Exercise

Customize the Code in a Basic Geo App

Section 6 Exercise 1

April 8, 2021

Customize the Code in a Basic Geo App

Instructions

Use this guide and ArcGIS Online to reproduce the results of the exercise on your own.

Note: ArcGIS Online is a dynamic mapping system. The version that you will be using for this course may be slightly different from the screen shots you see in the course materials.

Time to complete

Approximately 40-50 minutes

Technical note

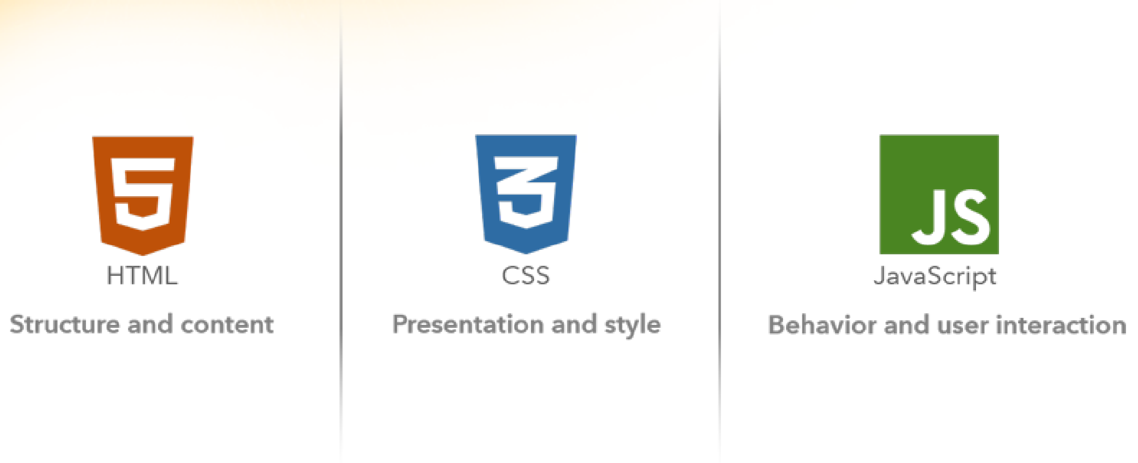
To take advantage of the web-based technologies available in ArcGIS Online, use the latest version of Google Chrome, Mozilla Firefox, Apple Safari, or Microsoft Edge. Other browsers may not display your maps and apps correctly.

Introduction

As you have seen throughout this course, Esri provides many geo app templates that do not require coding. Most of this course has focused on these types of apps, including various instant app templates and app builders. These apps provide great functionality, but they may not always meet all of your project requirements. In such cases, knowing a little bit of coding can be very helpful. You can customize an existing app or create a new web app from scratch.

One way to use code is to use an API, or application programming interface. The ArcGIS API for JavaScript provides libraries to build dynamic web apps for visualization, analysis, and feature editing in 2D and 3D.

JavaScript is a popular programming language, commonly used in conjunction with HTML and CSS to create web apps. HTML defines the content of web pages, CSS defines their layout and style, and JavaScript defines their behavior. For example, HTML can be used to create a button on a website, CSS can define the button's appearance, and JavaScript can control what happens when you click the button.



HTML, CSS, and JavaScript are languages with unique purposes in web page creation.

We won't get into a lot of detail about HTML, CSS, and JavaScript in this exercise, but there are many resources available for those of you who are new to web development. Look at the Learning Resources section at the end of the exercise if you would like to explore these topics more.

To introduce you to custom app development and the ArcGIS API for JavaScript, you will investigate a sample web app and see the underlying code. Then, you will add some simple HTML and JavaScript to change the displayed elements and behavior of the web page.

Part I - Guided

Remember that the exercises in Sections 2 through 6 have two parts: Guided, with step-by-step instructions; and Do-It-Yourself (DIY), in which you can explore further and build your own geo apps.

In the Guided part of this exercise, you will find a sample app and use it as the basis for your exploration. The DIY part contains optional stretch goals to apply what you have learned.

Step 1: Find a sample app

You will start by navigating the ArcGIS Developers site. You will locate an ArcGIS API for JavaScript code sample that you will later customize.

IMPORTANT

Make sure that you have completed step 1 of *Section 1 Exercise 1: Find Amenities in Denver, CO*. You'll need to use your provided course ArcGIS credentials to complete all of the exercises in this course.

a Open a new private (or incognito) web browser tab or window and go to developers.arcgis.com.

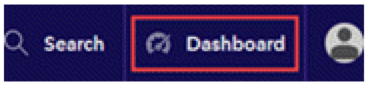
We recommend that you open a private or incognito browser window to help prevent conflicts with your accounts.

b If necessary, sign in with your course credentials.

Note: Step 1 of the Section 1 Exercise 1 PDF explains how to determine your ArcGIS credentials (user name and password) for this course. If you have trouble signing in, go to the Help tab in the MOOC platform.

c If necessary, click Return To Dashboard to close the Welcome message.

d If signing in does not take you to your dashboard page, at the top, click Dashboard.



e Review the Dashboard page.

As you scroll down, you will see that you can access your ArcGIS Online items, create new items, or develop apps using your items.

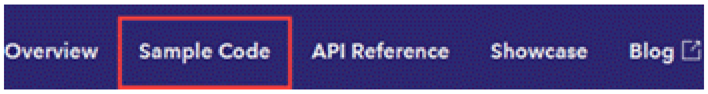
f At the top left, click Documentation to go to the home page for developer guides.

g Review the links to the various APIs.

h Under Web APIs, click the first icon, for ArcGIS API For JavaScript.

This support page for the JavaScript API has current information, technical support, videos, Twitter accounts to follow, the latest Esri Community discussions, and more.

i At the top right of the page, click the Sample Code tab.



On the left side of the page, you will see a list of hundreds of code samples organized by topic.

j Expand the Mapping And Views section.

k Click Load A Basic Web Map.

l Explore the map that loads.

This app contains a live web map of the United States that shows accidental deaths by state. You can pan and zoom the map and click on features to see their attributes.

m Below the map, click Explore In The Sandbox.

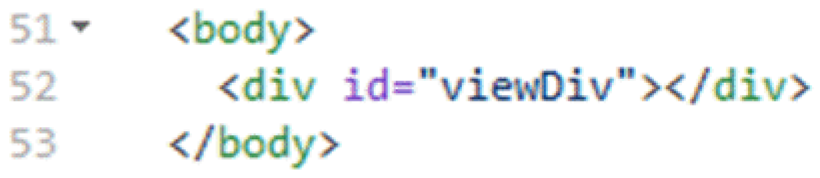
You will see the HTML, CSS, and JavaScript code in the editor pane on the left and the preview on the right. You can use the sandbox to experiment with code edits, and then click Refresh to see the changes in the app. Although you have a choice in how to structure your code, all the API code samples have a similar code structure. This helps you apply what you are about to learn to other code samples. This app preview displays a basic web map.

Step 2: Explore an HTML document

a Toward the bottom of the code, find the <body> section, which contains a <div> component.

Hint: Scroll down in the editor pane on the left to find the <body> section.

An HTML div is a division, or section, that acts as a container. In this case, it contains the map component.

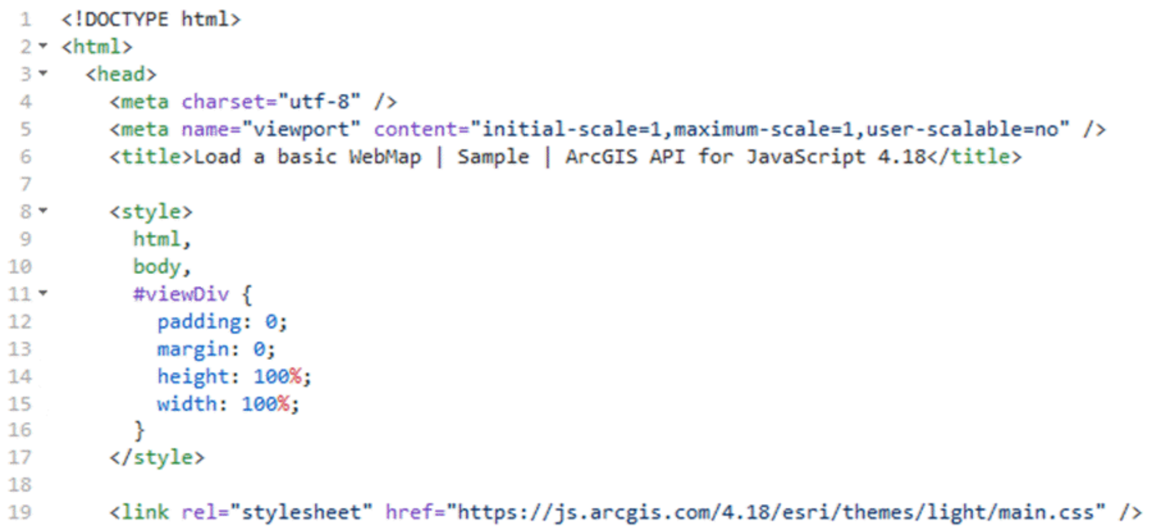


```
<body>
  <div id="viewDiv"></div>
</body>
```

Both <body> and <div> are HTML tags. Tags are keywords surrounded by angle brackets, like this: <keyword>. Tags tell the browser what the content within the tags is and, in some cases (like with <body> and <div> tags), how to display content. In this code, the <div> tag and its attribute, id, are telling the browser to display the web map.

b At the top of the code, examine the following components:

1. Metadata tags, which provide supplemental information for the web page and help with web search
2. Title of the web page
3. Style tags for internal CSS, which provide style information for elements of the web page
4. Links to external CSS stylesheets, or documents, which are another way to style page elements

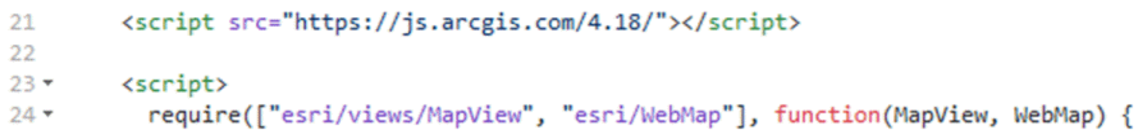


```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="initial-scale=1,maximum-scale=1,user-scalable=no" />
    <title>Load a basic WebMap | Sample | ArcGIS API for JavaScript 4.18</title>

    <style>
      html,
      body,
      #viewDiv {
        padding: 0;
        margin: 0;
        height: 100%;
        width: 100%;
      }
    </style>

    <link rel="stylesheet" href="https://js.arcgis.com/4.18/esri/themes/light/main.css" />
```

Note: This page uses a common pattern for creating and organizing HTML and CSS. There are other methods you can implement.

This exercise will not cover CSS in depth, but if you are interested in learning more, you can start here (https://bit.ly/1ofYbbT).

c After the external CSS links, examine the two <script> tags in the document.



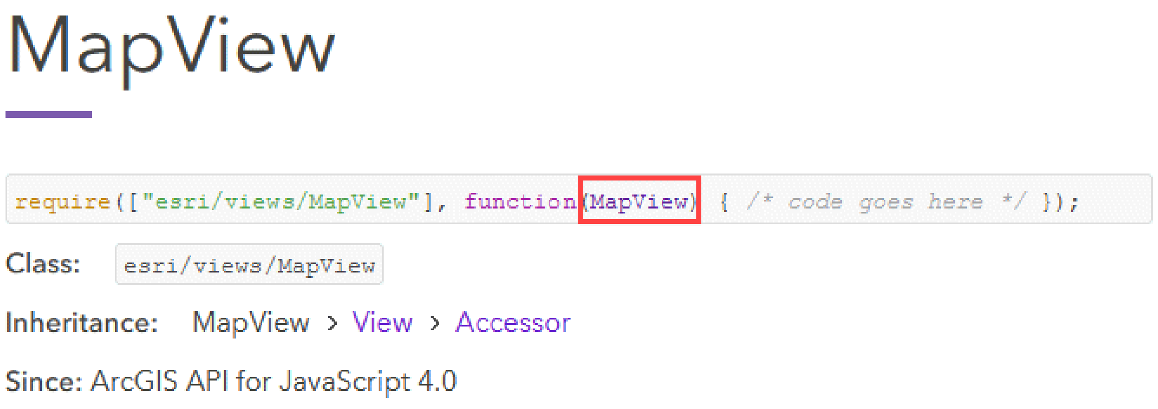
```html
    <script src="https://js.arcgis.com/4.18/"></script>

    <script>
      require(["esri/views/MapView", "esri/WebMap"], function(MapView, WebMap) {
```

The first script tag provides a reference to the location on the web where the API can be accessed. This reference is required for you to use the API code along with your own code on your HTML page. The src attribute in the script tag is the URL where the API code is accessed.

The second script tag lists two specific modules. Each module provides a single class that gives a specific set of functionality to develop applications. For example, the "esri/views/MapView" module provides the MapView class, which allows you to render web maps from ArcGIS Online in 2D. You can find a quick tutorial on using the MapView class here (https://bit.ly/2nLgAG4).

Documentation on all of the classes in the API for JavaScript can be found in the API Reference (https://bit.ly/2mY1FLl).

Hint: Most modules have an alias, or common name, that allows you to quickly access the module without typing the full path. For example, you can type **MapView** *in your code rather than* **esri/views/MapView**. *You can add any alias in the function, but using the preferred aliases is a best practice. You can find the preferred alias for a module from the module's entry in the API Reference documentation. In the following graphic, the preferred alias for the MapView module is outlined in red.*



MapView

```
require(["esri/views/MapView"], function(MapView) { /* code goes here */ });
```

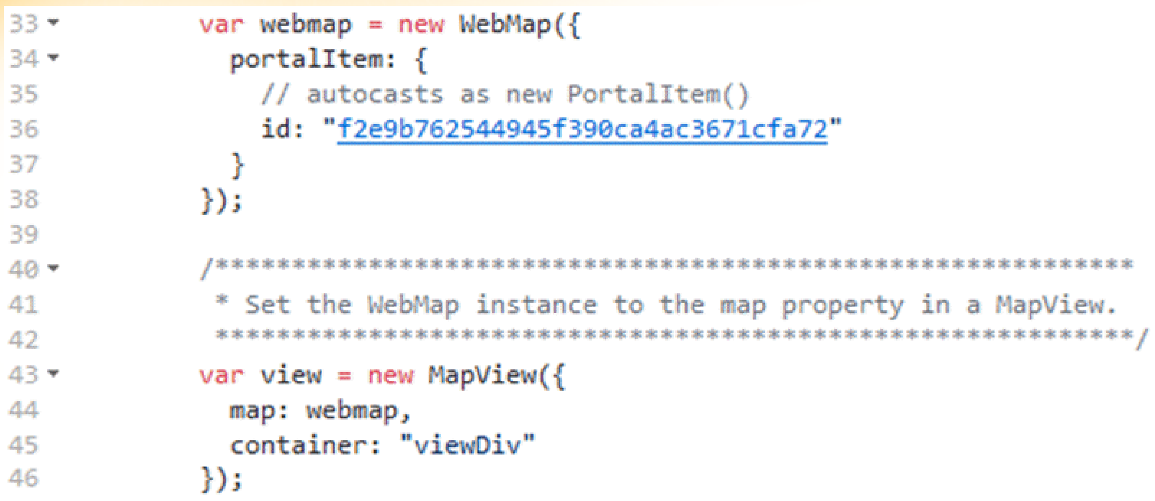
Class: `esri/views/MapView`

Inheritance: MapView > View > Accessor

Since: ArcGIS API for JavaScript 4.0

Note: You do not need to open the documentation. The screen shot is just for illustration.

d In the sandbox, investigate the code in the following graphic:



```
33          var webmap = new WebMap({
34            portalItem: {
35              // autocasts as new PortalItem()
36              id: "f2e9b762544945f390ca4ac3671cfa72"
37            }
38          });
39
40          /*************************************************************
41           * Set the WebMap instance to the map property in a MapView.
42           *************************************************************/
43          var view = new MapView({
44            map: webmap,
45            container: "viewDiv"
46          });
```

This code creates a WebMap instance from the ID of an existing web map in ArcGIS Online and displays that web map. Does anything look familiar? The code uses the module aliases MapView and WebMap, which you saw earlier in the function on line 24 in Step 2c.

For example, the code on lines 43-46 creates a variable, named view, which uses the class MapView:

```
var view = new MapView({
  map: webmap,
  container: "viewDiv"
});
```

If "esri/views/MapView" and the alias MapView above had not been loaded, the map would not display.

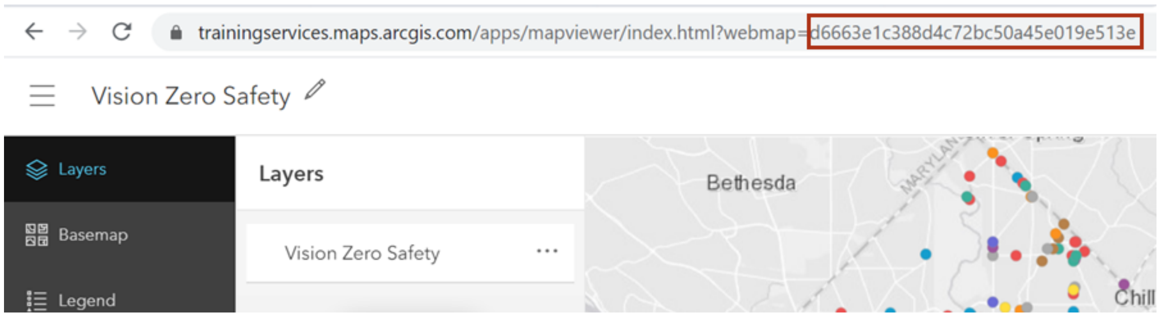
Again, you do not need to understand everything about the code. Just know that you need to load modules from the ArcGIS API for JavaScript to access that code and do things with it in your app. See the Learning Resources section for references.

Now, let's make some changes to the code to see some of the things it can do. First, you will change the web map in your app, using the web map ID.

Step 3: Change the web map in your app

You will make changes to the code in the sample web app, using the sandbox, to see how they affect what is displayed. First, you will look at the web map in the app, which is referred to by an item ID.

The web map ID is a unique set of characters stored in ArcGIS Online that represents the web map. When you are looking at a web map in ArcGIS Online, you can see the web map ID in the URL, as shown in the following graphic. Whereas you use the name of the map to identify it as the "Vision Zero Safety" web map, ArcGIS Online uses the web map ID to identify it as the "d6663e1c388d4c72bc50a45e019e513e" web map.

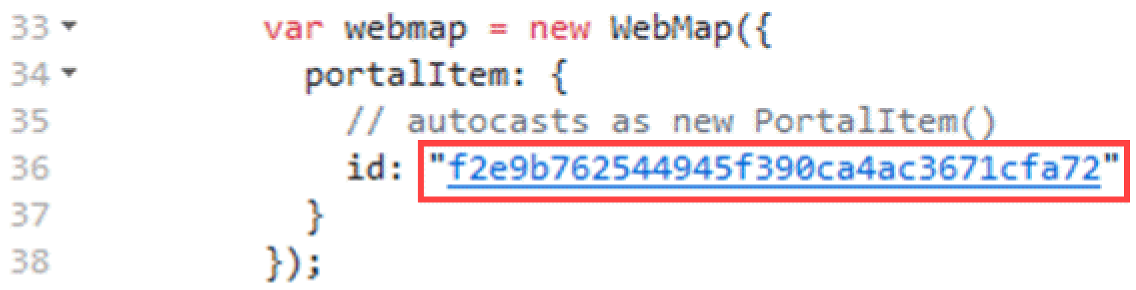


trainingservices.maps.arcgis.com/apps/mapviewer/index.html?webmap=d6663e1c388d4c72bc50a45e019e513e

You will use the sandbox to change the map in the Load A Basic Web Map app—from the Accidental Deaths map to the Vision Zero Safety map—by changing the web map ID.

a In the editor pane, go to the line containing the web map ID.

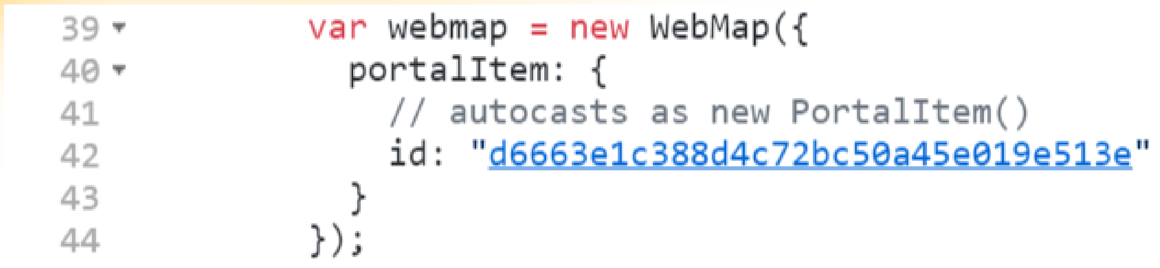
Note: The web map ID is a unique identifier that contains numbers and letters. Depending on your line spacing, the code for the web map ID may appear on a different line than shown in the following graphic.



```
      var webmap = new WebMap({
        portalItem: {
          // autocasts as new PortalItem()
          id: "f2e9b762544945f390ca4ac3671cfa72"
        }
      });
```

b Delete "f2e9b762544945f390ca4ac3671cfa72" and replace it with **"d6663e1c388d4c72bc50a45e019e513e"**, as shown in the following graphic.

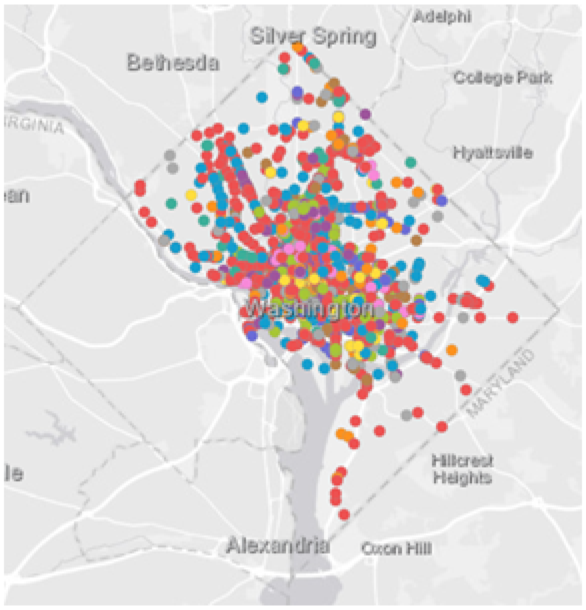
Note: Be sure to include the quotation marks when you type in the ID.



```
        var webmap = new WebMap({
          portalItem: {
            // autocasts as new PortalItem()
            id: "d6663e1c388d4c72bc50a45e019e513e"
          }
        });
```

c At the top right of the page, click Refresh to see your preview update.

You have now added a new web map to your app. The preview on your screen should resemble the following graphic.



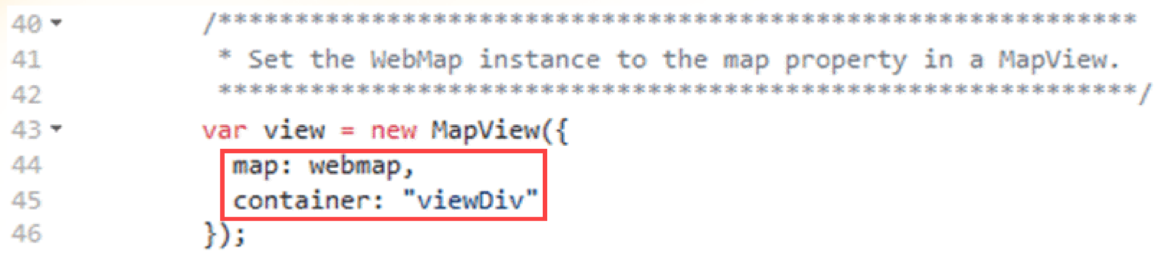
Assigning a different web map ID property for the WebMap instance changed the web map being rendered in the app.

Step 4: Explore the Search widget class

You saw examples of using classes to add functionality to your app with the loading of the WebMap and MapView modules. Each module corresponds to a class of the same name, and the class provides a group of properties. Properties can be set to define the behavior for an instance of the class. One instance can also be referred to as an object.

In Step 3, when you changed the ID of the web map, that code created an object that was of the WebMap class. The ID that controlled which web map was displayed in the app was one of the properties. You updated that property by passing in the ID of an existing web map.

a Look at the MapView code, discussed in Step 2.

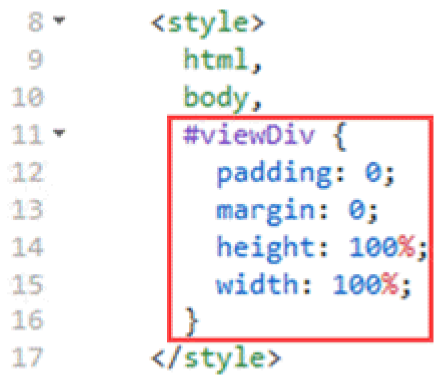


```
40          /************************************************************
41           * Set the WebMap instance to the map property in a MapView.
42           ************************************************************/
43          var view = new MapView({
44            map: webmap,
45            container: "viewDiv"
46          });
```

The code uses the MapView class to display a web map in 2D. Both a map property and a container property are passed to the view object (the MapView).

Question:
Which values are passed in to those two properties?

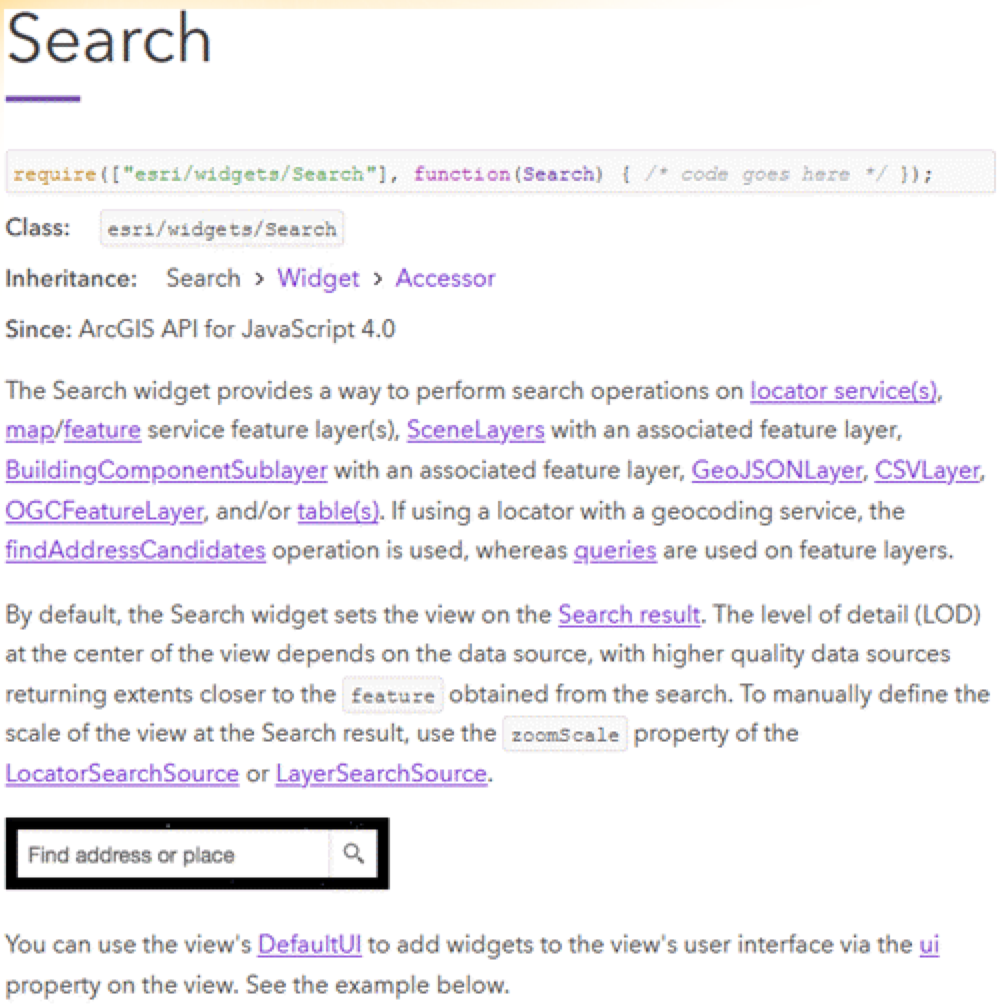
Answer:
--On line 44, webmap is the object of type WebMap, created on line 36.
--On line 45, viewDiv is an ID selector that was defined on line 11, which basically says that the MapView will render in the entire browser space (height: 100%, width: 100%).



```
 8      <style>
 9        html,
10        body,
11        #viewDiv {
12          padding: 0;
13          margin: 0;
14          height: 100%;
15          width: 100%;
16        }
17      </style>
```

Now you will take a look at another class, Search.

b Open a new browser window or tab, and go to the documentation for the Search widget (https://bit.ly/2MYihuL) in the API Reference.



Search

```
require(["esri/widgets/Search"], function(Search) { /* code goes here */ });
```

Class: `esri/widgets/Search`

Inheritance: Search > Widget > Accessor

Since: ArcGIS API for JavaScript 4.0

The Search widget provides a way to perform search operations on locator service(s), map/feature service feature layer(s), SceneLayers with an associated feature layer, BuildingComponentSublayer with an associated feature layer, GeoJSONLayer, CSVLayer, OGCFeatureLayer, and/or table(s). If using a locator with a geocoding service, the findAddressCandidates operation is used, whereas queries are used on feature layers.

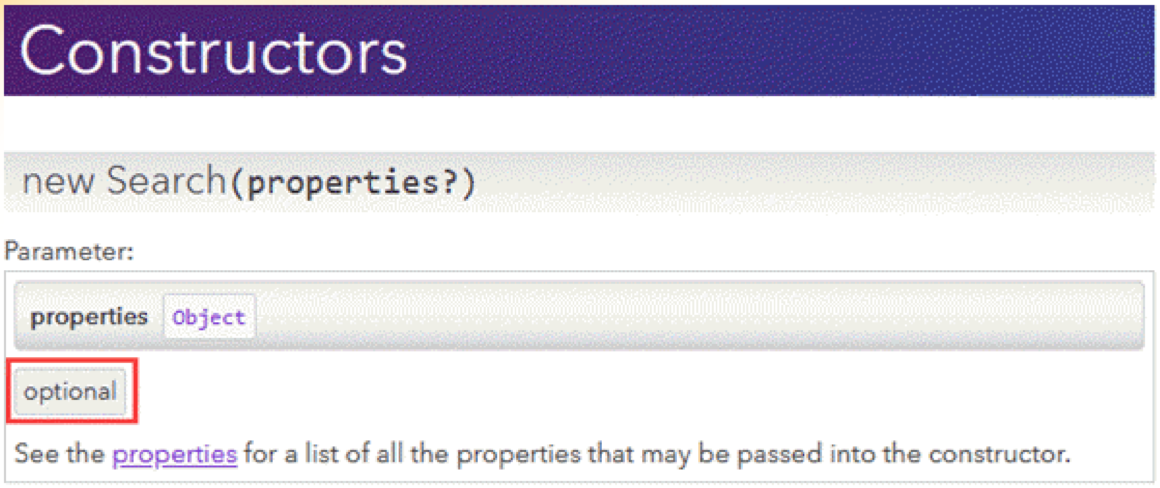
By default, the Search widget sets the view on the Search result. The level of detail (LOD) at the center of the view depends on the data source, with higher quality data sources returning extents closer to the `feature` obtained from the search. To manually define the scale of the view at the Search result, use the `zoomScale` property of the LocatorSearchSource or LayerSearchSource.

You can use the view's DefaultUI to add widgets to the view's user interface via the ui property on the view. See the example below.

The Search widget would allow a user of the web app to search for an address or place by using a locator service like Esri's ArcGIS World Geocoding Service (https://bit.ly/1MsqCAr) (the default).

c Scroll down to the Constructors section for the Search class.

A constructor is used to create an object and prepare the object for use, often accepting parameters. You can see that the Search class constructor takes one optional parameter that describes the properties for the Search widget. This parameter, if you decide to provide it, would be an Object specifying one or more of the properties found further down on the API reference page.



Constructors

new Search(properties?)

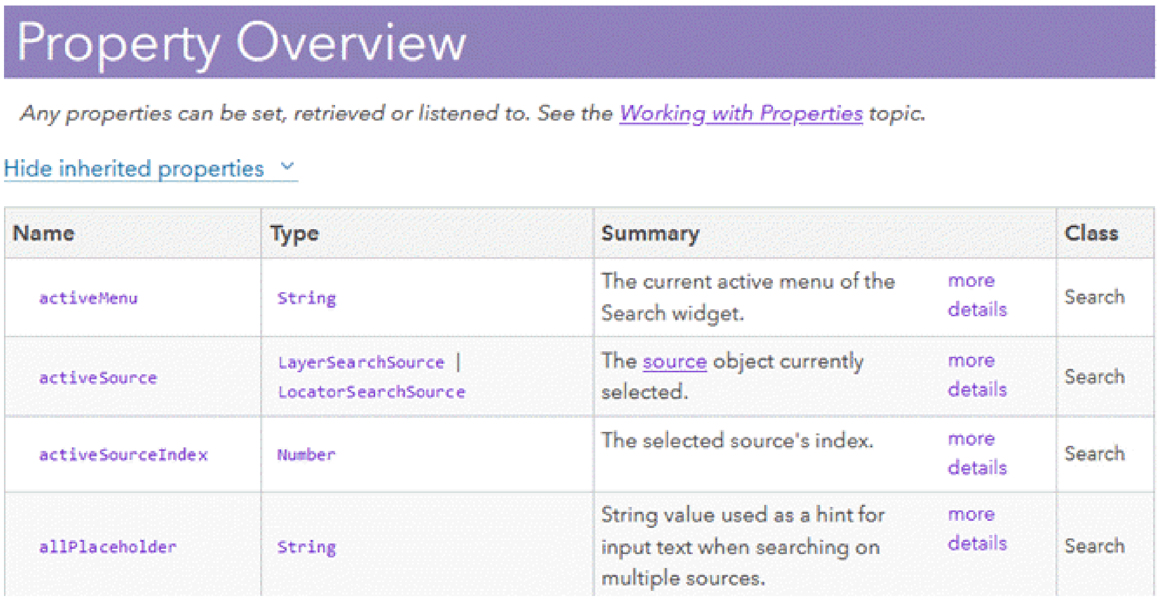
Parameter:

properties Object

optional

See the properties for a list of all the properties that may be passed into the constructor.

d In the API documentation below the Constructors section for every class, scroll down to the Property Overview.



Property Overview

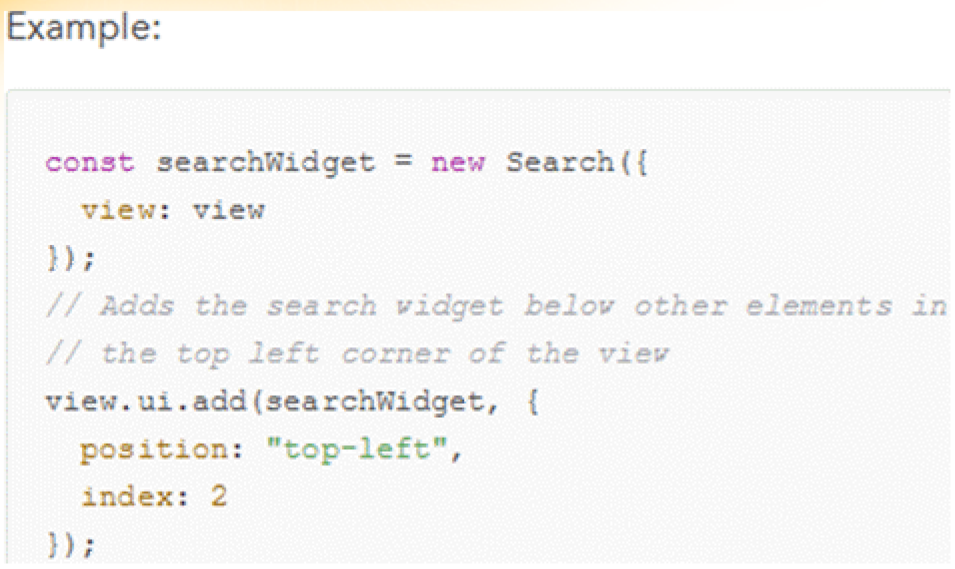
Any properties can be set, retrieved or listened to. See the Working with Properties topic.

Hide inherited properties

Name	Type	Summary		Class
activeMenu	String	The current active menu of the Search widget.	more details	Search
activeSource	LayerSearchSource | LocatorSearchSource	The source object currently selected.	more details	Search
activeSourceIndex	Number	The selected source's index.	more details	Search
allPlaceholder	String	String value used as a hint for input text when searching on multiple sources.	more details	Search

This section lists all the properties that can possibly be passed to the class referenced. If you keep scrolling, you see the Property Details, with more information and links to other resources for the properties listed in the Property Overview. Rather than go through all of these, let's take a look at one from the Example code.

e Scroll up, above the Constructors section, to the Example code.


Example:

```
const searchWidget = new Search({
  view: view
});
// Adds the search widget below other elements in
// the top left corner of the view
view.ui.add(searchWidget, {
  position: "top-left",
  index: 2
});
```

Here, searchWidget is only being passed one property, view, which associates the Search widget with a specific view. If you want more information about the view property, you can scroll down to find it in the Search class properties.

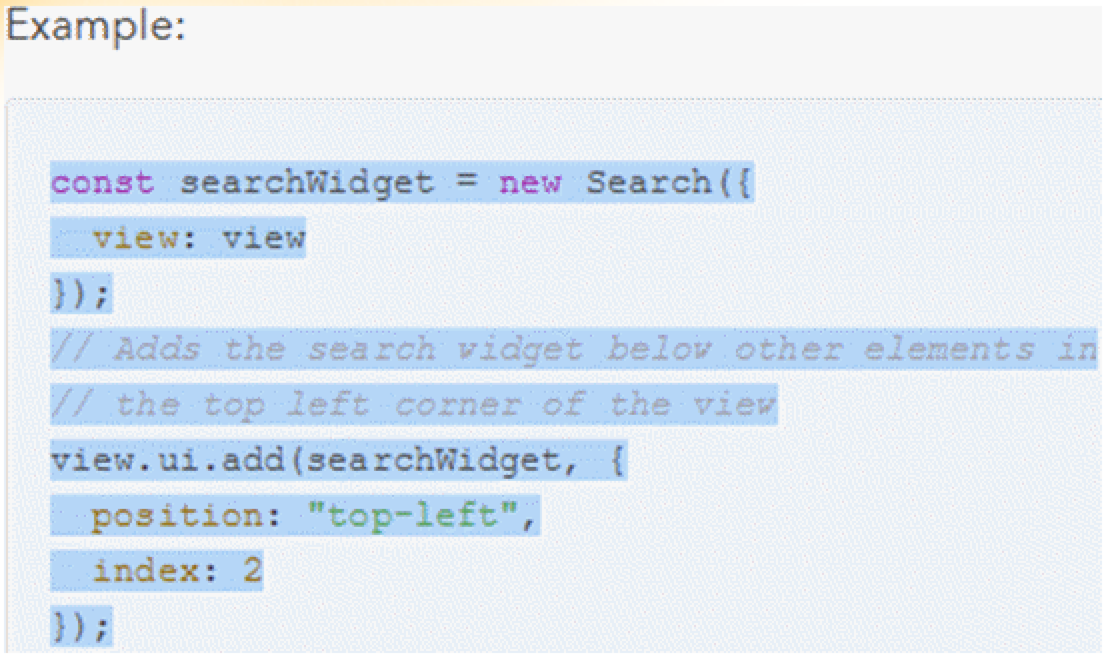
We don't need to dig any deeper into the API reference for this exercise. Just know that there is a lot of information there, and with a little patience and experimentation, you can learn to create some pretty amazing custom geo apps. Documentation is a developer's best friend!

Next, you will return to your code and add the Search widget to the Load A Basic WebMap sample app. You will use the example code from the API Reference documentation.

Step 5: Add the Search widget to the Basic WebMap sample

You will copy and paste the code that you need from the example section.

a In the Search page's Example code section, copy all of the code.



Example:

```
const searchWidget = new Search({
  view: view
});
// Adds the search widget below other elements in
// the top left corner of the view
view.ui.add(searchWidget, {
  position: "top-left",
  index: 2
});
```

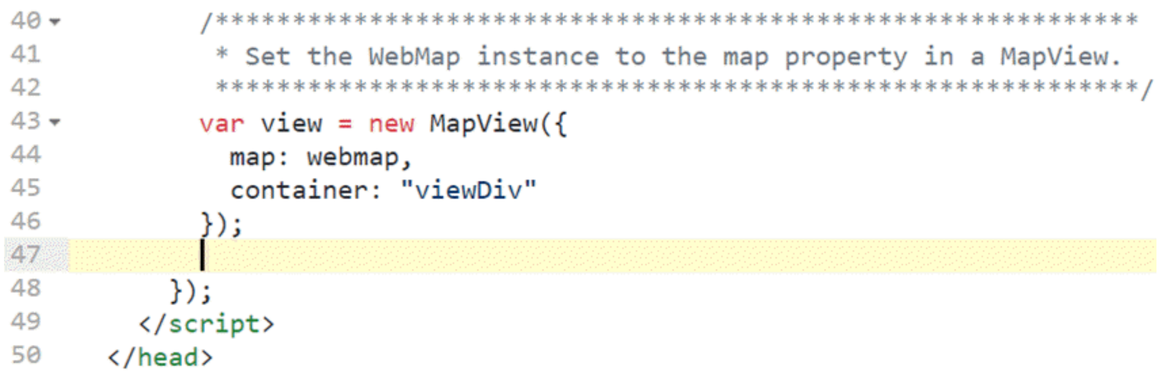
b Return to the Load A Basic WebMap sample sandbox.

Hint: If you no longer have this sandbox open, refer to the instructions in Step 1.

Note: Before continuing, make sure that you have switched the web map IDs so that the Washington D.C. Vision Zero Safety layer is displayed in the app preview, as described in Step 3.

c In the editor pane of the sandbox, on line 46 (after the view was created), click to the right of the semicolon.

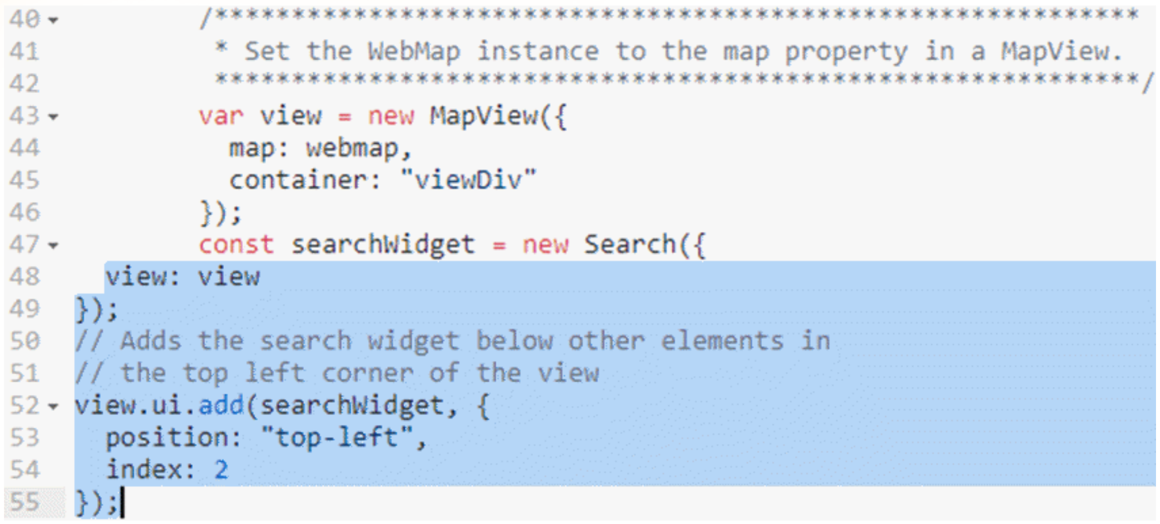
d Press Enter (Return on Apple computers) to create a new line.



```
40        /************************************************************
41         * Set the WebMap instance to the map property in a MapView.
42         ************************************************************/
43        var view = new MapView({
44          map: webmap,
45          container: "viewDiv"
46        });
47        |
48      });
49    </script>
50  </head>
```

e Paste the code copied from the Example code onto the new line.

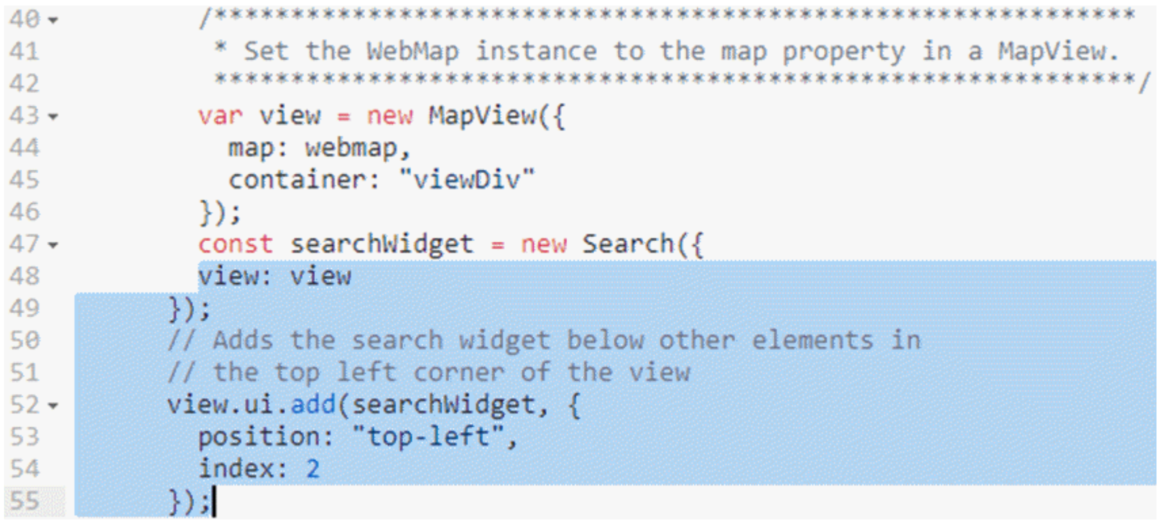
f Highlight lines 48 to 55—that is, all of the code you just pasted except for the first line—as shown in the following graphic.



```
40          /*************************************************************
41           * Set the WebMap instance to the map property in a MapView.
42           *************************************************************/
43          var view = new MapView({
44            map: webmap,
45            container: "viewDiv"
46          });
47          const searchWidget = new Search({
48    view: view
49  });
50  // Adds the search widget below other elements in
51  // the top left corner of the view
52  view.ui.add(searchWidget, {
53    position: "top-left",
54    index: 2
55  });
```

g On your keyboard, press Tab three times so that the code you pasted aligns properly with the rest of the code sample, as shown in the following graphic.

Aligning your code is a best practice that makes your code easier to read and troubleshoot.



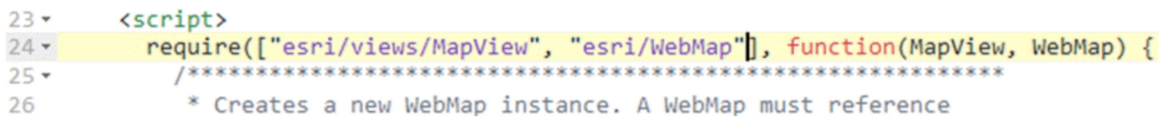
```
40          /*************************************************************
41           * Set the WebMap instance to the map property in a MapView.
42           *************************************************************/
43          var view = new MapView({
44            map: webmap,
45            container: "viewDiv"
46          });
47          const searchWidget = new Search({
48            view: view
49          });
50          // Adds the search widget below other elements in
51          // the top left corner of the view
52          view.ui.add(searchWidget, {
53            position: "top-left",
54            index: 2
55          });
```

You have just done the following:

- Created a search widget called searchWidget, an object of the class Search
- Associated a view to the MapView by setting the view property to an instance of the MapView that was instantiated (created) on line 43
- Added the Search widget to the MapView in the top-left corner on lines 52 and 53

But, for the Search widget to be added, you also need to load the Search module, which tells the app which widgets are going to be used.

h On line 24 of the editor pane, click to the right of the quotation mark following "esri/WebMap" and before the bracket, and then type a comma.

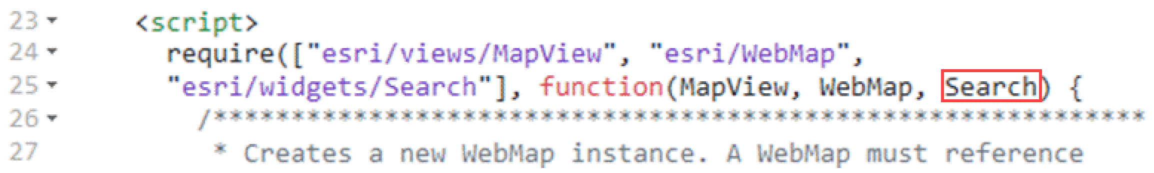


```
23    <script>
24      require(["esri/views/MapView", "esri/WebMap"], function(MapView, WebMap) {
25        /*************************************************************
26         * Creates a new WebMap instance. A WebMap must reference
```

i Press Enter to add a new line.

j On the new line, type **"esri/widgets/Search"**, making sure to capitalize Search and include the quotation marks.

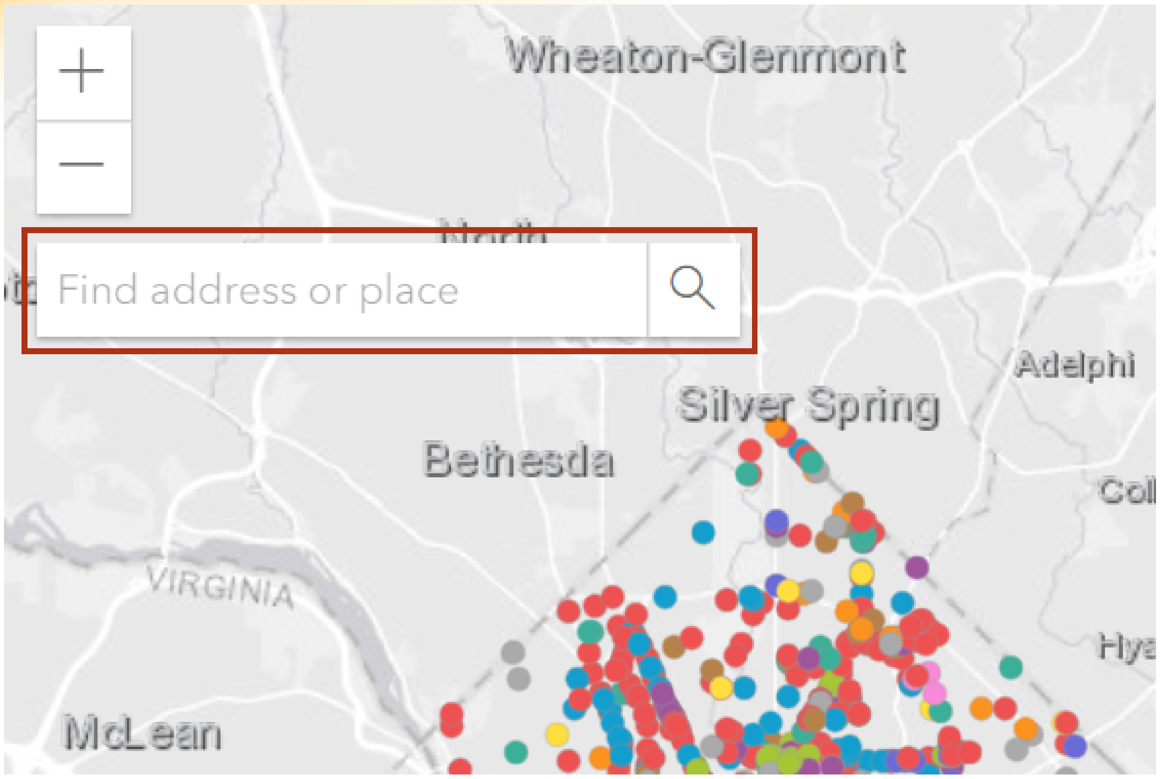
k On line 25, add a comma after WebMap and type **Search**, as shown in the following graphic.



```
23    <script>
24      require(["esri/views/MapView", "esri/WebMap",
25      "esri/widgets/Search"], function(MapView, WebMap, Search) {
26        /*************************************************************
27         * Creates a new WebMap instance. A WebMap must reference
```

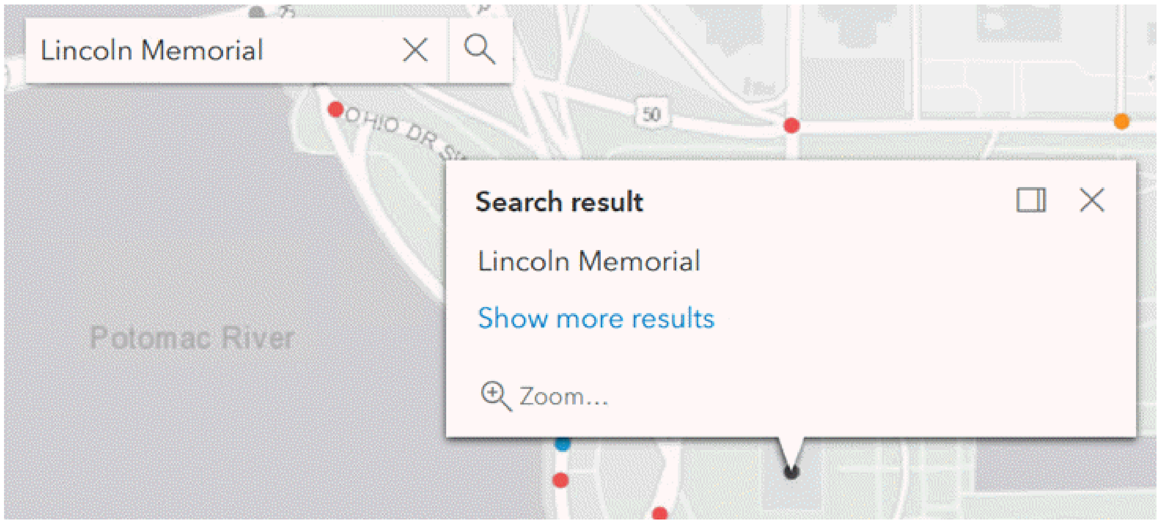
l In the sandbox, click Refresh.

In the preview window, you should see the Search widget in the top-left corner of the app, as shown in the following graphic. If there are errors in the code you just typed, you will not see an error message; however, the widget or the map itself will not display.



Now you will test the Search widget.

- In the search field, type **Lincoln Memorial**, and then either press Enter or click the magnifying glass icon.



The Search widget works as intended. You have just extended an existing web app by modifying the underlying code.

n Close your browser.

This concludes the Guided part of the Section 6 exercise. You can use the information you've learned whenever you work with geo apps. For example, you can make a web map, and then create a custom app to share it. Please continue on to the Do-It-Yourself part to apply what you've learned. Don't forget the developer and training resource links at the end.

Part II - Do-It-Yourself

Programming does not have to mean learning a new language and developing a program from scratch. Programming can also mean extending or modifying an existing program, like a web app, to change it or add functionality. In the Guided part of this exercise, you altered and added code to change the properties and functionality of a geo app. The Do-It-Yourself part contains optional goals to help you apply what you learned.

Even if you choose not to complete a DIY project, we ask that you read through this section so that you can find and learn from your fellow students' work. Don't forget to explore the additional developer and educational resources and samples found in the learning resources at the end.

Further explore the ArcGIS API for JavaScript

In this part of the exercise, we encourage you to challenge yourself and grow your custom app development skills. You can do either or both of the following options.

1. If you are new to JavaScript development
If you are a new developer, try experimenting with some of the other JavaScript samples (https://bit.ly/2k3c8nV). Look for the Explore In The Sandbox link.

Or, from the list on the ArcGIS for Developers site, choose a tutorial (https://bit.ly/3s61kDa) to guide you through a workflow of your choice.

Check out the resources at the end of this exercise to help you, and don't be afraid to experiment and ask questions.

2. If you are experienced with JavaScript
If you are not new to development, you can try building an app without using a template, relying on the API for JavaScript guide (https://bit.ly/2YYUjG6). Or you can download existing template source code from GitHub (https://bit.ly/2MjMYhK) and modify it to suit your needs, or use an SDK. You will need to host your custom app on a web server and then register (https://bit.ly/2GyBUem) the app in ArcGIS Online before you can share it with the MOOC community.

3. If you are a developer and new to GIS

If you are not new to development but are new to GIS, you can access ready-to-use location services and data services with ArcGIS Platform (https://bit.ly/2YKIOU3). These ArcGIS location services have REST API (https://bit.ly/3trJmw1) services that client APIs (https://bit.ly/2NYzKbU) can use to bring more functionality to your applications.

Share your work with the class

If you created a custom app with the ArcGIS API for JavaScript or any of the other APIs or SDKs, please share it with your peers in the class forums.

As with the other sections, to share your work with the class, you must post about it in the forums.

- If you created a new app, share your app with everyone. See below for more notes on sharing a custom app.
- Create a new forum post to tell students and instructors what you did.
- Add the link to your app, or screen shots, to the body of the post.
- Give the post a descriptive title, and add the hashtag **#DIYSection6**.

Sharing a custom app
There are many different routes you can take with this DIY option, and it is a little bit more advanced. But if you create an app to share, here are some notes on sharing:

1. If you are using the sandbox, like you did in the Guided part, click CodePen.
2. In CodePen, click Save, create a free CodePen account, and then click Sign Up and Save This Pen.

Now the page will have a unique URL that you can use to share your creation with others.

You can also click Download from the sandbox, which allows you to save an HTML file that you can share with others. Those with whom you share the file will have to save the file locally and then open it in their web browser. Although this option will not work for this class—because you cannot attach files to forum posts—it may be useful in other circumstances.

You can also choose to share custom code snippets, screen shots, and so on in the forum. It's up to you. Have fun!

Finding the work of other students

Now that you have shared your work in the forum, find and review the work of other students. Feedback from the MOOC community helps everyone learn and improve.

1. In the forums, search by the hashtag **#DIYSection6**.
2. Read other student posts and open their geo apps if they made one.
3. Reply to the forum post to give helpful feedback or ask questions.

Learning Resources

Congratulations! You just customized the code of a web app while learning a thing or two about the API Reference along the way. Developing is not always intuitive, but there are many resources from Esri and others that allow you to learn anything that you are motivated to learn. With a little time and dedication, you can create powerful geo apps, customized to your needs, for yourself or an employer.

Here are some more resources to continue learning:
HTML tutorial (https://bit.ly/3rObloB)
CSS tutorial (https://bit.ly/3lkhNRV)
JavaScript tutorial (https://bit.ly/2OZYcud)
ArcGIS API for JavaScript tutorials (https://bit.ly/3cBswDq)
Free developers account (https://bit.ly/2vQdIMI)
ArcGIS Developers documentation (https://bit.ly/1DiXbKg)
Script and Extend ArcGIS Products (https://bit.ly/3cCc85D)
Esri Community developers community (https://bit.ly/2BlghLV)
ArcGIS API for JavaScript videos on Esri Community (https://bit.ly/2OUMu3U)

More Esri training:
Esri course catalog search (https://bit.ly/2nOdlwW)
Web course: Basics of JavaScript Web Apps (https://bit.ly/2nXm5mh)
Web course: Introduction to the ArcGIS for Server REST API (https://bit.ly/2vR6XKo)
Instructor-led course: Introduction to Web Development Using ArcGIS API for JavaScript (https://bit.ly/2MjlOYu)