

The first parameter that we need is the HTML **container**.

That means we need to tell the Map object which section of the page will show the map.

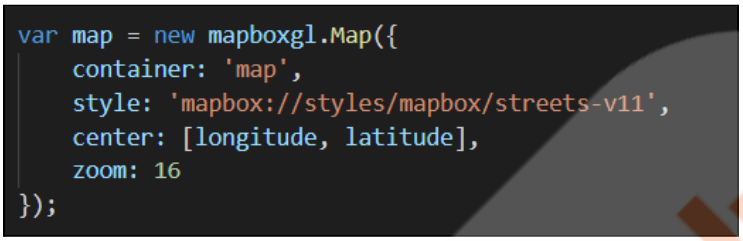
	Do you remember where in the HTML we have the **map** ? Yes. Great! We have used a **map** as the id of the . To set the container option we use the id of the HTML element. *<The teacher adds the **container** option in the Map object.>*	**ESR**: In between the heading and the button on the page.
	 	
	Now we need to set up the **style** of the map, which determines the look and feel of the map. The Mapbox style document is JSON object data. To set the map style we can: • Create our own style document in the format specified by Mapbox. • Use default styles provided by the Mapbox.	

	Well we are going to use one of [Mapbox's default styles]().	
		
	Next parameter option is **center**. This is the initial location point of the map. The location points are given in longitude and latitude. Can you tell me what longitude and latitude are? Amazing! These location points are stored in a standard format known as, **GeoJSON**, which has been designed to store various geographic data! It is based on the JSON format. We will not focus deeply on how GeoJSON files are written. But we have to know that Mapbox takes the following **longitude, latitude**	**ESR**: Longitude and latitude are a geographical coordinate system, which are to locate a point on the earth.

	coordinate order as per GeoJSON format, that means the first value must be longitude. The default values are [0,0]. How to find location coordinates using Google maps? ***Important Note**: The order of the location coordinates in Google Maps is **[latitude, longitude]**. It is different from standard map applications, which have reverse order as [longitude, latitude].* *<The teacher sets the **longitude, latitude variables** and the **centre** options.>*	

	
	`let latitude=22.7868542, longitude=88.3643296;`
	
	`var map = new mapboxgl.Map({`
	`container: 'map',`
	`style: 'mapbox://styles/mapbox/streets-v11',`
	`center: [longitude, latitude],`
	
	`});`	

	Then we have to set the **zoom** parameter option. This specifies the initial zoom level of the map. There are 22 zoom levels available in Mapbox API, starting from 0. 0 is for the complete Earth view.	

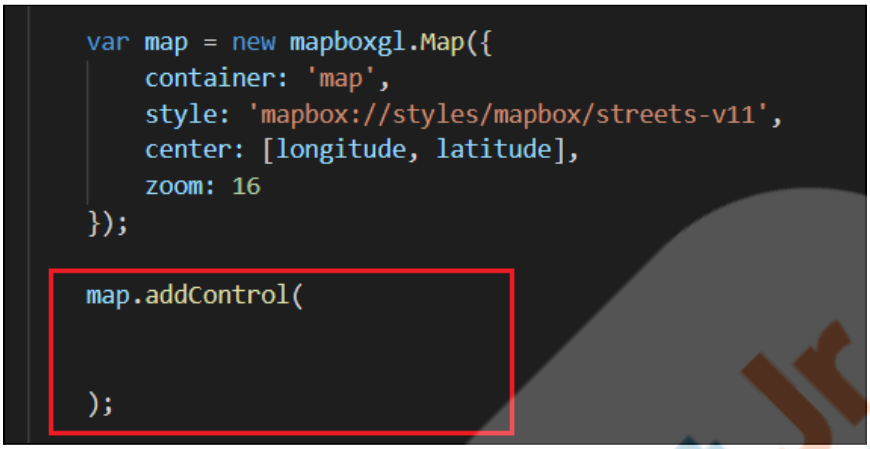
	As the zoom level number increases the map gets closer to the place.	
		
	Now let's run the app and see the output.	

We can now see that the map is rendered on the page.

The map's set at initial latitude and longitude.

Test the output with zoom level 0. We should be able to see the whole world map.

	We got the map. Now let's add some of the controls on the map that can let us select some location points on the map. Where would you like to start? Well let's begin with adding the control to find the **current location**. To add controls in the Mapbox maps, the Map class provides a method called **addControl(control, position)**.	**ESR**: Varied.



```
var map = new mapboxgl.Map({
    container: 'map',
    style: 'mapbox://styles/mapbox/streets-v11',
    center: [longitude, latitude],
    zoom: 16
});

map.addControl(


);
```

	control: control object to be added on the map. There a few predefined controls available in the Mapbox, like: • **GeolocateControl**: This control provides the button that will use the browser's location to find the current location of the user. • **NavigationControl**: This control contains the zoom in and zoom out buttons. • **position**: position of the control on the page. • **valid values**: 'top-left' , 'top-right' , 'bottom-left' , and bottom-right'. **Default**: 'top-right'. Which control will we use to find the current location then?	**ESR:** GeolocateControl.

	Yes. Let's add the **GeolocateControl** at the top-right corner of the map.	

```
map.addControl(
    new mapboxgl.GeolocateControl({

    })
);
```

	The **GeolocateControl** can also have many options inside it. But we are going to use:
	
	positionOptions:
	default:
	{
	enableHighAccuracy:false,
	timeout:6000
	}
	
	trackUserLocation:
	default: false	

```
map.addControl(
    new mapboxgl.GeolocateControl({
        positionOptions: {
            enableHighAccuracy: true
        },
        trackUserLocation: true
    })
);
```

	Now let's test the output.	

	That's really interesting to render the map by ourselves, right? Now you will render the map using the Mapbox. You will add the control to find directions from one point to another in the map! Are you excited?	**ESR**: Yes! **ESR**: Yes!

Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
Teacher Starts Slideshow  **Slide 14 to 15** Refer to speaker notes and follow the instructions on each slide.		
We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.		
Teacher Ends Slideshow 		
STUDENT-LED ACTIVITY - 20 mins		
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start screen share.** • **Teacher gets into fullscreen.**		
ACTIVITY • **Render maps using Mapbox APIs.** • **Add direction controls to the map.**		
Step 3: Student-led Activity (20 mins)	*The teacher guides the student to clone the code from Student Activity 2.* ***[Student Activity 2]*** ***Note**: The student will continue to add new functionality after teacher activity.*	

	Guide the student to create a Mapbox account by following the steps given in the reference document. **Create Mapbox Account** **Steps to Create Account Document**	
Mapbox Account Dashboard 		
	Guide the student to add accessToken property.	
`// Initializing Mapbox` `mapboxgl.accessToken = 'pk.[redacted]A';`		
	Guide the student to set Map Object to render maps.	

```
let latitude=22.7868542, longitude=88.3643296;
var map = new mapboxgl.Map({
    container: 'map',
    style: 'mapbox://styles/mapbox/streets-v11',
    center: [longitude, latitude],
    zoom: 16
});
```

Now let's add the control to find direction between one point to another.

For this we will have to use Mapbox Direction API (plugin) for directions.

<script src="https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-directions/v4.1.0/mapbox-gl-directions.js"></script>

We will also use Mapbox's style for this APIs.

<link rel="stylesheet" href="https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-directions/v4.1.0/mapbox-gl-directions.css" type="text/css">

Guide the student to add the API source.

```
<!-- Mapbox Direction Plugin-->
<script
    src="https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-directions/v4.1.0/mapbox-gl-directions.js"></script>
<link rel="stylesheet"
    href="https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-directions/v4.1.0/mapbox-gl-directions.css"
    type="text/css">
```

<table>
<tr><td></td><td>Guide the student to add the map controls.
Create a new MapboxDirection object using accessToken.
Set the position of the control on the map.</td><td></td></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><td></td><td>Guide the student to test the output.</td><td></td></tr>
</table>

Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 5 mins

Teacher Starts Slideshow
Slide 16 to 20

Activity details
Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

Continue WRAP-UP Session
Slide 21 to 26

Activity Details

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	Student Action
You get Hats off for your excellent work!	*Make sure you have given at least 2 Hats Off during the class for:* 

PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections		
Teacher Clicks 		

Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.*
	
	Use these as guiding questions:
	• What happened today?
	  ○ Describe what happened.
	  ○ The code I wrote.
	• How did I feel after the class?
	• What have I learned about programming and developing games?
	• What aspects of the class helped me? What did I find difficult?
		The student uses the markdown editor to write their reflections in a reflection journal.

Activity	Activity Name	Links
Teacher Activity 1	Create Mapbox Account	https://account.mapbox.com/auth/signin/?route-to=%22https://account.mapbox.com/%22
Teacher Activity 2	Mapbox Account Creation Steps	https://obj.whitehatjr.com/b16395ba-85d6-40ec-8d46-44c27a20a27a.pdf
Teacher Activity 3	Boilerplate Code	https://github.com/whitehatjr/PRO-C178-Boilerplate
Teacher Activity 4	Teacher Reference Code	https://github.com/whitehatjr/PRO-C178-Code-Ref

Teacher Activity 5	Mapbox GL JS library Reference	https://docs.mapbox.com/mapbox-gl-js/api/
Teacher Activity 6	Map Object Reference	https://docs.mapbox.com/mapbox-gl-js/api/map/
Teacher Activity 7	Map Direction API Reference	https://docs.mapbox.com/mapbox-gl-js/example/mapbox-gl-directions/
Student Activity 1	Create Mapbox Account	https://account.mapbox.com/auth/signin/?route-to=%22https://account.mapbox.com/%22
Student Activity 2	Boilerplate Code	https://github.com/whitehatjr/PRO-C178-Boilerplate
Student Activity 3	Mapbox GL JS library Reference	https://docs.mapbox.com/mapbox-gl-js/api/
Student Activity 4	Map Object Reference	https://docs.mapbox.com/mapbox-gl-js/api/map/
Student Activity 5	Map Direction API Reference	https://docs.mapbox.com/mapbox-gl-js/example/mapbox-gl-directions/
Teacher Reference 1	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/02b4d0c2-82ce-4835-ad91-77cda418a4a4.pdf
Teacher Reference 2	Project Solution	https://github.com/whitehatjr/PRO-C178-Project-Solution
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/90afafcd-428a-4403-8b2f-f5e439d0fdc5.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/7ec7f047-ab2a-4688-8d41-140c1f7586a1.pdf