ArcGIS API for JavaScript:

Modern Development Environments and Techniques

Andy Gup @agup

Noah Sager @NoashX

René Rubalcava @odoenet

Agenda

- Using CDN and NPM
- Advanced CDN Usage
- Building modern apps

Vanilla JS

or

Local builds

Local build bundle files

- JS 9560.7f93c300a38a1bddb76b.js
- JS 9569.2bbc1942c3847bae84be.js
- JS 9571.f66e42e342eb153a3324.js
- JS 9573.4153be83bb14fff1f481.js
- JS 9575.38daf77e472a52f0f66b.js
- JS 9609.34333a6f6f3214fa6e1a.js
- JS 9664.0a9b2d538f4fd124e4c7.js
- JS 9720.7387759300b5f1e9ba73.js
- JS 9724.8b884e2ddc60ee9804b6.js
- JS 9857.0c5b465cf47b6cbe9a47.js
- JS 9899.9f0447adcb6d63bdf193.js
- ★ favicon.ico
- <> index.html
- JS main.b53bb1c2d99a22312b8b.js
- JS polyfills.33c52343b69ce625ed65.js
- JS runtime.21d222b82264cca500e6.js
- # styles.9da6b7b594cec4e65d58.css

AMD
and
ES modules (ESM)

https://developers.argis.com/javascript

ArcGIS API for JavaScript / API Reference

Overview | Sample Code | API Reference

- esri/views
  - GroundView
  - Magnifier
  - MapView
  - SceneView
  - View
  - ViewAnimation
- 2d/layers
- 3d

MapView

AMD:
```
require(["esri/views/MapView"], (MapView) => { /* code goes here */ });
```

ESM:
```
import MapView from "@arcgis/core/views/MapView";
```

Class: `esri/views/MapView`

Inheritance: MapView > View > Accessor

Since: ArcGIS API for JavaScript 4.0

AMD

(Not a W3C standard)

```
// test-amd.js
define(() => {
    return {
        height:300,
        width: 300
    }
});
```

```
// index.html
require(['test-amd'],(test) => {
  let h = test.height);
})
```

AMD modules

```
<script src="https://js.arcgis.com/4.20/"></script>
<script>
  require([ "esri/Map", "esri/views/MapView" ],
  (Map, MapView) => {
    // Code to create the map and view will go here
  });
</script>
```

- Available since 4.0 (May 2016)

AMD CDN

Perfect for Vanilla JS apps

- Pros:
  - Easy to update
  - No installation, minimal configuration
  - Highly optimized
- Cons:
  - Requires a separate module loader
  - Use `esri-loader` for integration with frameworks and build tools

AMD Module Loader

Console Sources Elements Network Performance Memory

Preserve log Disable cache No throttling

Filter Hide data URLs All XHR JS CSS Img Media For

Name	Status	Type
4.20/	301	script / Redirect
init.js	200	script
MapView.js	200	script
webgl.js	200	script
mapViewDeps.js	200	script

AMD needs a module loader, otherwise...

```
Uncaught ReferenceError: require is not defined
```

```
Uncaught ReferenceError: define is not defined
```

AMD NPM

Use `arcgis-js-api` for API versions <= 4.18

- Pros:
  - Works with Dojo 1 and RequireJS
- Cons:
  - Still requires a module loader
  - Integration into frameworks isn't straightforward. For example, webpack requires `@arcgis/webpack-plugin*`

ESM

(ES6 or ECMAScript 2015+)

```
// test-esm.js
export const height = 300;
export const width = 300;
```

```
// index.js
import {height, width} from './test-esm.js';

let h = height;
```

ESM

Use `@arcgis/core` for API versions >= 4.19

```
import Map from '@arcgis/core/Map';

const map = new Map({
  basemap: "gray-vector"
});
```

ESM NPM

`@arcgis/core`

- Primary use case is local builds
- Pros:
  - Standardized module system
  - Works natively in modern browsers
  - Integrates well with most modern frameworks and build tools

ESM CDN

Testing and prototyping only

```
import Map from "https://js.arcgis.com/4.20/@arcgis/core/Map.js";

const map = new Map({
  basemap: "gray-vector"
});
```

- AMD is available as
  - CDN
  - NPM (Local install)
- ESM is available as:
  - CDN **
  - NPM (Local install)

Additional Resources

- github.com/Esri/jsapi-resources
- github.com/Esri/feedback-js-api-next

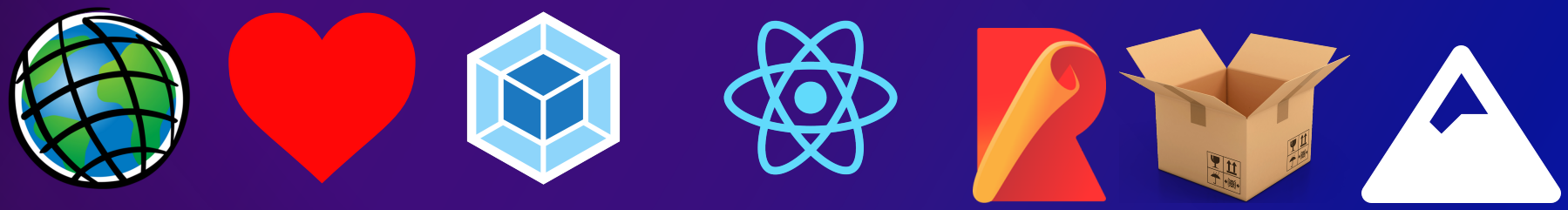
```
## install the 'next' ESM version of the API
npm i @arcgis/core@next

## install the 'next' AMD version of the API
npm i arcgis-js-api@next
```

Also, via AMD CDN:

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

Try esri-loader



Installing esri-loader


```
npm install --save esri-loader
```

Installing esri-loader

```
yarn add esri-loader
```

Using `loadModules()`

```
import { loadModules } from 'esri-loader';

loadModules([
  "esri/Map",
  "esri/views/MapView"
]).then(([Map, MapView]) => {
  // Code to create the map and view will go here
});
```

How it works

```
// calls require() once the ArcGIS script is loaded

require([
  "esri/Map",
  "esri/views/MapView"
], (Map, MapView) => {
  // Code to create the map and view will go here
});
```

Lazy loads the ArcGIS API

```
 // injects a script tag the first time
const esriConfig = await loadModules(["esri/config"])
esriConfig.useIdentity = false;

// don't worry, this won't load the API again!
const [Map, MapView] = await loadModules(
  ["esri/Map", "esri/views/MapView"]
);
```

Lazy loads the ArcGIS API

```
 // injects a script tag the first time
const esriConfig = await loadModules(["esri/config"])
esriConfig.useIdentity = false;

// don't worry, this won't load the API again!
const [Map, MapView] = await loadModules(
  ["esri/Map", "esri/views/MapView"]
);
```

Defaults to latest CDN version

esri-loader options

- Use an earlier release, even 3.x!
- Even use a later version?
- Use a local AMD build
- Lazy load CSS

Keeps ArcGIS API out of your build

Keeps ArcGIS API out of your build

- faster builds
- greater tool compatibility

What's the down side?

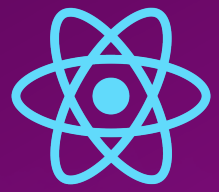
What's the down side?

- just AMD and vanilla JavaScript
- no `import` statements for ArcGIS modules
- requires pre-existing AMD build (CDN or local)

When to use esri-loader?

- Rapid prototyping, hackathons
- Your (hipster) tools have trouble with `@arcgis/core`

Demo: esri-loader with react



- Scenario: superhero themed hackathon
- Tools: React, esri-loader

ESM

- Standard Module System for JavaScript
- Better support in modern build tooling

Getting Started

```
npm i @arcgis/core
```

Usage

```
import WebMap from '@arcgis/core/WebMap';
import MapView from '@arcgis/core/MapView';
```

npm and build tools

- Benefits
  - customized local build
  - total JS between 400KB to 2MB
  - depends on your application

ESM CDN

- *Testing purposes only*
- I'm serious, listen to me

```
<script type="module">
    import ArcGISMap from "https://js.arcgis.com/4.18/@arcgis/cor
    import MapView from "https://js.arcgis.com/4.18/@arcgis/core/

    const map = new ArcGISMap({
        basemap: "topo-vector"
    });

    const view = new MapView({
        container: "viewDiv",
        map: map,
        zoom: 4,
        center: [-118, 34]
    });
</script>
```

ESM CDN

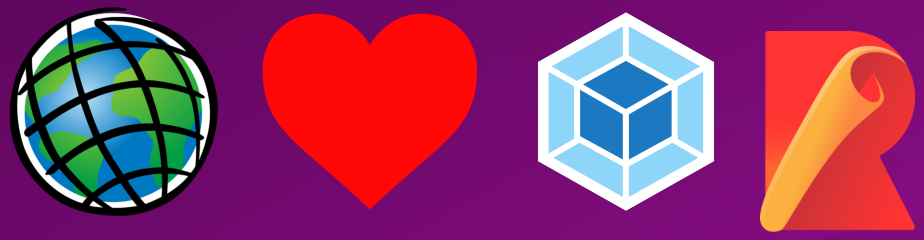
- Too many files requested for real-world use
- Convenience for prototyping
- *Please use a build tool*

ESM CDN

- Too many files requested for real-world use
- Convenience for prototyping
- *Please use a build tool*

  w/ defaults

@arcgis/core



ArcGIS API is different

- powerful library with large footprint
- uses dynamic module loading & web workers

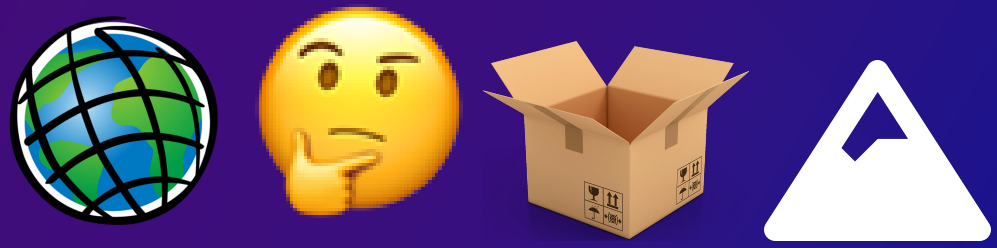
ArcGIS API is different

- powerful library with large footprint
- uses dynamic module loading & web workers
- can slow your build; or not work w/ defaults

Dev Environment

- Easiest way to get started... use VS Code
- Review samples on github.
- *I* like vitejs
  - minimal, to zero config
  - that's it, so easy you feel guilty

Is your bundler smarter than you?



Conclusion

Consuming the ArcGIS API is easier than ever!

esri

THE SCIENCE OF WHERE