JavaScript *for Geographers*

Patrick Arlt & Allison Davis

Slides: http://bit.ly/2T4zaKJ

2019 ESRI DEVELOPER SUMMIT
Palm Springs, CA

Agenda

1. Fundamentals
2. Patterns
3. The JavaScript Ecosystem
4. Resources and Ways to Keep Learning

What is this talk?

An attempt to get you *started* down the path of learning JavaScript.

JS can be overwhelming, you're more equipped than you think!

- Scripted with ArcPy?
- Scripted with Python?
- Configured an app?
- Used Arcade?
- Used Model Builder?

Don't Feel Overwhelmed

You don't need fancy tools and a huge body of knowledge to do useful things with JavaScript.

Fundamentals

variables

```
var dog;
let nifty;
const notGonnaChange;

> undefined

var dogName = 'spot';
var age = 21;
var canBark = true;
// value type is **not** explicitly declared
```

arithmetic operators

```
(age / 7) // 3

5 + 5 // 10

3 - 2 // 1


3 * 2 // 6

12 % 5 // 2 (modulus)

age++ // 22

age-- // 20

'high' + 'five' // 'highfive'
```

comparison & logical operators

```
3 === 3   // true
3 === '3' // false

'dog' != 'cat' // true


3 > 2 // true
3 >= 2 // true

// logical 'and'
true && anotherTruthy
> true

// 'or'
true || somethingFalsy
> true
```

functions

```
function dogYears(age) {
  return age * 7;
}

dogYears(3);
> 21
```

arrow functions

```
age => {
  return age * 7
}

age => age * 7
// these are the same!
```

arrays

```
var dogs = ['Spot', 'Lassie'];

dogs[0] // 'Spot'

dogs.push('Fido');


dogs.length // 3

dogs.forEach(dog => {
  console.log(dog);
});
> undefined

dogs.map(dog => dog.toUpperCase());
> ['SPOT', 'LASSIE', 'FIDO']
```

objects

```
let dog = {
  age: 7,
  canBark: true,
  _sshhh: 'top secret',

  ageInDogYears: function(age) {
    return age * 7;
  }
}

> Object {age: 7, canBark: true, _sshhh: 'top secret', ageInDogYears: ageInDogYears() }

dog.ageInDogYears(dog.age);
> 49
```

classes

```
class Dog {
  constructor(name) {
    this.name = name;
  }
}
let myDog = new Dog('Ginsburg'); // Object { name: 'Ginsburg'}

class Dalmation extends Dog {
  constructor(name) {
    super(name); // super calls the parent class' constructor
    this.breed = 'Dalmation';
  }
}
let myOtherDog = new Dalmation('Spot'); // Object { breed: 'Dalmation', name: 'Spot'}
```

JavaScript Patterns

JavaScript is *Asynchronous*

- JavaScript is *single threaded*
- Only does 1 thing at a time
- Lots of things might happen at once
- This is the "Event Loop"

JavaScript Event Loop

1. Executes one function at a time
2. Run the entire function
3. Start the next function

Demo

Callbacks

```
<button id="button">Click Me!</button>
```

```
let button = document.getElementById('button');

button.addEventListener('click', function () {
  console.log('The button was clicked');
});
```

Callback are functions that are run *later* when things happen.

Promises

```
let user = fetch('https://randomuser.me/api/')
  .then(processResponse)
  .then(doSomethingWithUser)
  .catch(anyErrors);

function processResponse (response) {
  return response.json();
}

function doSomethingWithUser (user) {
  console.log(user); // prints a bunch of user info
}

function anyErrors (error) {
  console.error('what have you done!', error);
}
```

Promises represent a future value that will be "resolved".

I `Promise` to be a useful value in the future.

Demo

Function Scope

```
var prefix = 'Hello';

function go () {
  var suffix = "World!"
  console.log(prefix + " " + suffix); // "Hello World"
}

go();

console.log(suffix); // undefined
```

Functions remember the variables around them, this is referred to as "lexical scope".

The DOM

- select elements (HTML tags)
- listen for events
- change elements

```
console.log(value); // prints value to js console
debugger; // pauses application code
```

Demo

Sharing JavaScript

As applications grow, divide code into different files to stay organized. For small apps, you can just use `<script>` tags.

```
<!-- Add script tags at the bottom of index.html before </body>-->
<script src="/alert.js"></script>
<script src="/form.js"></script>
```

```
// alert.js file
var alert = "alert!"
```

```
// form.js file
var form = document.getElementById("form");

form.addEventListener("submit", (event) => {
console.log(alert); // this will work thanks to global scope
});
```

View full example

JavaScript Modules

```
import { something } from 'some-module';
```

This is the future: as you learn JavaScript, you will encounter this more often.

Demo

AMD Modules (JS API)

```
require([
  "esri/Map",
  "esri/views/MapView",

], function (Map, MapView) {
  // Map and MapView have been loaded!
});
```

`require` is a fancy way of adding `<script>` tags to load code on demand.

Demo

Putting the pieces together

- Chaining Promises JS API Sample

The JavaScript Ecosystem

The JavaScript Language

JavaScript (the language) updates every year.

2015 had LOADS of new features and established most of modern JavaScript.

Build tools, bundlers, and frameworks

- Modules - formats for splitting up and sharing code
- Compilers - Transform JS > JS, add features to JS
- Bundlers - Combine modules and other assets
- Frameworks - Architecture and structure for large apps/teams

Build tools, bundlers, and frameworks

- Modules - AMD , JS Modules
- Compilers - TypeScript
- Bundlers - WebPack
- Frameworks - React , Angular , Vue , Ember , Dojo

Node JS and NPM

- Node JS - Run JavaScript on a server or desktop computer. Build web servers, APIs and CLI tools
- NPM - Package manager and distribution system for JS Modules. Analogus to Pip or Conda in Python.

Learn Node JS at NodeSchool

Some people have "opinions" about JavaScript

Many JavaScript developers have **very** strong opinions about JavaScript.

- Which framework you *should* use
- Which build tool is *the best*
- The *only* way to do _ is…

JavaScript Fatigue

Look, it's easy. Code everything in Typescript. All modules that use Fetch compile them to target ES6, transpile them with Babel on a stage-3 preset, and load them with SystemJS. If you don't have Fetch, polyfill it, or use Bluebird, Request or Axios, and handle all your promises with await.

We have very different definitions of easy.

the JavaScript ecosystem

You don't know what you don't know.

and that is great.

Fight JavaScript Fatigue

- The JS API is MORE then enough for simple mapping apps
- Many configurable apps and storymaps are built without frameworks or excessive tools
- Add tools when you **KNOW** you will benefit from using them
- Too many tools === Lots of complexity to manage
- Don't touch tools until you feel limited by your current approach

Development tools

- Set up your local dev environment: Do I have a web server running?
- Prototype with CodePen, JSBin or StackBlitz
- Visual Studio Code
- Chrome Developer Tools
- ArcGIS JS CLI

Keep learning

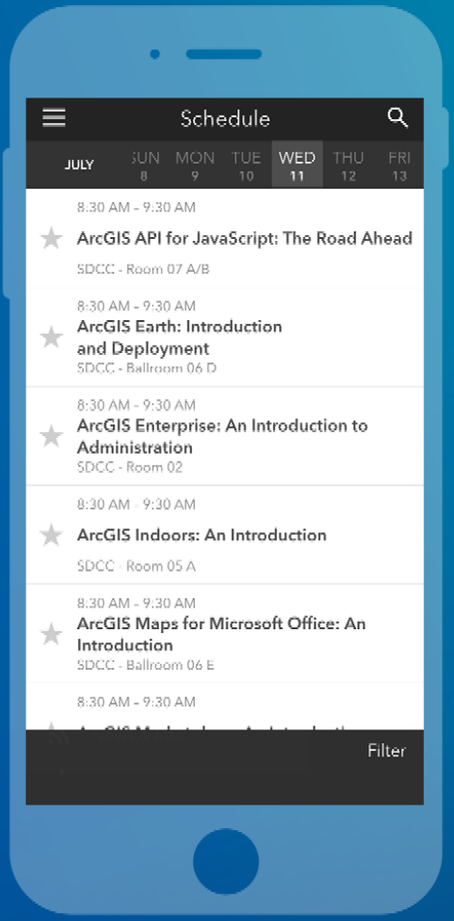
- ArcGIS DevLabs
- MDN: Learn web development
- MDN: JavaScript
- Eloquent JavaScript
- You Don't Know JS
- JavaScript 30
- NodeSchool
- Command Line Power User
- Front End Handbook

Slides at http://bit.ly/2T4zaKJ

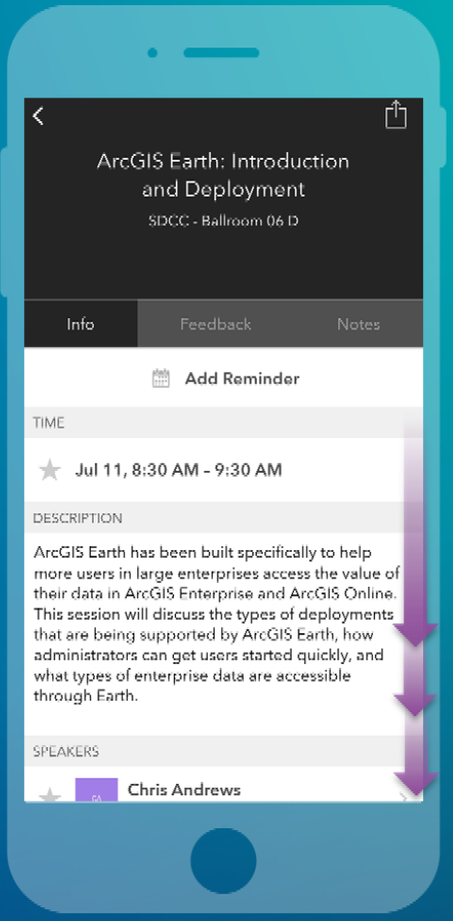
Please Take Our Survey on the App

Download the Esri Events app and find your event

Select the session you attended



Scroll down to find the feedback section



Complete answers and select “Submit”