3D visualisation with ArcGIS API for JavaScript

Esri Singapore Pte Ltd | 82 Ubi Avenue 4, 07-03 Edward Boustead Centre

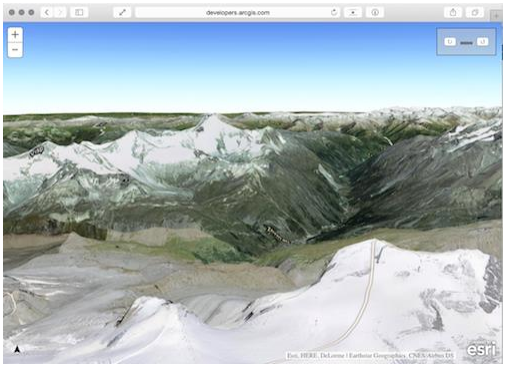
+65 6742 8622 connect@esrisingapore.com.sg esrisingapore.com.sg

Discover ArcGIS API for JavaScript 4.0

Version 4.0 is the next generation ArcGIS API for JavaScript and it supports both 2D and 3D.

With version 4.0, the API continues to be a modern, top-of-the-line web mapping API with new cutting-edge features, improved developer experience and cleaner implementation. It offers a more simple interface, which includes support for 3D.

Mapping components in 4.0 cleanly separate map and layers from their display logic. This separation allows map and layers to represent your data, enabling the display engine (map views and layer views) to use traditional HTML/SVG DOM or Canvas or WebGL, depending on the type of data you want to visualise and the type of applications you want to build.



As a developer, you will be able to write both 2D and 3D applications, and consume and author web maps and web scenes using the same concepts, programming patterns and a consistent API. The API provides structured, streamlined methods for developing JavaScript-based mapping applications. Class names have been modified to be shorter and clearer with consistent casing.

Confusing concepts such as accessing properties and multiple constructor signatures have also been enhanced, modified and removed. As an example, a class may be created by providing properties in an object inside the constructor. This means users no longer have to memorise the order of constructor parameters.

For more information, see the ‘Working with the API’ section in the developers guide.

Introduction

This document will take you through a streamlined process of creating a simple map in a 3D scene view.

Get started with web 3D visualisation

Reference the ArcGIS API for JavaScript

First, set up a basic HTML document similar to the example below:

```
<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no">

<title>Get started with SceneView - Create a 3D map</title>

</head>

</html>
```

Inside the `<head>` tag, reference the ArcGIS API for JavaScript using the `<script>` and `<link>` tags:

```
<link rel="stylesheet"
href="https://js.arcgis.com/4.0beta3/esri/css/main.css"></p>
```

```
<script src="https://js.arcgis.com/4.0beta3/"></script>
```

The `<script>` tag loads the ArcGIS API for JavaScript from a CDN. When new versions of the API are released, update the version number to match the newly released version.

The `<link>` tag references the `main.css` style sheet that contains styles specific to Esri widgets and components. For more information about this style sheet, refer to the help topic on required CSS.

Load the modules

Use a second `<script>` tag to load specific modules from the API. Load the following modules using the syntax below:

- `esri/Map` – loads code specific to creating a map
- `esri/views/SceneView` – loads code that allows users to view the map in 3D
- `dojo/domReady!` – ensures the DOM is available before executing the code.

In addition to dojo/domReady! (the bang or exclamation point denotes that domReady is an AMD loader plug-in), Dojo also provides `dojo/ready`. The difference between domReady! and ready is the former waits to fire the callback provided until the DOM is available, while the latter waits for the DOM to be ready and waits for all outstanding required calls to finish. For more information, see the Dojo documentation for dojo/ready. In simple cases, dojo/domReady! should be used. If an app uses `parseOnLoad: true`, Dojo dijits, widgets from the Esri library or custom dijits, dojo/ready, should be used.

Create the map

A new map is created using `Map`, which is a reference to the map class that was loaded from the `esri/Map` module. You can specify map properties, such as `basemap`, by passing an object to the map constructor.

```
require([
  "esri/Map",
  "esri/views/SceneView",
  "dojo/domReady!"
], function(Map, SceneView) {
  var map = new Map({
    basemap: "streets"
```

```
  });
});
```

Additional basemap options are `satellite`, `hybrid`, `topo`, `gray`, `dark-gray`, `oceans`, `osm`, `national-geographic`. Use alternate basemaps by modifying the `basemap` option in the sandbox. View the map class for more details on additional map options.

Because the map variable is declared outside of a function, it is a global variable. This is useful while testing or debugging because the browser's developer console can be used to interact with the map.

Create a 3D view

Views reference nodes that serve as containers in HTML files, allowing users to view the map inside an HTML page. Create a new `SceneView` and set its properties by passing an object to its constructor.

```
require([
  "esri/Map",
  "esri/views/SceneView",
  "dojo/domReady!"
], function(Map, SceneView) {
  var map = new Map({
    basemap: "streets"
  });
  var view = new SceneView({
    container: "viewDiv",  //reference to the DOM node that will contain the view
    map: map  //references the map object created in step 3
  });
});
```

In this snippet we set the `container` property to the name of the DOM node that will hold the map. The `map` property references the map object we created in the previous step. See the scene view documentation for additional properties you may set on the view, including `center` and `scale`, which may be used to define the initial extent of the view.

There are two types of views available: map view (used for viewing 2D maps) and scene view (used for viewing 3D maps). Click here to learn how you can create maps with 2D views.

Define the page content

The JavaScript needed to create a map and a view is now complete. The next step is to add the associated HTML for viewing the map. For this example, the HTML is very simple: add a `<body>` tag, which defines what is visible in the browser, and a single `<div>` element inside the body where the view will be created.

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

The `<div>` has an ID of scene view to match the ID passed to the scene view's container property in the constructor.

Style the page

Style the content of the page using `<style>` tags inside the `<head>`. To make the map fill the browser window, add the following CSS inside the page's `<style>`:

```
<style>
  html, body {
    padding: 0;
    margin: 0;
  }
</style>
```

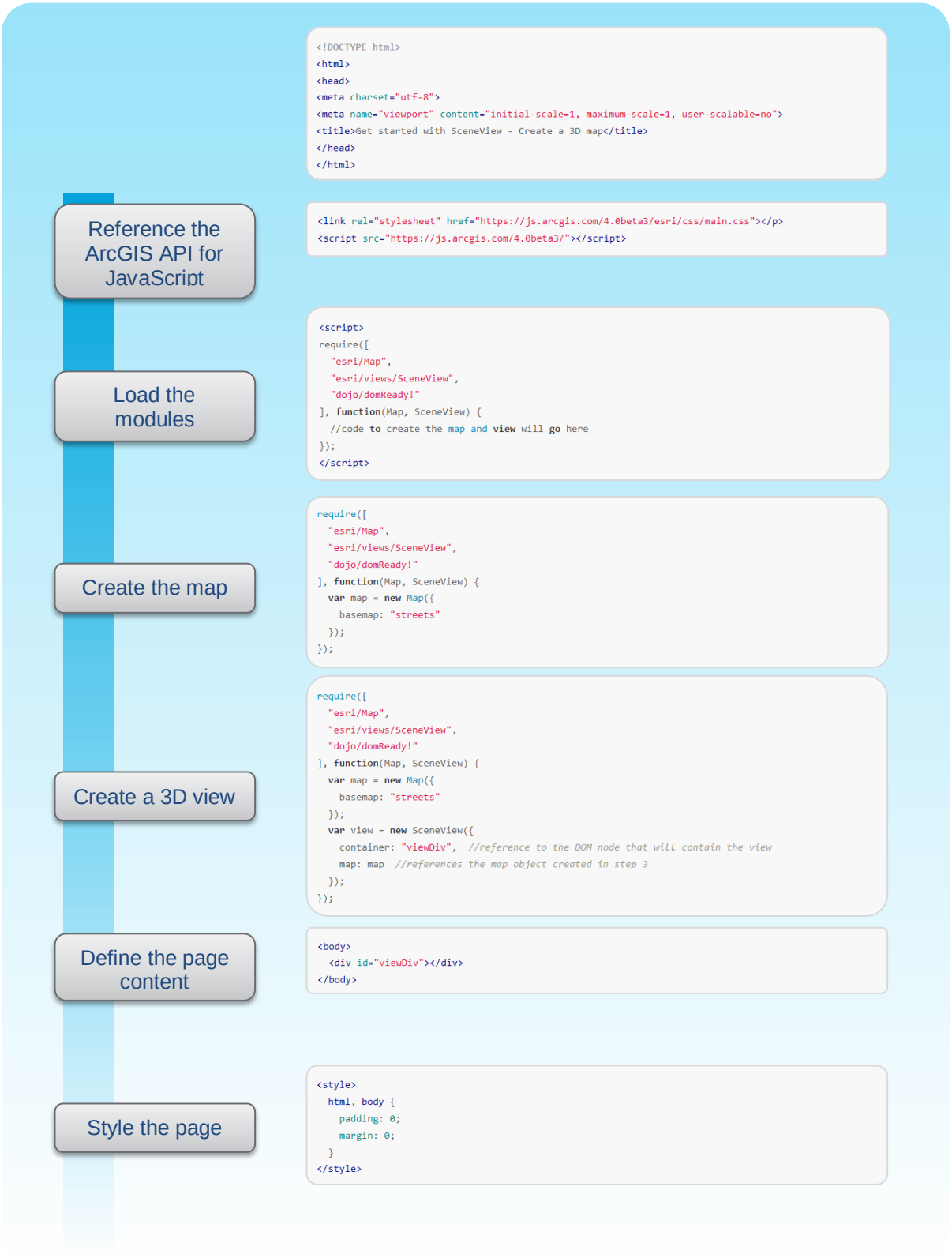
Code Summary

You have now built your first web application in 3D using the ArcGIS API for JavaScript 4.0. Your final HTML code should look like the example below:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no">
<title>Get started with SceneView - Create a 3D map</title></p>
<style>
  html, body {
    padding: 0;
    margin: 0;
  }
</style>
<link rel="stylesheet" href="https://js.arcgis.com/4.0beta3/esri/css/main.css">
<script src="https://js.arcgis.com/4.0beta3/"></script>
<script>
require([
  "esri/Map",
  "esri/views/SceneView",
  "dojo/domReady!"
], function(Map, SceneView){
  var map = new Map({
    basemap: "streets"
  });
```

```
  var view = new SceneView({
    container: "viewDiv",  //reference to the scene div created in
step 5
    map: map,  //reference to the map object created before the scene
    scale: 50000000,  //sets the initial scale to 1:50,000,000
    center: [-101.17, 21,78]  //sets the center point of view with
lon/lat
  });
});
</script>
</head>
<body>
  <div id="viewDiv"></div>
</body>
</html>
```

Steps summary



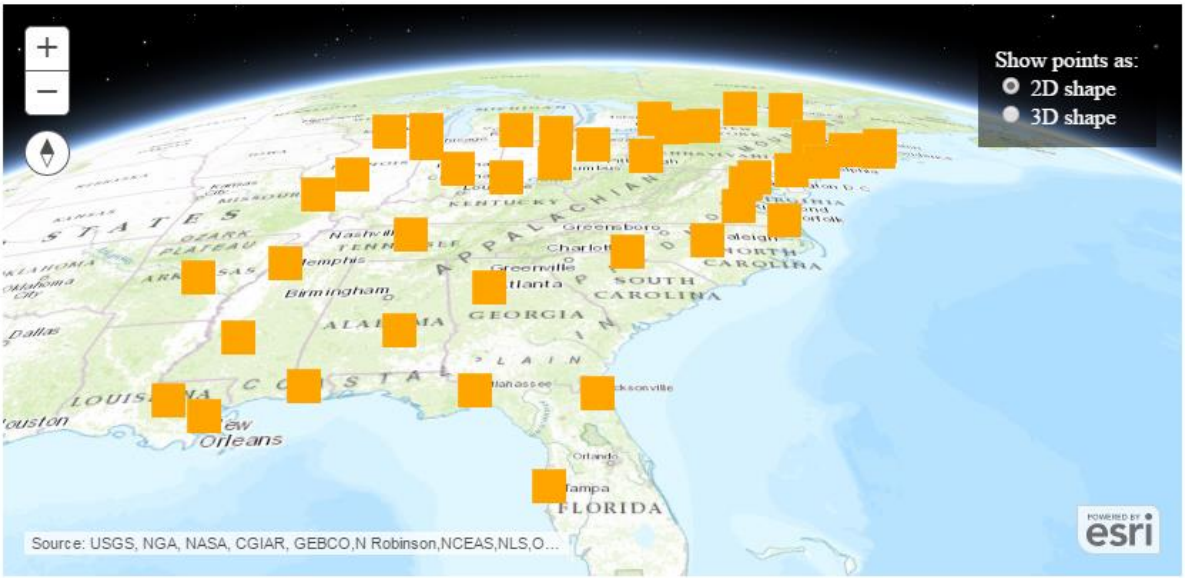
Next Steps

How to add a 3D scene layer

View live sample

Explore in the sandbox

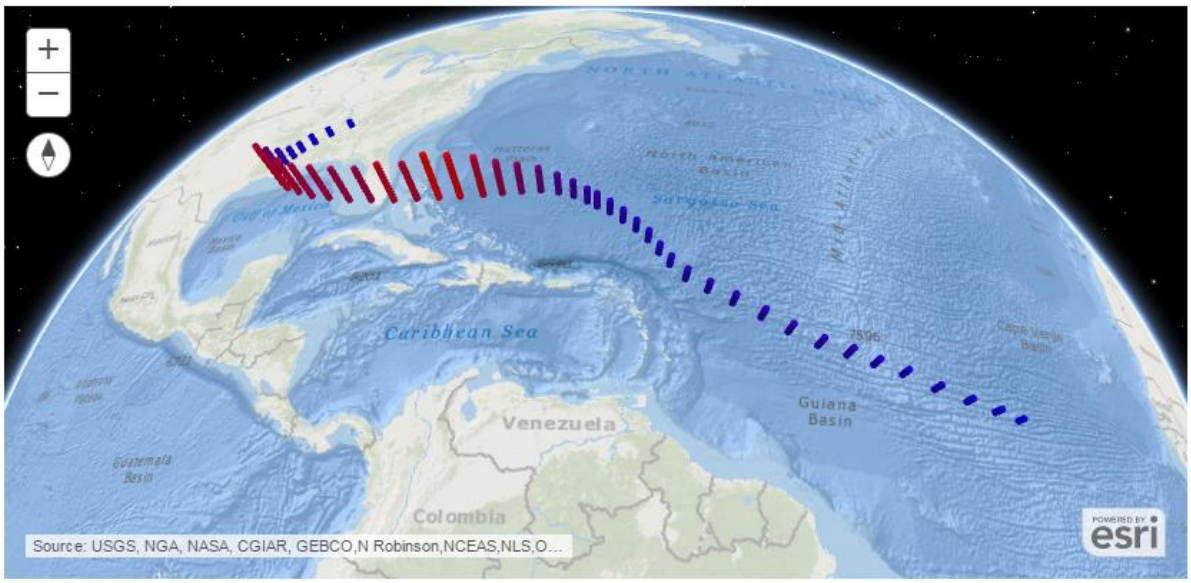
How to visualise a 2D feature layer in 3D



View live sample

Explore in the sandbox

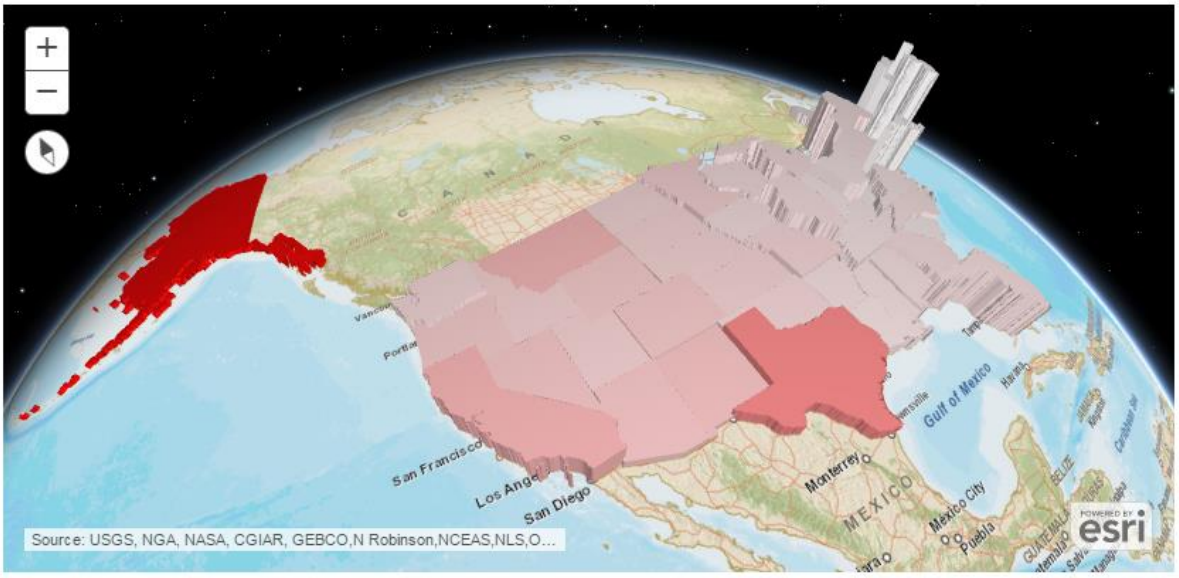
How to render 2D geometries with 3D symbols



View live sample

Explore in the sandbox

How to extrude 3D polygon features based on a field value



View live sample

Explore in the sandbox