ArcGIS API for JavaScript: Building Mobile Web Apps

Andy Gup

@agup

2019 ESRI
FEDERAL GIS CONFERENCE
WASHINGTON, DC

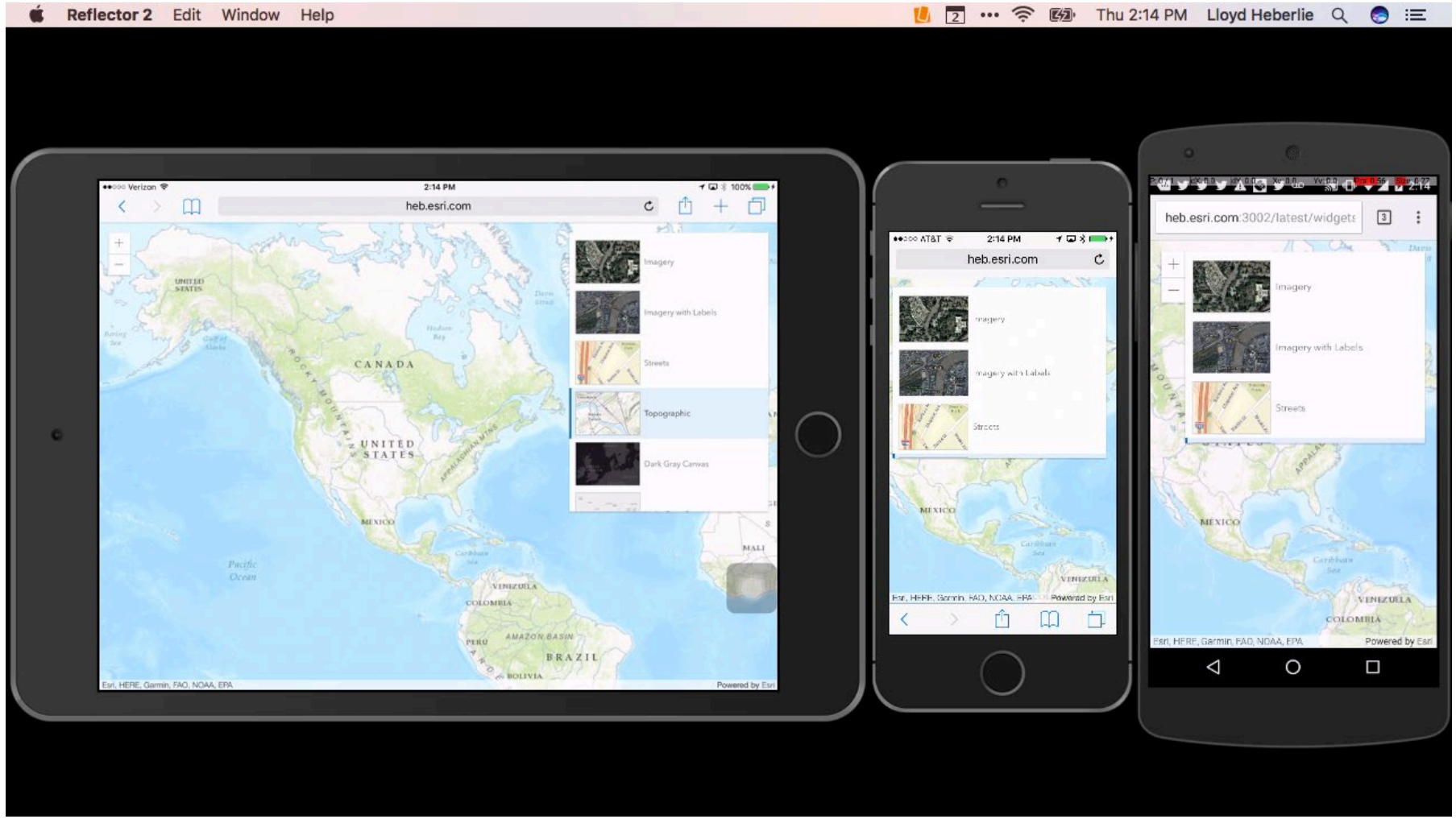
Agenda

- Responsive design
- Popups and Widgets
- Hybrid Approaches
- Geolocation
- Performance and Usability

Responsive Map App Design

- Support a variety of screens
- Uses breakpoints
- Fluid design

Support a variety of screens

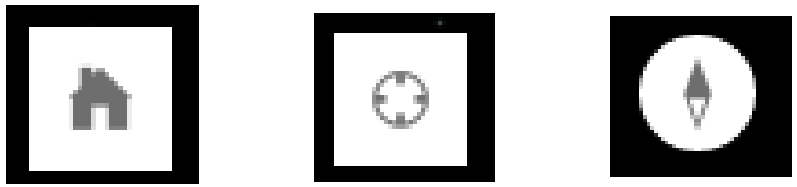


Responsive Popups

- Demo 1 – dockable popup
- Popup.docEnabled, Popup.docOptions
- Demo 2 - docEnabled

Widgets important for Mobile

- Demo 1 - Home
- Demo 2 - Locate
- Demo 3 - Compass

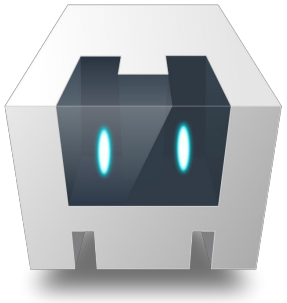


Hybrid = JavaScript + Native

There are several frameworks…here are a few examples

Provides access to native functionality

e.g. Geolocation, Bluetooth, SQLite, Touch ID



APACHE CORDOVA™

Cordova/PhoneGap

github.com/Esri/quickstart-map-phonegap

github.com/Esri/cordova-plugin-advanced-geolocation

APACHE
CORDOVA™

Ionic (Angular + Cordova)

github.com/andygup/ionic2-esri-map

Geolocation

github.com/Esri/html5-geolocation-tool-js

github.com/Esri/cordova-plugin-advanced-geolocation

Mobile Web Map Performance

Smart device vs Laptop

UI Jerkiness

Improving the mobile experience

Smart device vs. Laptop

Often significantly less user memory

Slower application loading/performance

Mobile internet speeds fluctuate widely

VS

UI Jerkiness (Jank)

Interruptions in frame production (fps) and latency

Temporary app lockups

Scrolling latency

Slow app response

Browser warning messages

Browser crashes

UI Jerkiness (Jank)

Caused by ANY operation taking longer than

16ms*

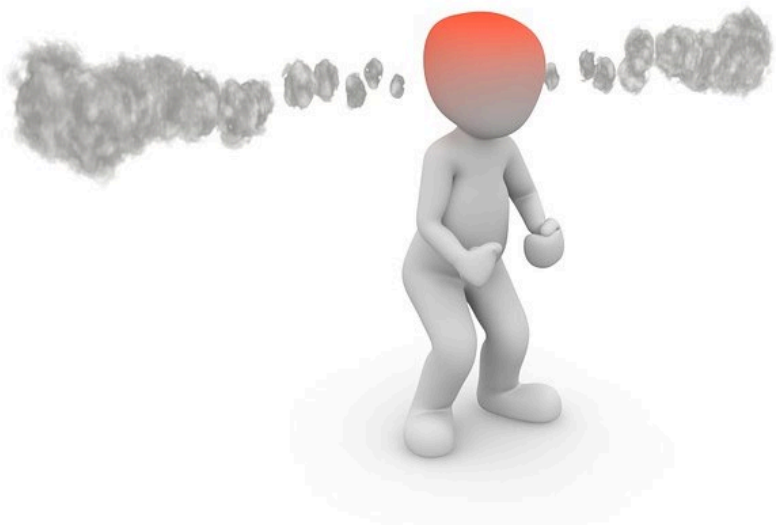
- **~60Hz or 60 FPS**

[Demo]

UI Jerkiness (Jank)

Significant jank can lead to:

- **Decreased usability**
- **Decreased productivity**
- **User frustration**
- **Lack of user return visits**



Common Causes of UI Jerkiness (Jank)

Slow(er) internet – delays in loading

Downloading large files or many files

Heavy processing on main thread

DOM layout thrashing

Improving Mobile Experience

Simplify!

Lazy load content

Use fewer map layers

Specify outfields on feature layers

Simplify rendering

Use Web Workers

Lazy load content

Defer the initialization of Classes

Demo

```
function lazyLoadTest(){
    require([
      "esri/geometry/geometryEngine"
    ], function(geometryEngine) {
      console.log("geometryEngine lazy loaded!");
    });
}
```

Lazy Load Content

Defer loading layers with `Map.add()` **or** `Map.addMany()`

```
// Load layers immediately
var map = new Map({
  layers: [featureLayer, graphicsLayer]
};


// Defer adding layers
map.addMany([ featureLayer, graphicsLayer]);


// Or, defer the loading of a single layer
map.add(featureLayer);
```

Use fewer layers for mobile

Fewer layers == better performance and less memory usage!

Fewer layers take less time to load

Remove unused layers via `Map.remove()`

Specify outfields on feature layers

Reduce response payload size and avoid using:

```
featureLayer.outFields = [“*”];
```

Instead, only specify what you need, for example:

```
featureLayer.outFields = [“Name”, “OBJECTID”, “Address”];
```

Simplify Rendering

Set a `FeatureLayer.definitionExpression` where possible

- **Reduces number of features looped by Renderer**
- **Speeds up rendering**

Demo 1 – Basic Definition Expression
Demo 2 – Definition Express + Arcade

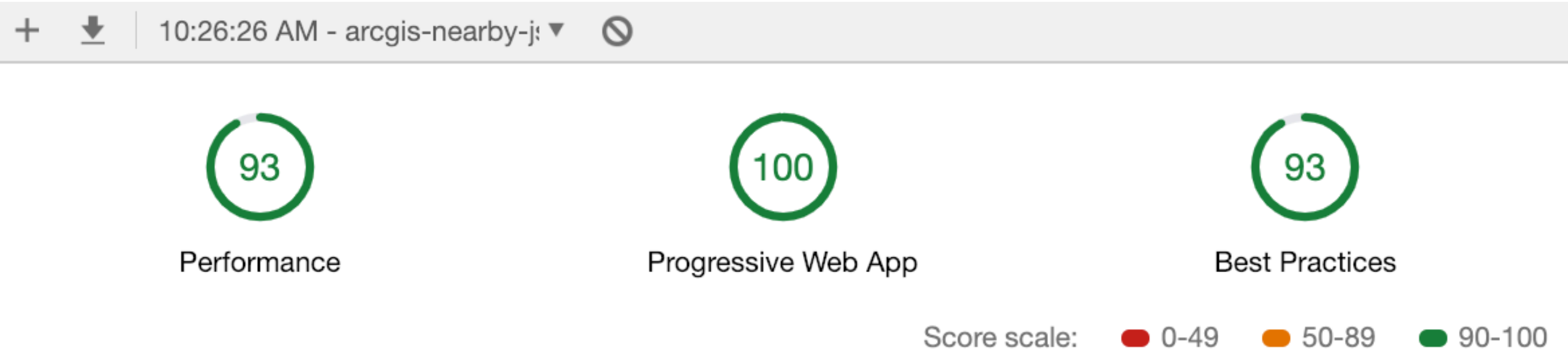
Use Web Workers

Where possible, move heavy processing off main thread

Demo 1 – Earthquake GeoJSON feed
Demo 2 – esri/core/workers

Chrome Lighthouse

- Chrome add-in



Wrap-up!

Best practices for mobile web mapping apps

Use progressive web development patterns

Eliminate user interface jerkiness/jank

Looks for small tweaks to get big performance gains

Questions:

Andy Gup
agup@esri.com
@agup
github.com/andygup


esri
THE
SCIENCE
OF
WHERE