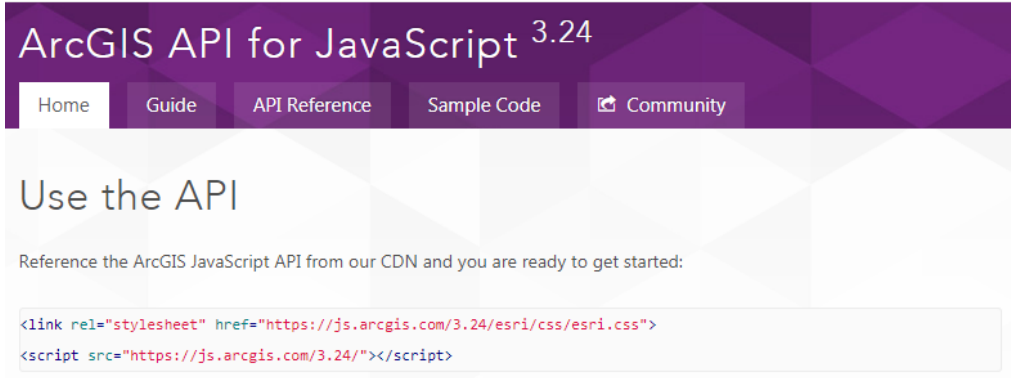
Using the ESRI JavaScript API

Accessing the API

To get the latest and greatest, we go to the ESRI Javascript API documentation website, js.arcgis.com. Right on the home page we can see the links we need to copy and paste into our HTML document.



Accessing the API

After copy/pasting the lines into the “head” element, we are ready to go!

```
<!DOCTYPE html>
<html>
        <head>
        <link rel="stylesheet" href="http://js.arcgis.com/3.24/esri/css/esri.css">
        <script src="http://js.arcgis.com/3.24/"></script>
        </head>

        <body>
                <div id=”mapDiv”></div>
                <script src=”javascript.js”></script>
        </body>
</html>
```

Asynchronous Module Definitions

Asynchronous Module Definitions (AMD) is how the ESRI Javascript API retrieves different **modules** from the API.

To keep from downloading more content than a web application needs, the ESRI JS API is separated into logical sections (modules).

One must call each module separately using the **require** statement.

AMD Require

The **require** statement wraps all code into one function that is only run when all modules are downloaded to the browser.

This function can also be told to wait to run until the page is finished loading.

```
require(["moduleName"],function(moduleName){
        //code that uses moduleName
});
```

AMD Require

Let's look at a real-life example of a require statement:

Esri "map" module

Wait to run until page is ready!

Only need a variable for the map module. domReady has an exclamation point.

```
require(["esri/map","dojo/domReady!"],function(Map){
    var options = {
        center: [-97.742581, 30.2837352],
        zoom: 12,
        basemap: "topo"
    };
        var map = new Map("mapDiv",options);
});
```

This bit of code places an esri map with a topographic basemap centered on Austin, TX into an HTML div element with an id of "mapDiv".

Javascript Constructors

The ESRI Javascript API uses **constructors** to create new API elements.

```
var map = new Map("mapDiv",options);
```

There are many elements that can be created. We will focus on these:

1. Map class
2. PictureMarkerSymbol class
3. Point class
4. InfoTemplate class
5. Graphics class

Add require statement

At the very top of your .js put these opening statements

```
require([
"dojo/on",
```

Insert the paths to ESRI modules here

```
"dojo/domReady!" ],
function (
on,
```

Insert variable names for ESRI modules here

```
) {
```

Remember you don't need a variable name for dojo/domReady!

At very the bottom of your .js put this closing tag

```
});
```

paths to ESRI modules:

```
"esri/map",
"esri/symbols/PictureMarkerSymbol",
"esri/graphic",
"esri/geometry/Point",
"esri/InfoTemplate",
```

variable names for ESRI modules:

```
Map,
PictureMarkerSymbol,
Graphic,
Point,
InfoTemplate
```

Javascript Constructors

The ESRI Javascript API documentation page *(js.arcgis.com)* can show you how each class is constructed.

	API location:
1. Map class	esri
2. PictureMarkerSymbol class	esri/symbols
3. Point class	esri/geometry
4. InfoTemplate class	esri
5. Graphics class	esri

The Map Class

Properties

Name	Type	Summary
height	Number	Current height of the map in screen pixels.
visible	Boolean	Indicates whether map is visible.

```
var myMapHeight = Map.height;
Map.height = 200;
```

Methods

Name	Return type	Summary
centerAndZoom(mapPoint, levelOrFactor)	Deferred	Centers and zooms the map.
disablePan()	None	Disallows panning a map using the mouse.

This is the Map class constructor: `new Map("mapDiv",options?)`

The Map Class

`new Map(`**`"mapDiv"`**`,options?)`

This is a parameter that tells the API which div element to place the map into. This is a string containing the element id.

We already have a "mapDiv" element defined in the .html.

Add this statement to your .js right above `PageLoad();`

```
var map = new Map("mapDiv", options);
```

Then replace `PageLoad();` with this statement

```
on(map, "load", function () { PageLoad(); });
```

This tells it to wait until after the map loads to run the `PageLoad()` function

The Map Class

```
new Map("mapDiv",options?)
```

This parameter handles all the different options regarding how the map is displayed and used. The acceptable input parameter is a Javascript object containing the appropriate properties and their values. The following is a sample map options object.

```
var options = {
     center: [-97.742581, 30.2837352],
     zoom: 12,
     basemap: "topo"
};
```

The PictureMarkerSymbol Class

```
new PictureMarkerSymbol(url,width,height);
```

```
require(["esri/symbols/PictureMarkerSymbol"], function(PictureMarkerSymbol) {
    var symbol = new PictureMarkerSymbol('http://www.esri.com/graphics/aexicon.jpg', 51, 51);
});
```

Collection of symbols
https://developers.arcgis.com/javascript/3/samples/portal_symbols/index.html

Add this to your .js after the `on(map, "load")` function

```
var symbol = new PictureMarkerSymbol("Images/camera-icon.png", 20, 20);
```

The Point Class

```
new Point(x,y,<SpatialReference>);
```

Great for special projections

```
new Point(coords,<SpatialReference>);
```

Useful if you already have an array

```
new Point(long,lat);
```

Simple lat/long numbers

```
new Point(point[]);
```

Simple lat/long array

```
new Point(point{});
```

Simple lat/long object

Put this after your new PictureMarkerSymbol() code

```
function loadGraphics() {
        for (var i in images) {
                var geometry = new Point(images[i].location);
        }
}
on(map, "load", function () { loadGraphics(); PageLoad(); });
```

The Graphic Class

```
new Graphic(geometry?,symbol?,attributes?,infoTemplate?)
```

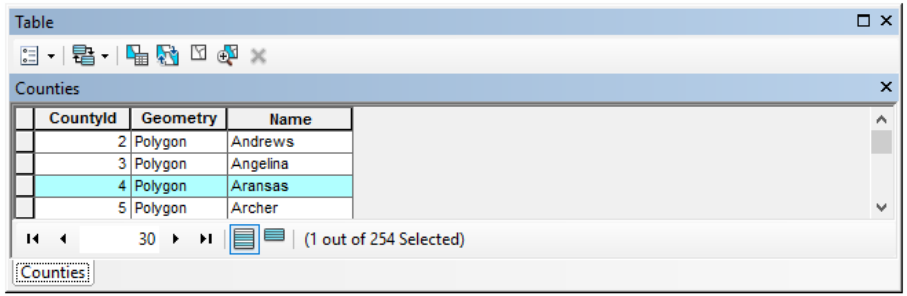
<Geometry> geometry	Optional	The geometry that defines the graphic.
<Symbol> symbol	Optional	Symbol used for drawing the graphic.
<Object> attributes	Optional	Name value pairs of fields and field values associated with the graphic.
<InfoTemplate> infoTemplate	Optional	The content for display in an InfoWindow.

Create a Graphic for each image and add each Graphic to the map:

```
function loadGraphics() {
        for (var i in images) {
                var geometry = new Point(images[i].location);
                var graphic = new Graphic(geometry, symbol);
                map.graphics.add(graphic);
        }
}
```

Attributes

The **attributes** are name-value pairs that describe the graphic.



Name

Name

```
var attr = { "index": i, "Image": images[i].title };
```

Value

Value

The InfoTemplate Class

```
new InfoTemplate(title, content);
```

Use a wildcard to automatically include all the attribute's name value pairs.

```
require(["esri/InfoTemplate"], function(InfoTemplate) {
    var infoTemplate = new InfoTemplate("Attributes", "${*}");
});
```

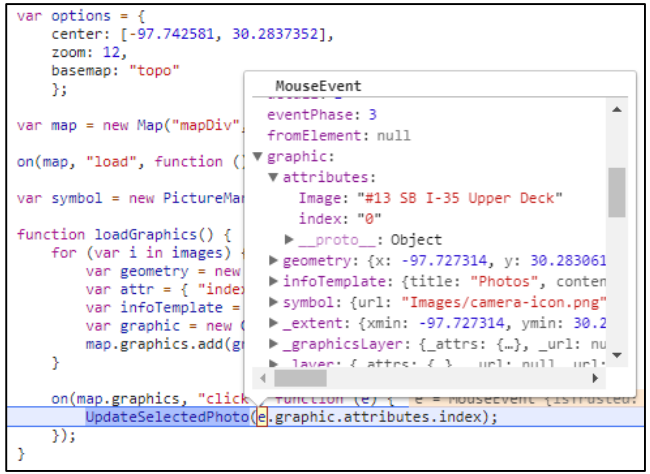
Display only the specified fields.

```
require(["esri/InfoTemplate"], function(InfoTemplate) {
    var infoTemplate = new InfoTemplate("Attributes",
                        "<tr>StateName:<td>${STATE_NAME}</tr></td><br><tr>Population:<td>${Pop2001}</tr></td>");
});
```

```
var infoTemplate = new InfoTemplate("Photos", "${Image}");
var graphic = new Graphic(geometry, symbol, attr, infoTemplate);
```

Dojo on()

```
require(["dojo/on"], function(on){
  on(target, "event", function(e){
    // handle the event
  });
});
```



Add this inside of your loadGraphics() function

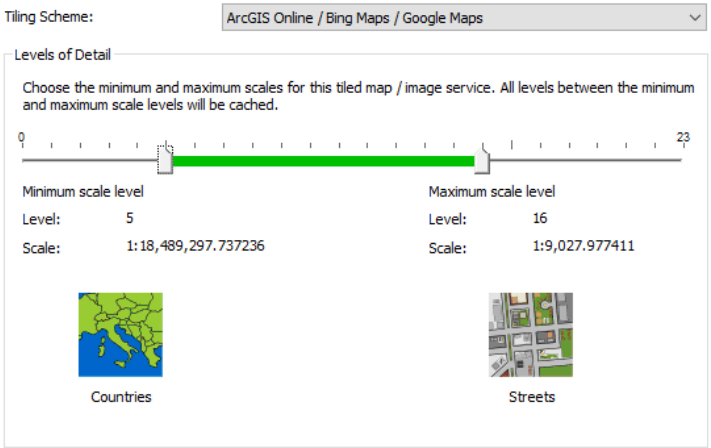
```
on(map.graphics, "click", function (e) {
        UpdateSelectedPhoto(e.graphic.attributes.index);
});
```

It will update the photo when you click on a graphic.

Pulling it together

Write a function that zooms to a point

```
function zoomToGraphic(point) {
    map.centerAndZoom(point, 16);
}
```



Create a point and call the zoom function each time you move to a different photo

```
function MoveSelectedPhoto(newPhotoIndex) {
    UpdateSelectedPhoto(newPhotoIndex);
    zoomToGraphic(new Point(images[newPhotoIndex].location));
}
```