ArcGIS API for JavaScript: Advanced Topics

Andy Gup

@agup

My goals?

Discuss important coding patterns

Help your long-term coding skills

Faster, better, more maintainable code

Save you a ton of time!

Your skill sets?

- Moderate knowledge of JavaScript
- Moderate usage of ArcGIS JavaScript API
- Basic familiarity with dojo & AMD
- Moderate usage of 3rd party libraries

Who am I?

Andy Gup

Developer Evangelist

agup@esri.com

www.andygup.net

github.com/agup

@agup

We will cover:

- Object-oriented JavaScript
- Unit Testing
- Custom widgets
- Performance
- Working with 3rd party JS libraries
- Offline JavaScript

Object Oriented JavaScript

AMD

require, define

modules

declare Classes

Closure

Scope

Legacy dojo loader – bye bye

```
<script>var dojoConfig = { parseOnLoad: true };</script>
<script src="http://js.arcgis.com/3.8/"></script>
<script>
  dojo.require("dijit.layout.BorderContainer");
  dojo.require("dijit.layout.ContentPane");
  dojo.require("dijit.layout.StackContainer");
  dojo.require("dijit.form.HorizontalSlider");
  dojo.require("dijit.form.HorizontalRuleLabels");
  dojo.require("dojox.figure.fun.fold")
  dojo.require("esri.map");
  dojo.require("esri.layers.FeatureLayer");
```

AMD

Asynchronous Module Definition API

Global functions for loading code

Normally asynchronous & concurrent

Decreases page load times

Also used by RequireJS, curl, bdLoad, etc

What about Dojo AMD??

AMD community standards compliant

Modules work with other AMD loaders/libraries!

require – load modules

```
require(
   // (optional; object) configuration object
   configuration,

   // (optional; array of strings) module identifiers to
   // load before calling callback
   dependencies,

   // (optional; function) function to call when
   // dependencies are loaded
   callback
)
```

require – load modules

```
require([
        "esri/map",
        "esri/arcgis/utils",
        "dojo/domReady!"
        ], function(Map, arcgisUtils){

    var map = new Map(“mapDiv”...);

});
```

Module

Alias

define – load modules (no config)

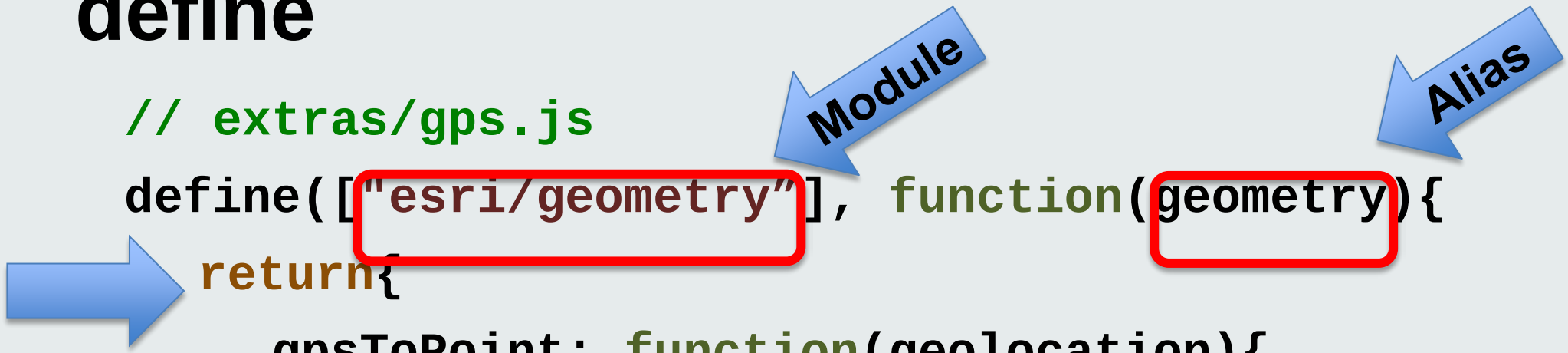
```
define(
   // (optional; string) explicit module identifier
   moduleId,

   // (optional; array of strings) list of module
   // identifiers to load before calling factory
   dependencies,

   // (function or value) the value of the module, or a
   // function that returns value of the module
   factory
)
```

define

Module creation is lazy and asynchronous

Modules are cached in the module namespace

define




```
// extras/gps.js
define(["esri/geometry"], function(geometry){
   return{
      gpsToPoint: function(geolocation){
         var point = new geometry.Point(...);
      }
   }
})
```

Custom module pattern – using dojo CDN

```
<script><!- Declare before loader executes -->
   var locationPath = location.pathname.replace(/\/[^/]+$/, "");
   var dojoConfig = {
      async: true,
      paths: {
         extras: locationPath  + "/extras"
      }
   }
</script>
<script src="http://js.arcgis.com/3.8/"></script>
<script>
require([“esri/map”,”extras/gps”],function(map,gps){
   . . .
   var point = gps.gpsToPoint(...);
}
</script>
```

Incorrect path

DOM ready

```
//Works with both define and require
define(["esri/geometry”,”dojo/domReady!”],
   function(geometry){

      //Wait until DOM is ready
      function init(){
         // TO-DO
      }()// Run immediately
   }
)
```

declare - named Classes

```
// You must define "dojo/_base/declare" first
// Allows for inheritance
declare(”namespace.ClassName”,null, {
   _privateVar : null,
   publicString : ””,
   init: function(){
      //TO-DO
   },
   changeSymbol : function(){ . . . }
})
```

declare - anonymous Classes

```
// Scoped anonymous class (no namespace)
// Allows for inheritance
var ClassName = declare(null, {
   _privateVar : null,
   publicString : “”,
   init: function(){
      //TO-DO
   },
   changeSymbol : function(){ . . . }
})
```

Why care about require, define, declare?

- Scalable
- Avoids repeated code (smaller download)
- More stability
- Reduced global variable leakage
- Reduced debugging time
- You “can” use it with 3rd party JS libraries!

require, define, declare

```
    var locationPath = location.pathname.r
    var dojoConfig = {
        paths: {
            extras: locationPath  + "/extr
        }
    }
</script>
<script src="http://js.arcgis.com/3.8/"></
<script>
    "use strict"
    var map;    //Global

    require(["esri/map","extras/test2","do
        map = new Map("map", {
            basemap: "topo",
            center: [-122.45,37.75], // lo
            zoom: 13,
            sliderStyle: "small"
        });

        var myTest = new extras.MapTools()

        map.on("load",function(evt){

            myTest.setExtent(map);
            console.log("Test 2 get map id
        })

    });
```

Unit Testing

Validates methods, properties and callbacks

Isolates the smallest piece of testable code

Jasmine, Mocha and others

http://pivotal.github.io/jasmine/

Why unit tests?

Absolutely helps you write better code

Helps you quickly test code after changes

Downsides to unit tests?

Requires learning the testing library.

They are “only” as good as the test case!!

Jasmine – one example

Open source project

Behavior driven

Independent of 3rd party JS frameworks

Does not require a DOM (fast)

Can be used stand-alone (locally)

Jasmine – app life cycle

Load CSS

Load Map

Load Layer(s)

Final Layer event loads Jasmine

e.g. “load” or “layer-add-result”

Jasmine loads unit test spec

Jasmine – example

```
var map;

require(["esri/map",
  "esri/layers/ArcGISTiledMapServiceLayer", "dojo/domReady!"],
  function(Map, Tiled) {
     map = new Map("map");
     var tiled = new Tiled("http://. . ./Terrain/MapServer");
     map.on(“load”,runJasmine)
     map.addLayer(tiled);

     function runJasmine(){
        . . .
     }
  }
);
```

Jasmine – example spec

```
“use strict”
describe(“My map tests”, function(){
   it("validate map", function(){
        expect(map.id).toEqual("map");
        expect(map).toEqual(jasmine.any(Object));
   })
   it(“load feature layer”, function(){
      var done = function(evt){
         expect(evt.layer.type).toEqual(“Feature Layer”)
      }
      map.on("layer-add-result", done);
   }
}
```

Jasmine test example

Widgets

Application framework

Singular purpose

Reusable/shareable

Examples:

- Attachment Editor Widget
- Directions Widget
- Geocoding Widget

Custom Widgets

Lifecycle of dijits

dojo/dom.byId

dijit/registry

registry.byId()

dijit/_WidgetBase

dijit/_TemplatedMixin

Dijit Basics

Created with dojo.declare

Inherit from dijit._WidgetBase

Use dijit._TemplatedMixin

Dijit Lifecycle (minimum req's)

Override *at least* the following:

- `constructor`
- `startup`
- `destroy`

Custom widget example

Client Performance

How long does user have to wait??

…on their machine

…on their internet connection

Client Performance

Generalizing features

Custom scale dependencies

Reducing number of layers

Reducing attributes and properties

Client Performance

Querying/Retrieving features

Parsing

Displaying

Analyze CPU performance

Use timers to check elapsed times

Performance - Parsing Features

How many features per response?

How complex are the features?

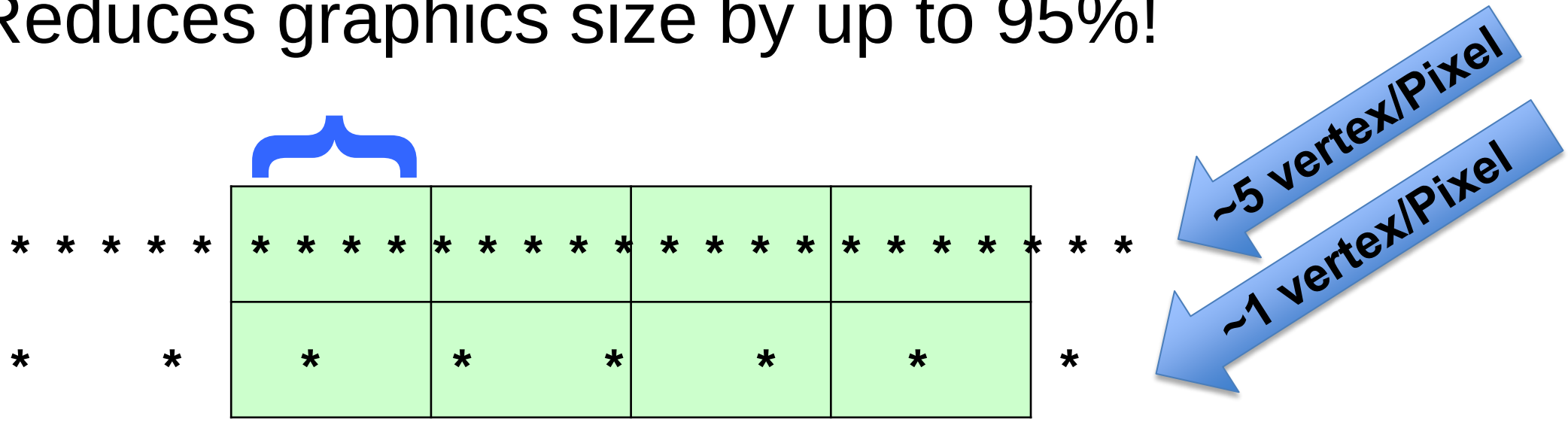
How efficient is your parser algorithm?

maxAllowableOffset (FeatureLayer)

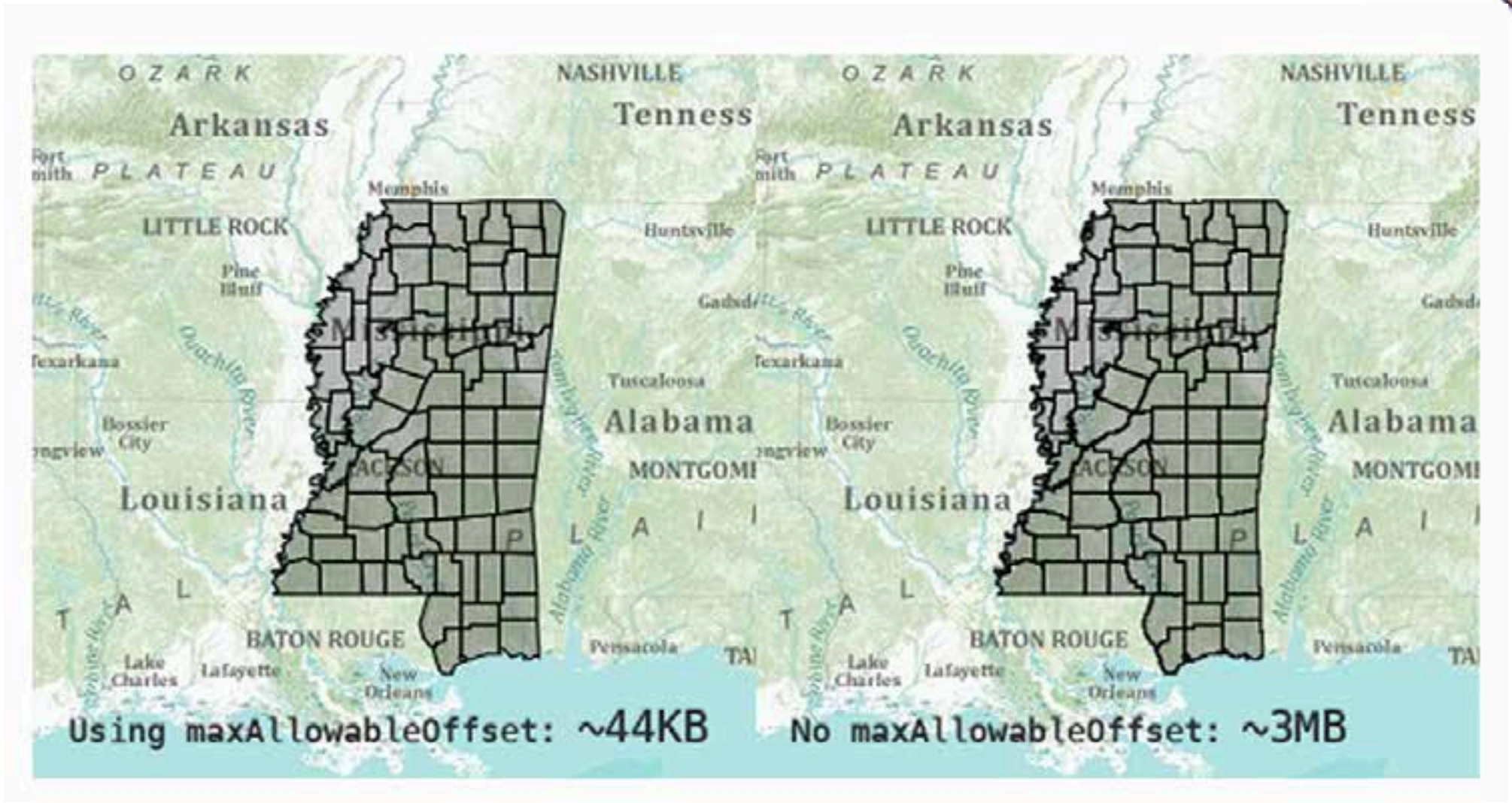
Display more than one vertex per pixel

JS API v2.7+ with ONDEMAND mode

Reduces graphics size by up to 95%!



maxAllowableOffset (FeatureLayer)



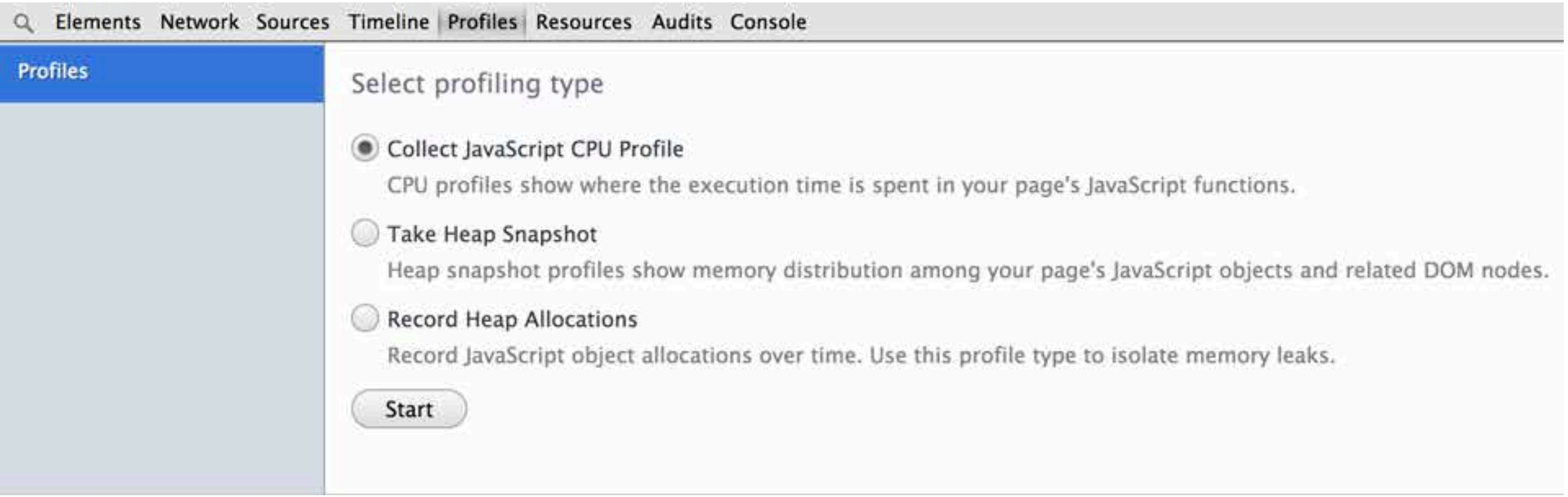
FeatureLayer.minScale & maxScale

```
// Limit zoom levels
var fl = new esri.layers.FeatureLayer(url, options);
   on(“load”, function() {
      fl.minScale = minScale; //min visible. Number
      fl.maxScale = maxScale; //max visible. Number
});
```

Performance demos

[Polygon Performance]

Performance Profiling



Working with 3rd Party JavaScript

jQuery, Bootstrap, Angular.js, node.js etc etc

Problem areas:

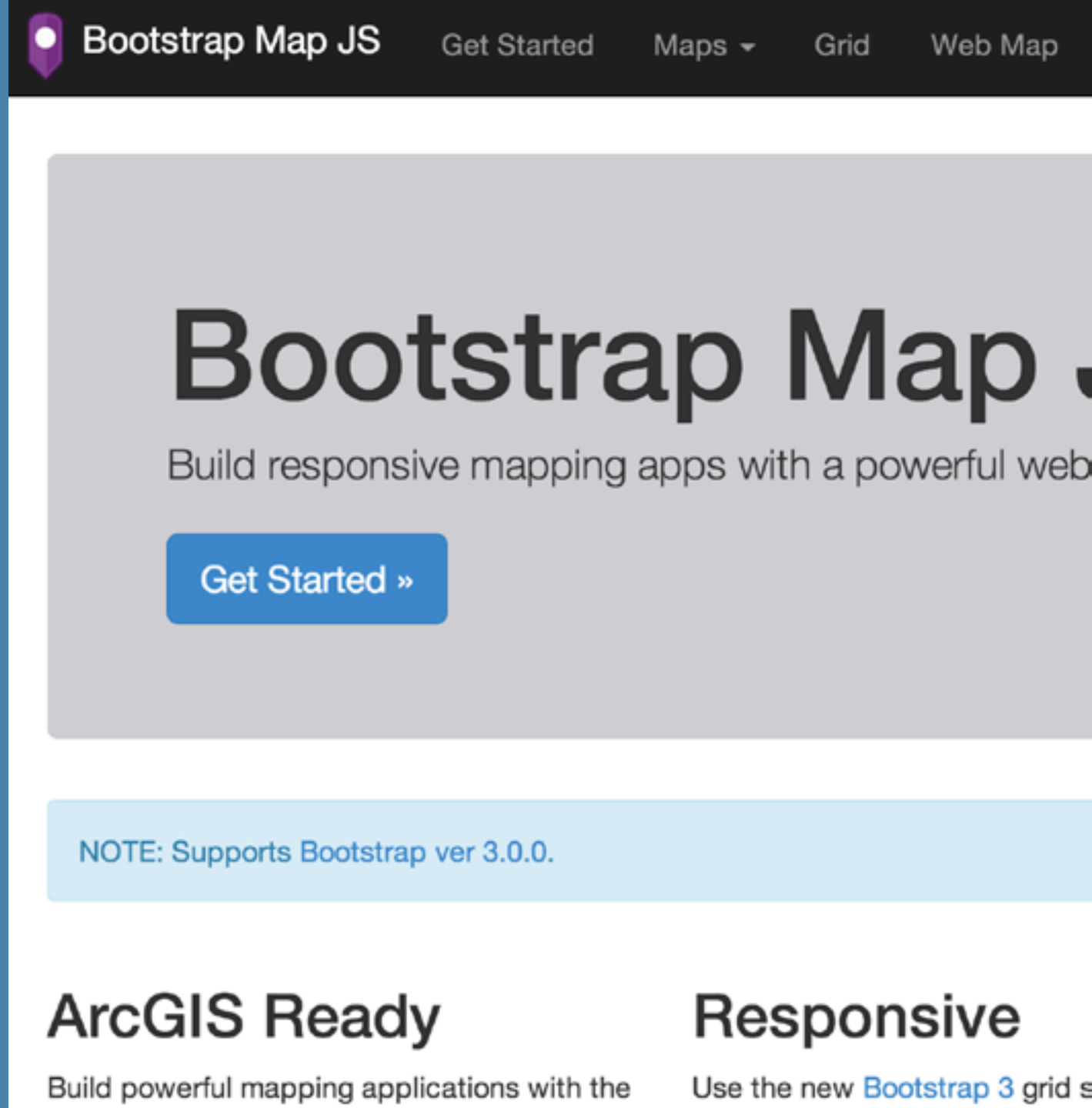
- Getting map to load
- Rotating map
- Resizing map
- Loading maps in secondary views (or tabs)

Working with 3rd Party JavaScript

github.com/esri

- Twitter bootstrap
- Dojo boostrap
- jQuery Mobile
- PhoneGap Quickstart
- Sencha Touch
- angular.js (coming soon)

Working with 3rd party JavaScript libraries



Offline JavaScript

Partially or fully disconnected workflows

Reuses existing JavaScript skills

Cross-platform

Offline data collection

Search & Rescue, inspections, delivery, etc

Offline JavaScript

Offline Editing

Offline Tile Management

Resyncing Edits

github.com/esri repository coming soon!!

Offline JavaScript – storing data in browser

IndexedDB (~50 MB)

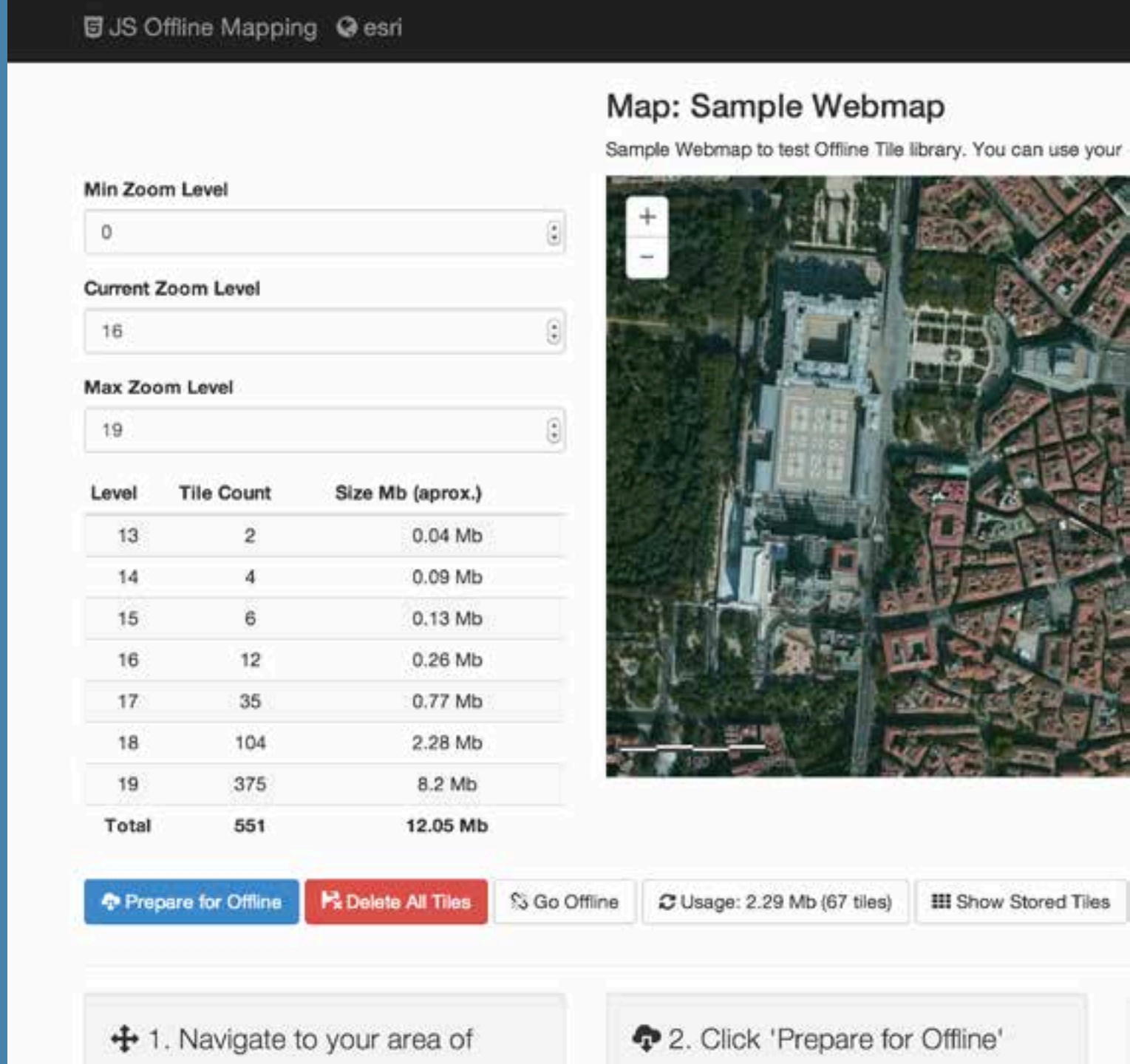
Web Storage (=< 5 MB)

SQLite (PhoneGap)

SD Card (PhoneGap)

FileWriter/FileReader (?)

Offline JavaScript



Recap

Object-oriented JavaScript

Unit Testing

Custom widgets

Performance

Working with 3rd party JS libraries

Offline JavaScript

Questions??

Andy Gup

Developer Evangelist

agup@esri.com

github.com/agup

@agup

Understanding our world.