WhiteHat Jr
Live online Coding for Kids

Topic	CURRENT LOCATION & QUERY PARAMETERS	
Class Description	Students will understand how they can write JavaScript functions to get the user's current location and use query parameters to pass data from one page to the other.	
Class	C179	
Class time	45 mins	
Goal	● Write JavaScript functions to get the current location of the user. ● Use query parameters to pass information from one page to another.	
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		

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

CONTEXT
• **Understanding the browser's navigator object.** • **Understanding the query parameters.**
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 11**
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!
	Can you tell me what we have learned in the previous class?	**ESR:** We learned about the Mapbox API and integrated

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

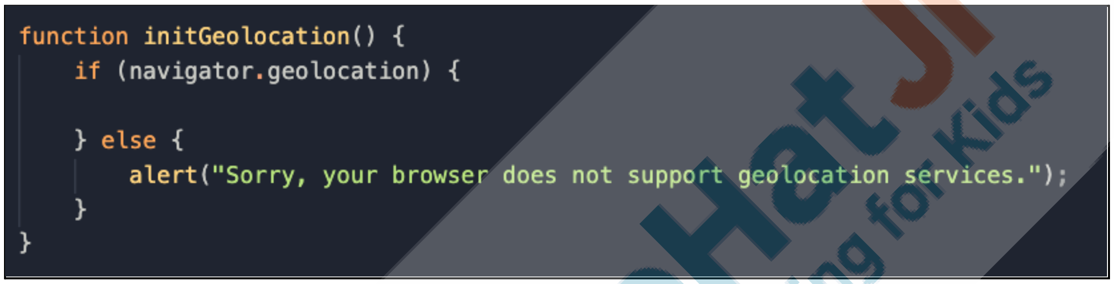
	Superb! Now we hard-coded the latitude and longitude to mark the center of the map in the previous class, but our user might be in any part of the word! It is better if we fetch the user's latitude and longitude based on their location. Also, in our project, we will have 2 pages. First is the one we built in the previous class, but once the users click on the navigate button, we want to take them to a different page from where they will be able to navigate. For this, we will need the latitude and the longitude they chose as their destination in order to help them navigate, and we will need to pass this information to a different page somehow. For that, we will be using parameters. In today's class, we will be learning about query parameters with JavaScript and writing some JavaScript functions to fetch the user's location. Are you excited? Let's get started then!	it in our page along with the directions API. **ESR**: Yes.

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

Teacher Ends Slideshow		
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
<u>CHALLENGE</u> ● **Get the latitude and longitude based on the user's location.**		
Step 2: **Teacher-led Activity** **(15 mins)**	*<The teacher clones the activity from the Teacher Activity 1 and shows the output>.* This is the code from the previous class. In JavaScript, there is a special object called **“navigator”** that contains a lot of things. You can find out all the interesting things it has by referring to *Student Activity 1.* *(The teacher refers to Teacher Activity 2.)* That one thing that we will be using is its **geolocation** property. It returns a geolocation object that can be used to locate the user’s position.	 *The student refers to Student Activity 1.*

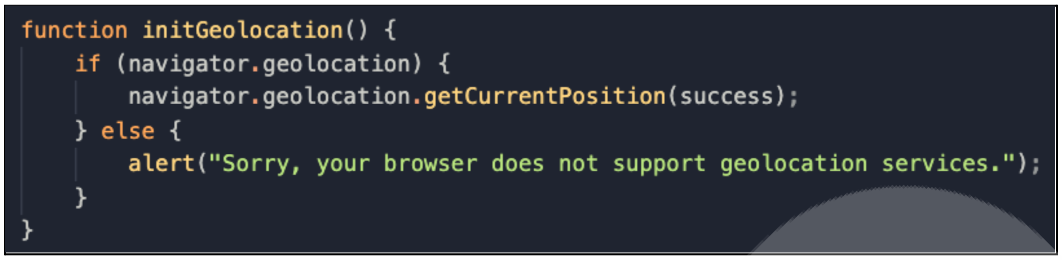
© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

	Remember that this is browser-dependent. It could be the case where the user's browser does not provide geolocation services and hence, it will not work. Keeping all this in mind, let's create a function in our ***main.js*** file -	


`function initGeolocation() {`
`    if (navigator.geolocation) {`
 	
`    } else {`
`        alert("Sorry, your browser does not support geolocation services.");`
`    }`
`}`
 	
Here, we are checking if there exists a **navigator.geolocation** object or not. If it doesn't, we are throwing an alert for the user informing them that the browser does not support geolocation services.		

	Okay, but now what? How do we get the latitude and longitude of the user based on their location? The object that ***navigator.geolocation*** returns has a function called ***getCurrentPosition()*** which takes a function it should call if it can successfully get the current position of the user. It provides the user's current position to the success callback function in the form of an argument. Let's see how it works -	**ESR:** Varied.

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

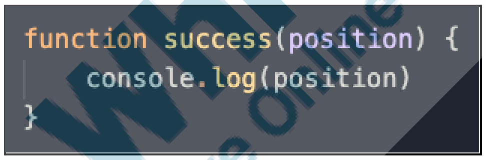


```
function initGeolocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(success);
    } else {
        alert("Sorry, your browser does not support geolocation services.");
    }
}
```

Now here, we have added a new line ***navigator.geolocation.getCurrentPosition(success)*** in the if statement of our **initGeolocation()** function.

This means that if the **getCurrentPosition()** works properly and is able to get the user's location, it will call the **success()** function in which it will provide the user's current position as an argument.

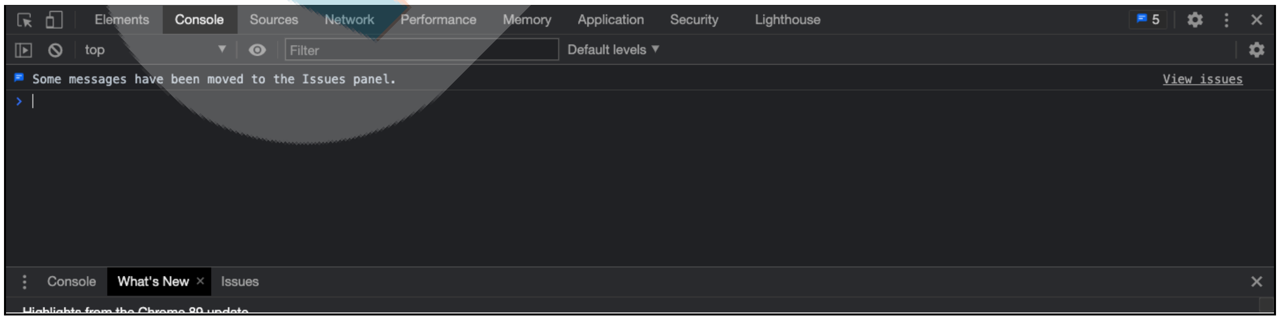
Now in order for this to work, we will also have to create another function called ***success()*** -



```
function success(position) {
    console.log(position)
}
```

Okay, now the function should work.

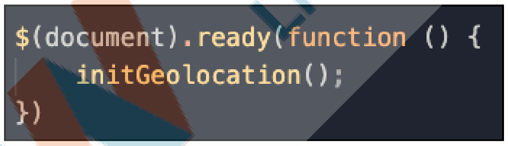
Let's now open our ***main.html*** and see what we get in the console -



```
Some messages have been moved to the Issues panel.
> |
```

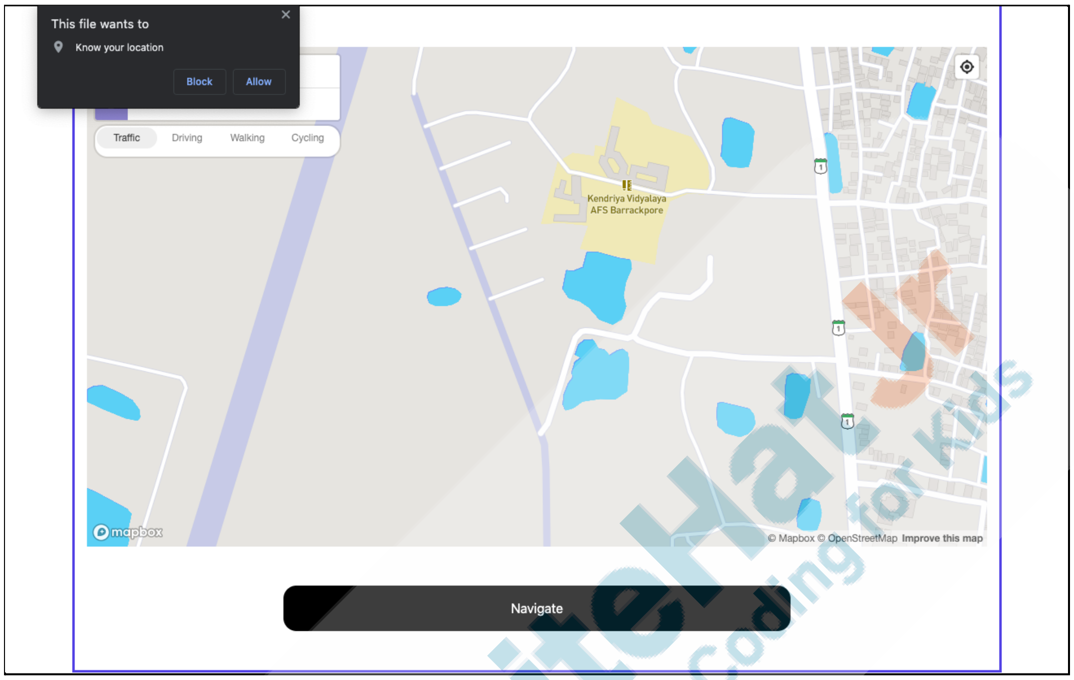
Highlights from the Chrome 89 update

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

	As we can see, our console is empty. Can you tell me why?
	
	That's right! What should we do then?
	
	And why should we do it?
	
	Awesome! Let's do that and see what we get -
	
	
	`$(document).ready(function () {`
	`    initGeolocation();`
	`})`
		ESR:
		The console is empty because we are not calling the ***initGeolocation()*** function anywhere and it is not executing.
		
		ESR:
		We should have a **$(document).ready()** function and call this function inside it.
		
		ESR:
		So that this function can get executed as soon as the page is loaded and we get the current position of the user.

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

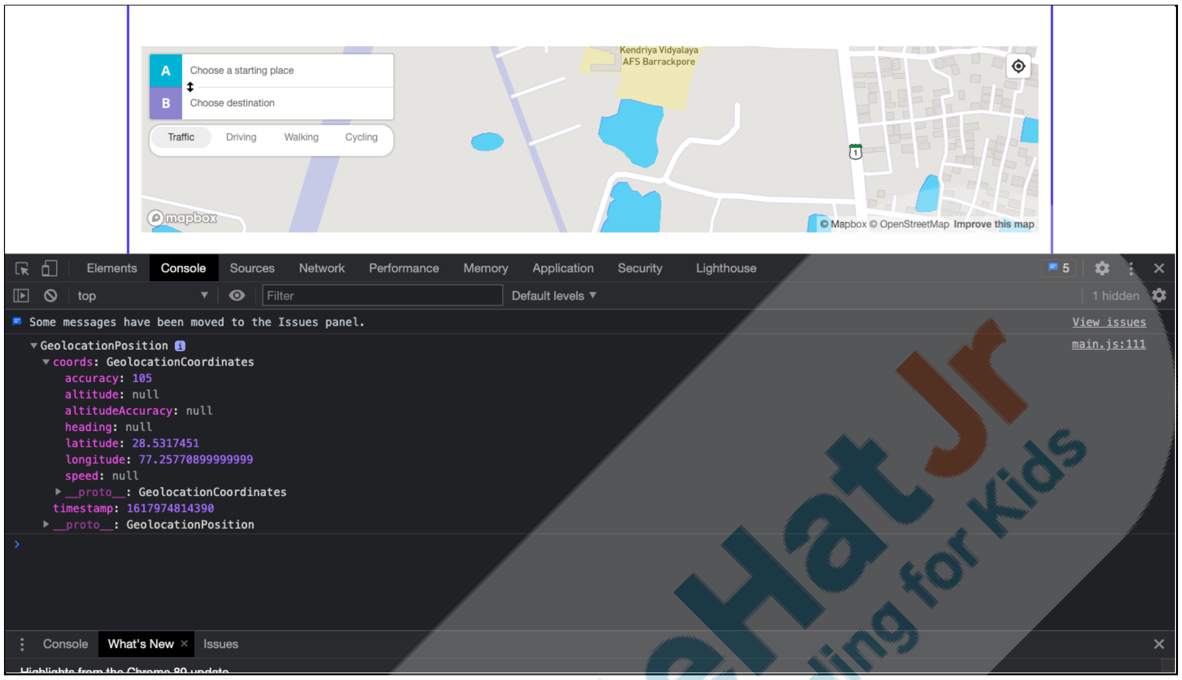
On opening in the browser, we see -



We see a popup in the top-left which asks us if we want to Allow our file to know our location.

Let's click on allow and see if we see our position in the console -

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.



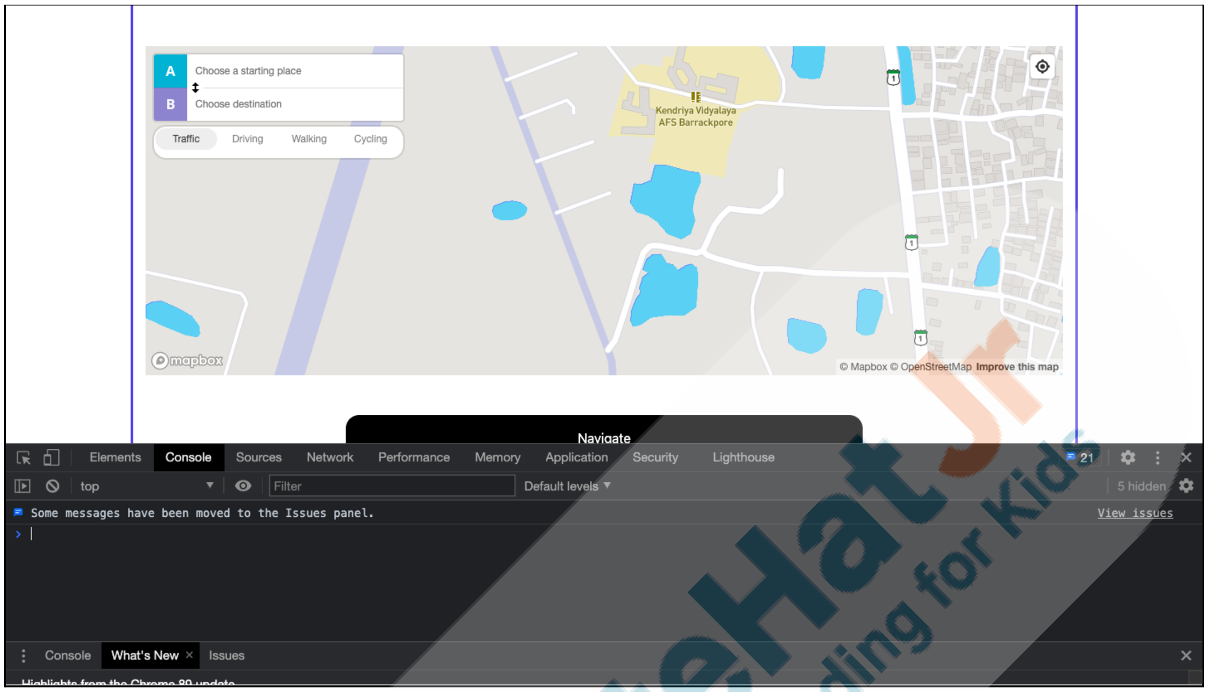
```
Some messages have been moved to the Issues panel.
▼GeolocationPosition
  ▼coords: GeolocationCoordinates
     accuracy: 105
     altitude: null
     altitudeAccuracy: null
     heading: null
     latitude: 28.5317451
     longitude: 77.25770899999999
     speed: null
   ▶__proto__: GeolocationCoordinates
   timestamp: 1617974814390
  ▶__proto__: GeolocationPosition
```

Great! We now have an object which contains a key called ***coords***. Inside this key, we can find the ***latitude and the longitude.***

Let's reload this page and see what happens if we don't allow the browser to know the position -

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

WhiteHat Jr
Live online Coding for Kids



A Choose a starting place
B Choose destination
Traffic Driving Walking Cycling
Kendriya Vidyalaya AFS Barrackpore
mapbox
© Mapbox © OpenStreetMap Improve this map
Navigate
Elements Console Sources Network Performance Memory Application Security Lighthouse
top Filter Default levels 5 hidden
Console What's New Issues

```
Some messages have been moved to the Issues panel.                View issues
```

This time around, we don't see any output on the console. Therefore it is important for the user to allow the browser to know their current location in order to make our project work.

For that, let's add an alert in our **$(document).ready()** function to **"alert"** the users about it -

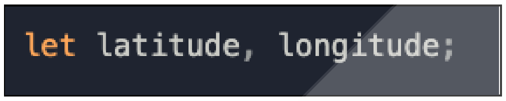
```
$(document).ready(function () {
    alert("Please allow the device to know your location!")
    initGeolocation();
})
```

Now, we're all set!

We are rendering the map as soon as a user opens the page, but since we are going to use our newly received latitude and longitude based on the user's current location, we will have to

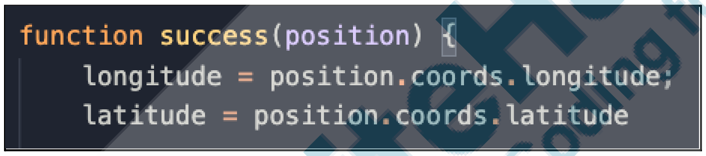
© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

	wait to get this information first and then render the map! We can move our map variable and **addControl()** functions inside our success function for that, so that everything happens in order -	



```
let latitude, longitude;
```

We will first remove the values from latitude and longitude at the top, and just create the variables.

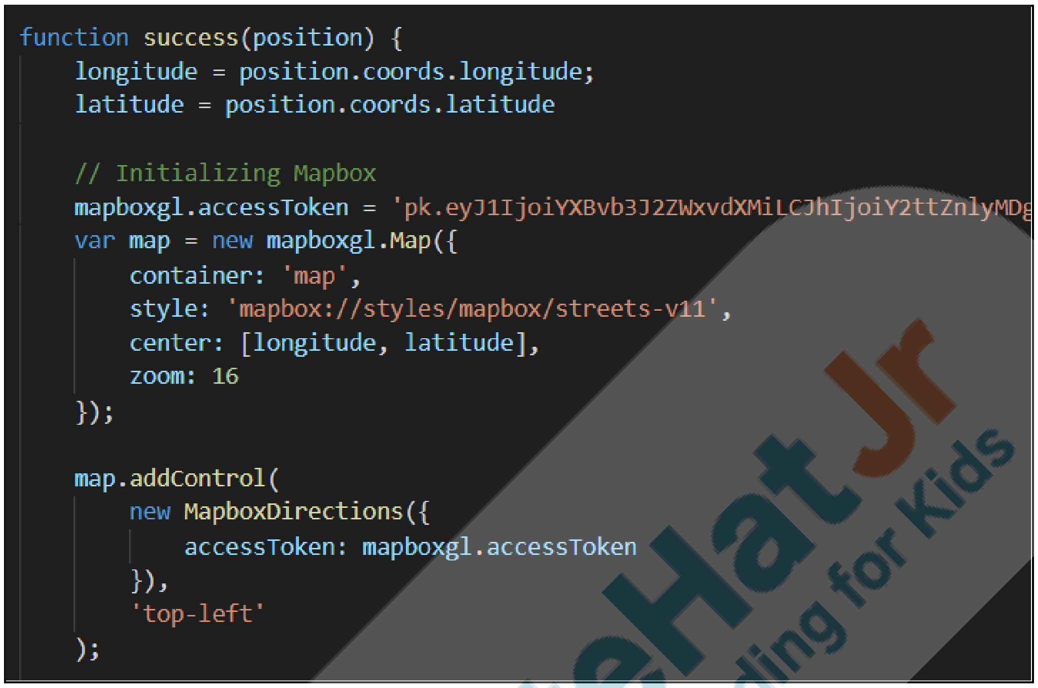


```
function success(position) {
    longitude = position.coords.longitude;
    latitude = position.coords.latitude
```

Next, we will give values to our latitude and longitude from the ***position*** argument in our ***success()*** function.

We will then move all our code for mapbox inside this ***success()*** function. Our success function will look like -

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.



```
function success(position) {
    longitude = position.coords.longitude;
    latitude = position.coords.latitude

    // Initializing Mapbox
    mapboxgl.accessToken = 'pk.eyJ1IjoiYXBvb3J2ZWxvdXMiLCJhIjoiY2ttZnlyMDg
    var map = new mapboxgl.Map({
        container: 'map',
        style: 'mapbox://styles/mapbox/streets-v11',
        center: [longitude, latitude],
        zoom: 16
    });

    map.addControl(
        new MapboxDirections({
            accessToken: mapboxgl.accessToken
        }),
        'top-left'
    );
```

	Now, the latitude and longitude that we have fetched based on the user's current location can be considered as the starting point. We need to have the destination's latitude and longitude as well, so that we can use the direction's API and navigate our user. For that, let's use some jQuery and save the latitude and longitude for the destination.	

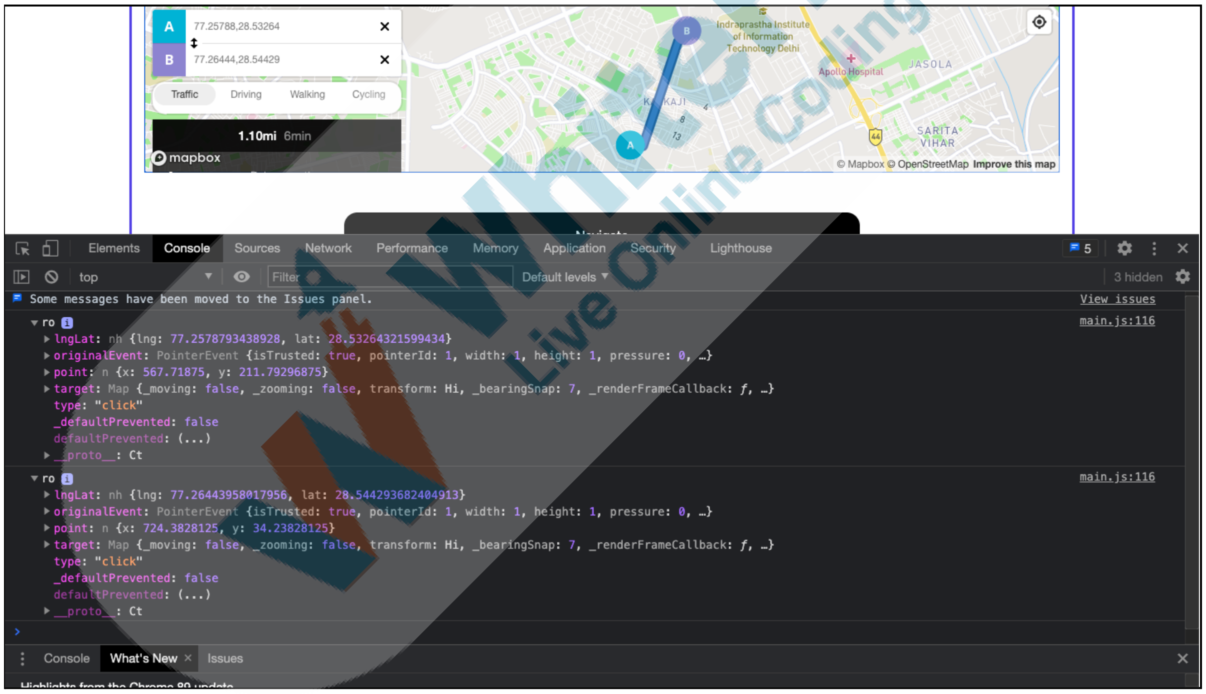
© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

Inside our ***success()*** function, let's add an **on click** function first and see the things we are getting out of it -

```
map.on('click', function (e) {
    console.log(e)
});
```

This is another way of how we can write our jQuery functions for various events, such as **click, change, etc.**

Now here, on the click event of the map, we are creating a function with argument **e**. This **e** will contain information about the event that happened. Let's reload our page and select something for our starting and ending location and see what this prints.



```
Some messages have been moved to the Issues panel.
▼ ro                                                                              main.js:116
  ▶ lngLat: nh {lng: 77.2578793438928, lat: 28.53264321599434}
  ▶ originalEvent: PointerEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure: 0, …}
  ▶ point: n {x: 567.71875, y: 211.79296875}
  ▶ target: Map {_moving: false, _zooming: false, transform: Hi, _bearingSnap: 7, _renderFrameCallback: f, …}
    type: "click"
    _defaultPrevented: false
    defaultPrevented: (...)
  ▶ __proto__: Ct
▼ ro                                                                              main.js:116
  ▶ lngLat: nh {lng: 77.26443958017956, lat: 28.544293682404913}
  ▶ originalEvent: PointerEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure: 0, …}
  ▶ point: n {x: 724.3828125, y: 34.23828125}
  ▶ target: Map {_moving: false, _zooming: false, transform: Hi, _bearingSnap: 7, _renderFrameCallback: f, …}
    type: "click"
    _defaultPrevented: false
    defaultPrevented: (...)
  ▶ __proto__: Ct
```

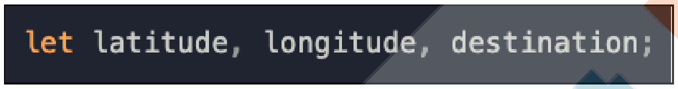
Here, we can see the ***console.log()*** happening 2 times. The first time when we select the **A point (*starting location*)** and the second time when we select the **B point (*destination location*).**

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

We can also see a **lngLat** key which contains the latitude and the longitude of the points we have clicked on.

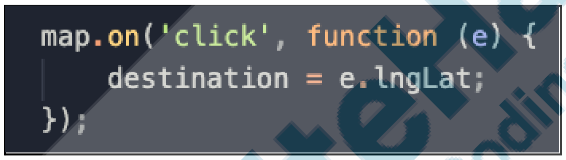
Now assuming that the user will *always* select the starting point first and the destination point second, we can create a variable ***destination*** in which we can save the latitude and the longitude both the times, and we will receive our destination's latitude and longitude.

Let's do that.



```
let latitude, longitude, destination;
```

We first create our ***destination*** variable, along with ***latitude and longitude*** and then -



```
map.on('click', function (e) {
    destination = e.lngLat;
});
```

We are saving the **lngLat** from our **e** (event) variable in the destination!

	We are now mostly done with this screen. The only thing that's left is, what happens when the user clicks on the ***Navigate*** button. We discussed earlier that we will take the user to a different page where we will navigate them, and we somehow need to send their starting coordinates and destination coordinates to this new page. Do you have any ideas on how we can do that?	**ESR:** Varied.

WhiteHat Jr Live online Coding for Kids

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

	We can send the data through the URL. Now if you remember, we have already done this in the previous module. These are known as the query parameters. Let's take a look at a sample URL.	

www.example.com/index?arg1=1&arg2=2&arg3=3

Here, anything before the question mark - (?) - is the URL and anything after that are the query parameters.

Question Mark (?) separates the URL from the query parameters.

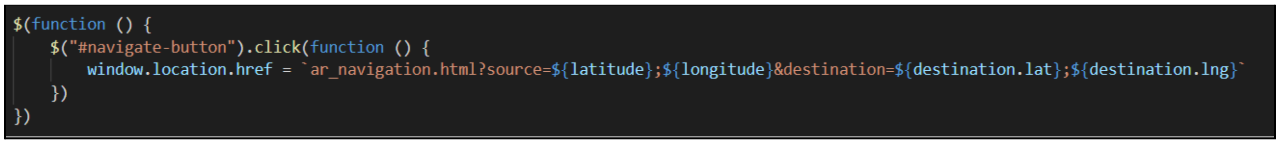
Now the parameters are separated by an ampersand - (&) - so in this example, we have 3 query parameters:

1. arg1=1
2. arg2=2
3. arg3=3

Now it's easy to guess the rest of it. **arg1, arg2 and arg3** are the variables whose respective values are **1, 2 and 3**.

	Following this format, we can pass our starting coordinates and ending coordinates to the next page. We will have to create a **click()** event function on our ***navigate*** button to do that.	

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.



```
$(function () {
    $("#navigate-button").click(function () {
        window.location.href = `ar_navigation.html?source=${latitude};${longitude}&destination=${destination.lat};${destination.lng}`
    })
})
```

Here we have a **$(function())** inside which, we take the **id** of our navigation button - **#navigate-button** and create a **click()** function for it.

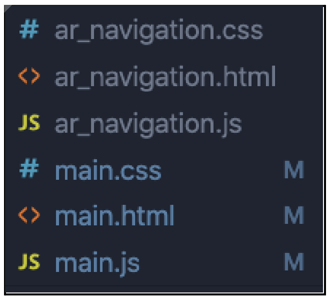
Inside this function, we have **window.location.href**, a method in JavaScript that takes a URL to which you want to navigate.

As a value for this method, we give "ar_navigation.html" and our query parameters here are "source" & "destination".

Both these variables have values in the format of ***latitude;longitude*** where the latitude and longitude are separated by a semicolon(;).

	Now we know how to add query parameters in the URL. You will be writing a function to get values of the query parameters from the URL.	
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
Teacher Starts Slideshow **Slide 12 to 14** 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.		

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

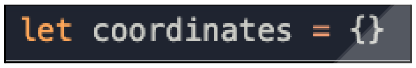
Teacher Ends Slideshow		
STUDENT-LED ACTIVITY - 20 mins		
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start screen share.** • **Teacher gets into fullscreen.**		
ACTIVITY • **Write a function to get latitude and longitude from the query parameters.**		
Step 3: Student-Led Activity (20 mins)	*The teacher guides the student to clone the code from Student Activity 1.* ***[Student Activity 1]*** ***Note**: The student will continue to add new functionality after teacher activity.*	
	Now we will need ***ar_navigation.html*** for the next screen and the corresponding files for it - ***ar_navigation.js*** *and* ***ar_navigation.css****.* ***Note**: We have added our HTML file and our CSS file already.*	
		

```
# ar_navigation.css
<> ar_navigation.html
JS ar_navigation.js
# main.css        M
<> main.html      M
JS main.js        M
```

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

WhiteHat Jr
Live online Coding for Kids

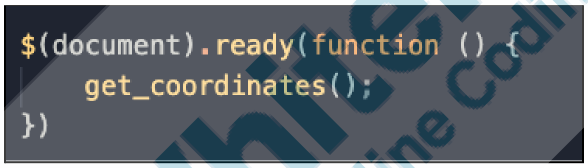
	In the JavaScript file, the first thing that we need to do is to fetch the query parameter. Let's see how to do that now(in **ar_navigation.js** file).	



```
let coordinates = {}
```

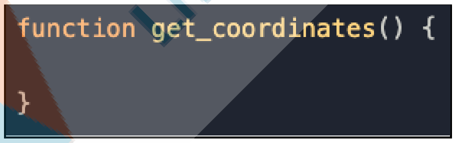
We first create an object called ***coordinates*** in which we will be saving the coordinates for our starting point and destination.

Next, we will need to create a **$(document).ready()** function where we will call a function that can fetch the coordinates from the query parameters for us.



```
$(document).ready(function () {
    get_coordinates();
})
```

And now, we will create our **get_coordinates()** function to get the data from the query parameters.

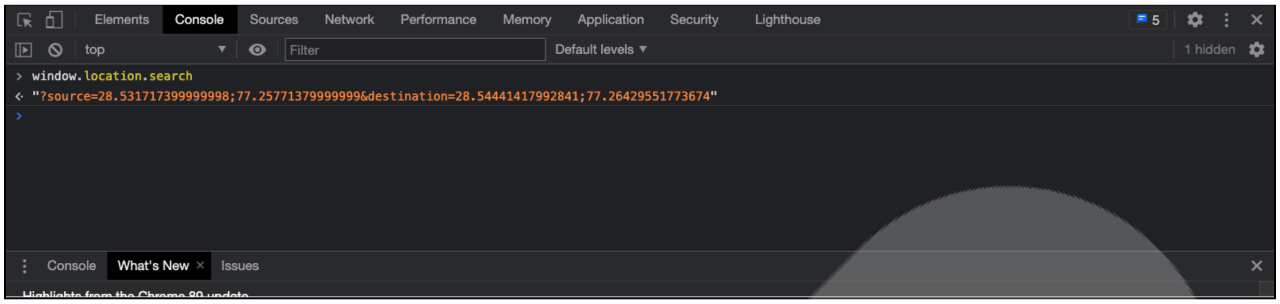


```
function get_coordinates() {

}
```

Now, there's a method that gets the query parameters for us in JavaScript. It's called ***URLSearchParams()*** that takes the query parameters and creates an object for the key-value pairs from it.

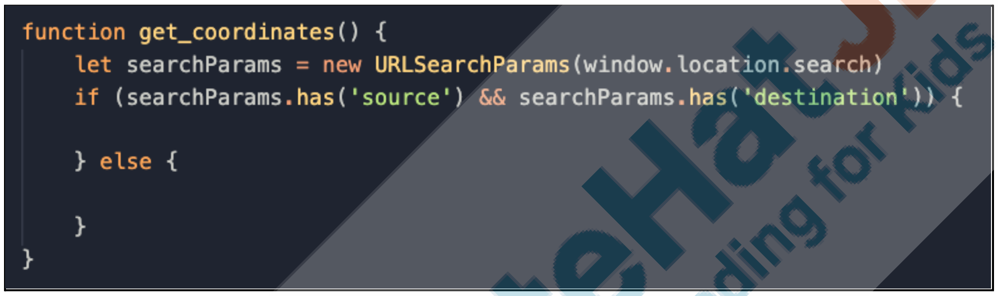
We can get our query parameters from ***window.location.search***. This gives us the following.

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.



```
> window.location.search
<· "?source=28.531717399999998;77.25771379999999&destination=28.54441417992841;77.26429551773674"
>
```

So our code would look like:

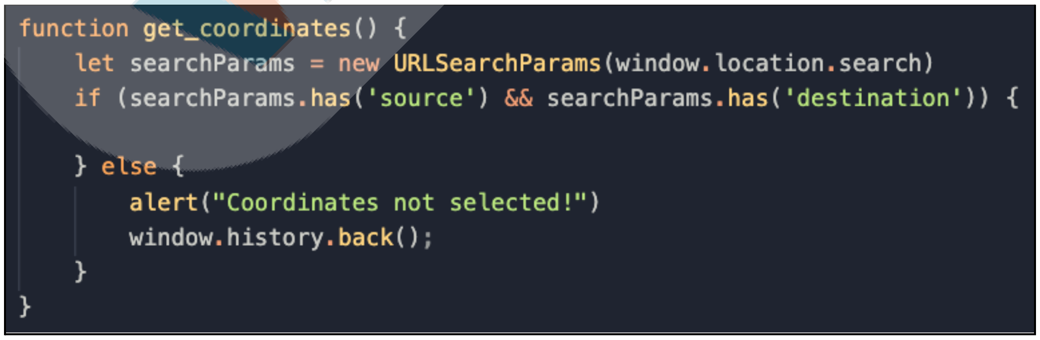


```
function get_coordinates() {
    let searchParams = new URLSearchParams(window.location.search)
    if (searchParams.has('source') && searchParams.has('destination')) {

    } else {

    }
}
```

Here we are creating a new **URLSearchParams** object and passing our query parameters (through ***window.location.search***) in it. This gives us a usable ***searchParams*** object.

Once we have it, we can check if it has a particular query parameter using the ***has()*** function. We are using an if else statement to check if both the ***source && destination*** are there or not.

If not, we have to alert the user and send them back to the previous page:



```
function get_coordinates() {
    let searchParams = new URLSearchParams(window.location.search)
    if (searchParams.has('source') && searchParams.has('destination')) {

    } else {
        alert("Coordinates not selected!")
        window.history.back();
    }
}
```

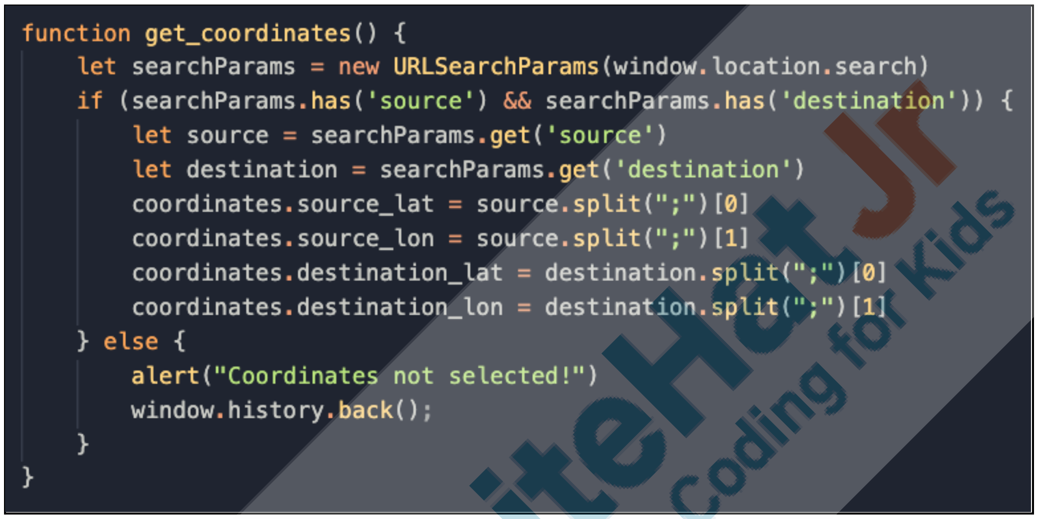
window.history.back in JavaScript can be used to go back to the previous page.

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

WhiteHat Jr
Live online Coding for Kids

Now we need to fetch the data from the query parameters and save it in our ***coordinates*** object.

For that, we can use the ***get()*** function on our ***searchParams*** object:



```
function get_coordinates() {
    let searchParams = new URLSearchParams(window.location.search)
    if (searchParams.has('source') && searchParams.has('destination')) {
        let source = searchParams.get('source')
        let destination = searchParams.get('destination')
        coordinates.source_lat = source.split(";")[0]
        coordinates.source_lon = source.split(";")[1]
        coordinates.destination_lat = destination.split(";")[0]
        coordinates.destination_lon = destination.split(";")[1]
    } else {
        alert("Coordinates not selected!")
        window.history.back();
    }
}
```

Here, we are first getting the ***source*** and ***destination*** parameters, and then with the help of the ***split()*** function, we are saving the ***source_lat, source_lon, destination_lat and destination_lon*** values in our ***coordinates*** object.

This is how we formatted our ***source*** and ***destination*** in ***latitude;longitude*** format in the URL.

Therefore we are splitting our parameter values with a semicolon - (;) - and we are taking the **first element [0] as latitude** and **second element [1] as longitude.**

	With this, we have successfully fetched our source and destination coordinates from the query parameters.	

Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 5 mins

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

Teacher Starts Slideshow **Slide 15 to 19**
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 20 to 25**
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:*

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

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.*

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

Activity	Activity Name	Links
Teacher Activity 1	Boilerplate Code	https://github.com/whitehatjr/PRO-C178-Code-Ref
Teacher Activity 2	Navigator Object	https://developer.mozilla.org/en-US/docs/Web/API/Navigator/geolocation
Teacher Activity 3	Teacher Reference Code	https://github.com/whitehatjr/PRO-C179-Code-Ref
Student Activity 1	Boilerplate Code	https://github.com/whitehatjr/PRO-C179-Student-Boilerplate
Student Activity 2	Navigator Object	https://developer.mozilla.org/en-US/docs/Web/API/Navigator/geolocation
Teacher Reference 1	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/796deb57-70b0-4ce7-84ca-cd27956ab3d3.pdf
Teacher Reference 2	Project Solution	https://github.com/whitehatjr/PRO-C179-Project-Sloution
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/2d8b9a43-640c-4915-86ae-16242f324e5e.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/b8a8d330-eb80-4f33-a01b-f7ba4d31dc33.pdf

© 2020 - WhiteHat Education Technology Private Limited.
Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.