esri Deutschland
THE SCIENCE OF WHERE

esri Suisse
THE SCIENCE OF WHERE

Topomat

Julian Kissling

Agenda

- **Typescript**
  - What is Typescript?
  - Demo
  - Migration
- **Custom Widgets**
  - About Widgets
  - Demo
  - Migration
- **Tooling**
  - Modules & CDN
  - Webpack

Schedule

09:00 Start

…

…

…

…

…

…

Migrating to Typescript

What is Typescript?

- Superset of JavaScript -> Transpiles to JavaScript
- Compatible with existing JS
- ESNext features (import, =>, res/spread, async/await, …)
- Types

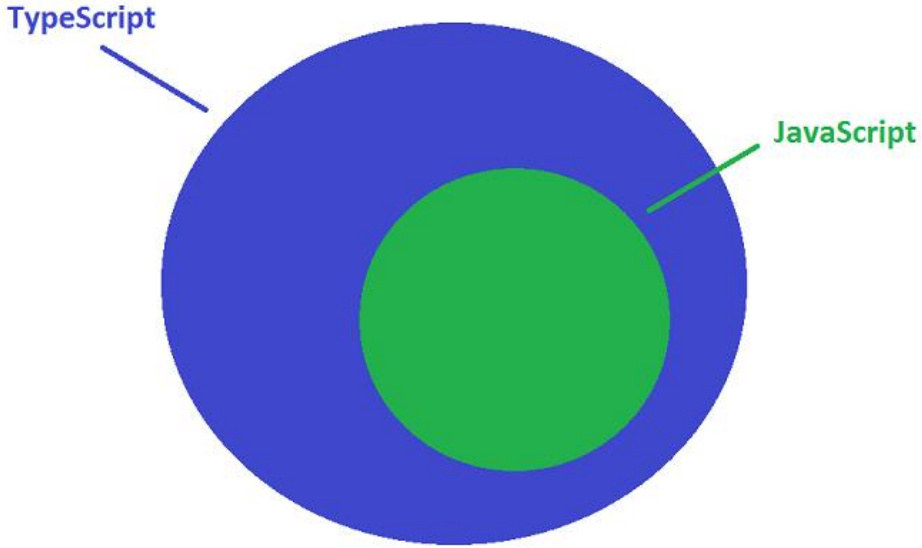


*figure not drawn to scale

Why should you use TypeScript?

- Easier for multiple people to work on
- Easier to refactor
- Can helpt prevent technical debt
- …

Benefits of TypeScript

Primitive (Basic) Types

```
type Foo = number;

const foo: Foo = 8;
const bar: string = "Lorem ipsum";

// Here be dragons
const waldo: any = {
  doStuff: (things: any) => something
};
```

Benefits of TypeScript

Type Inference

```
let foo = 8; // number type is inferred

foo = 12; // Ok

foo = "12"; // Error!
```

Benefits of TypeScript

Define contracts between parts of an application

```
type Foo = number;
type Bar = string;

interface Foobar {
  foo: Foo,
  bar: Bar
}

const baz: Foobar = { foo: 8, bar: "Lorem ipsum" }; // Ok
const qux: Foobar = { foo: "12", bar: "Lorem ipsum" } // Error!
```

Benefits of TypeScript

Classes

```
class Waldo {
  public doStuff(things: Foobar): Foobar { ... }

  private iterateNumber(num: number) {
    return num + 1;
  }

  private addExclamationPoint(str: string) {
    return `${str}!`;
  }
}

const testWaldo = new Waldo(); // Create a Waldo instance
testWaldo.iterateNumber(2); // Error!
```

Benefits of TypeScript

Extensions

- Interfaces can extend other interfaces or classes
- Classes can extend other classes and implement interfaces

```
interface Point {
  x: number;
  y: number;
}

interface Point3d extends Point { z: number; }

class MyPoint implements Point3d {
  x = 0;
  y = 0;
  z = 0;
}

class My4dPoint extends MyPoint {
  time = Date.now();
}
```

Benefits of TypeScript

Union types

- **Type something as one of multiple choices of a type**

```
// Set a size as either a number, of a string like "1px",
// "2em" etc
function setSize(v: number | string) {
  // ...
}
```

Benefits of TypeScript

Type guards

- Type guards allow TS to infer a specific type when a value may take multiple types (union)
- Types are narrowed to a more specific set by type guards
- Built in type guards like typeof, instanceof or tagged unions

```
function foo(v: number | string) {
  if (typeof v === "number") {
    // TS infers that v: number
    return v + 1;
  }
  else {
    // TS infers that v: string
    return `${v} + 1`;
  }
}
```

Benefits of TypeScript

Generics

- **"Generalizes" types over type parameters**

```
class List<T> {
  constructor(private data?: T[]) {
  }

  find(f: (item: T) => boolean): T {
    // ...
  }
}

// Fails
const list = new List<number>(["1", "2"]);

// OK
const list = new List<number>([1, 2]);

// TS infers v to be of type number
list.find(v => v > 1);
```

Benefits of TypeScript

Promises & Async/Await

- "Generalizes" types over type parameters

promises

```
function demoTheater() {
  getQuestion()
    .then(function question() {
      console.log(question);
      return "answer";
    })
}

demoTheater();
```

async / await

```
async function demoTheater() {
  console.log(await getQuestion());
  return "answer";
}

demoTheater();
```

Some Ressources

- https://www.typescriptlang.org/
- https://developers.arcgis.com/javascript/latest/typescript-setup/
- https://odoe.github.io/ds2021-slides/using-typescript/index.html

Typescript demo

1. **Install Typescript**

   ```
   npm install typescript
   ```

2. **Init a new project (tsconfig.json)**

   ```
   npx tsc –init
   ```

3. **Run Compiler**

   ```
   npx tsc --watch -p .
   ```

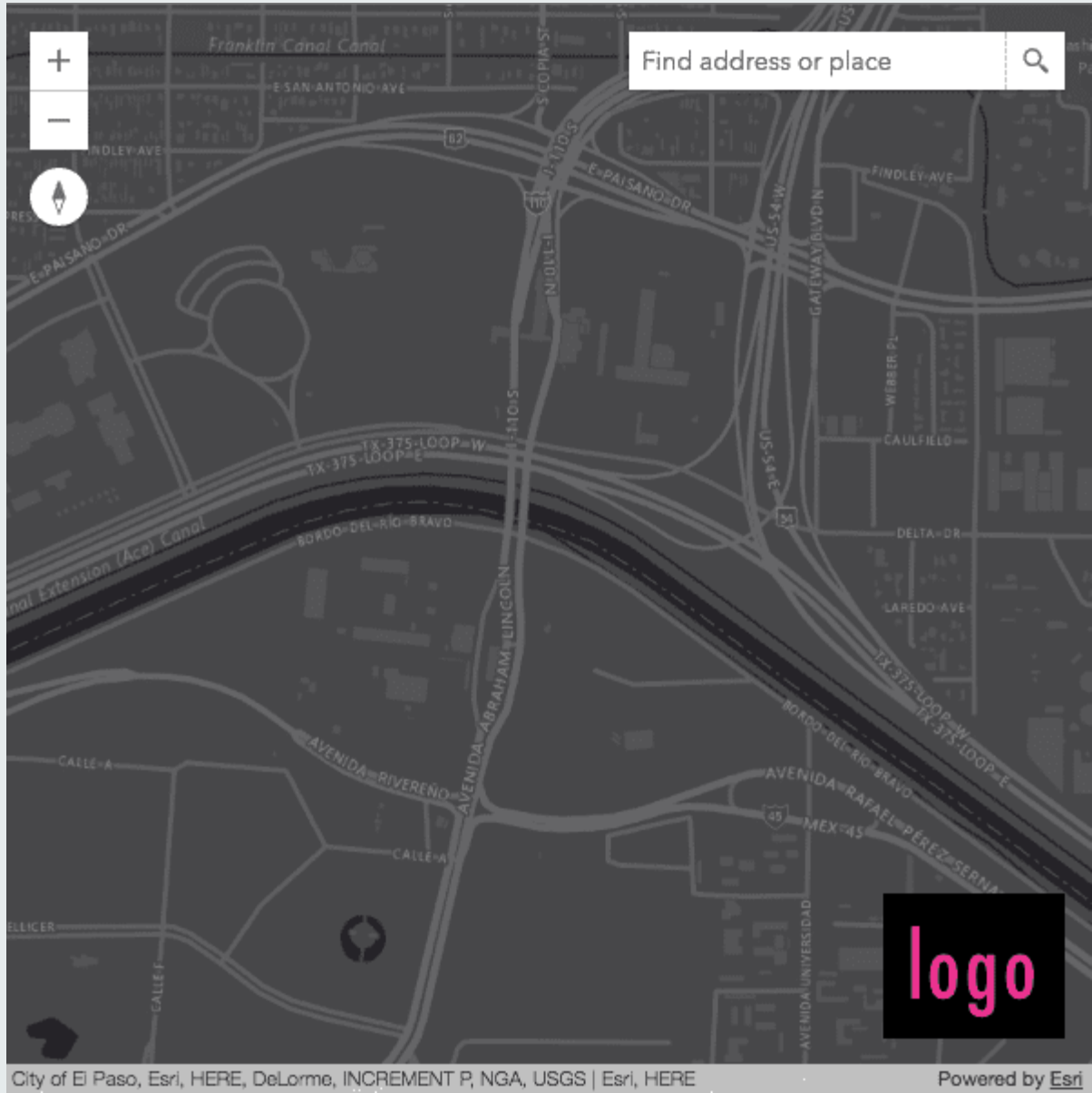
Migrate Sample Application

Custom Widgets

Good to Know: View UI

- **A View provides an API for placing Widgets and DOM elements**

```
view.ui.add(search, "top-right");
view.ui.add(logo, "bottom-right");
```



About Widgets

What?

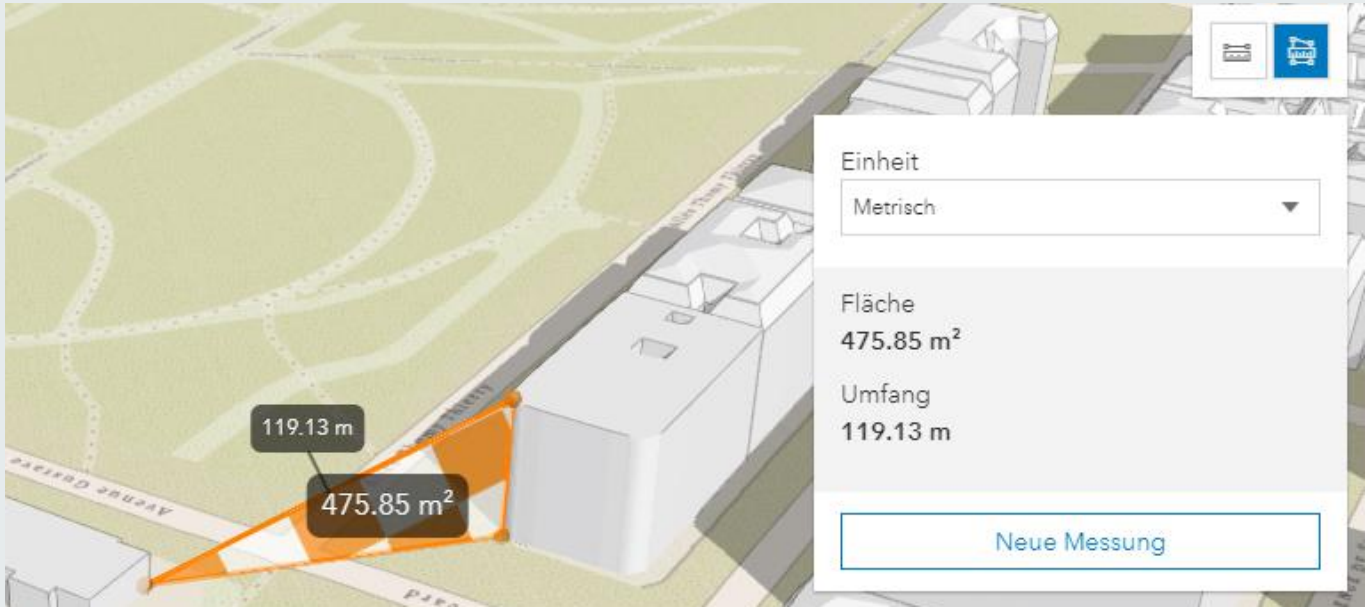
- Encapsulated UI components
- Cohesive (integrated, unified)
- Single-purpose pieces of functionality

Why?

- Reusable
- Interchangable

How?

- Extend `esri/Widgets/Widget`



esri/widgets/Widget

- Base widget class (View)
- Extends esri/core/Accessor
  - Properties
  - Watching properties
- Lifecycle

Accessor

- JavaScript API foundation
- Aims to make developing classes easy
  - Getter/Setter
  - Watch
- Consistent developer experience

```
// unified object constructor
const me = new Person({ name: "Franco", age: 33 });

// watch for changes to `age`
me.watch("age", singHappyBirthday);
```

Lifecycle

constructor (params)

- Widget initially created
- Get, Set and watch properties

postInitialize()

- Before UI is rendered

render() (Required)

- Used to render UI
- Returns the Widgets Markup
- Reacts to state Changes
- Uses JSX (VDOM)

destroy()

- Release the widgets instance

Demo

1. Setup (optional)
2. Create a custom class - Extend the Accessor
3. Create a Widget – View only
4. Creat a Widget – Using a ViewModel

Migrate Widgets

Tooling

Getting the API

- **The API can be accessed as:**
  - **AMD Modules via CDN (our demos before)**
  - **AMD via npm**
  - **ES modules via npm**
  - **ES modules via CDN (we won’t have a look at this)**

AMD via CDN

- **Fast download and highly optimized caching for the API modules**
- **No installation or configuration**
- **Easy to update applications to the next API version**

```
<link rel="stylesheet" href="https://js.arcgis.com/4.18/esri/css/main.css">
<script src="https://js.arcgis.com/4.18/"></script>
```

AMD via npm (Local Build or esri-loader)

- **If version 4.17 or earlier of the API with most frameworks and build tools.**
- **Works with Dojo 1 and RequireJS.**

Disadvantages:

- Requires helper libraries such as esri-loader and arcgis-webpack-plugin when working with modern frameworks and build tools.
- Modules loaded with esri-loader are not bundled with the build, they are requested from the CDN at runtime.
- The arcgis-webpack-plugin needs to be configured and it may require additional libraries to extend webpack, for example when using Angular 9+.

```
npm install esri-loader

npm install arcgis-js-api

npm install @arcgis/webpack-plugin
```

ES modules via npm

- **Standardized**
- **No helper or module loader required**
- **Seamless integration with most modern frameworks**
- **Provides server-side capabilities for node.js deployments, for example, geometry engine**

Disadvantages:

- Updates require installing a new version.

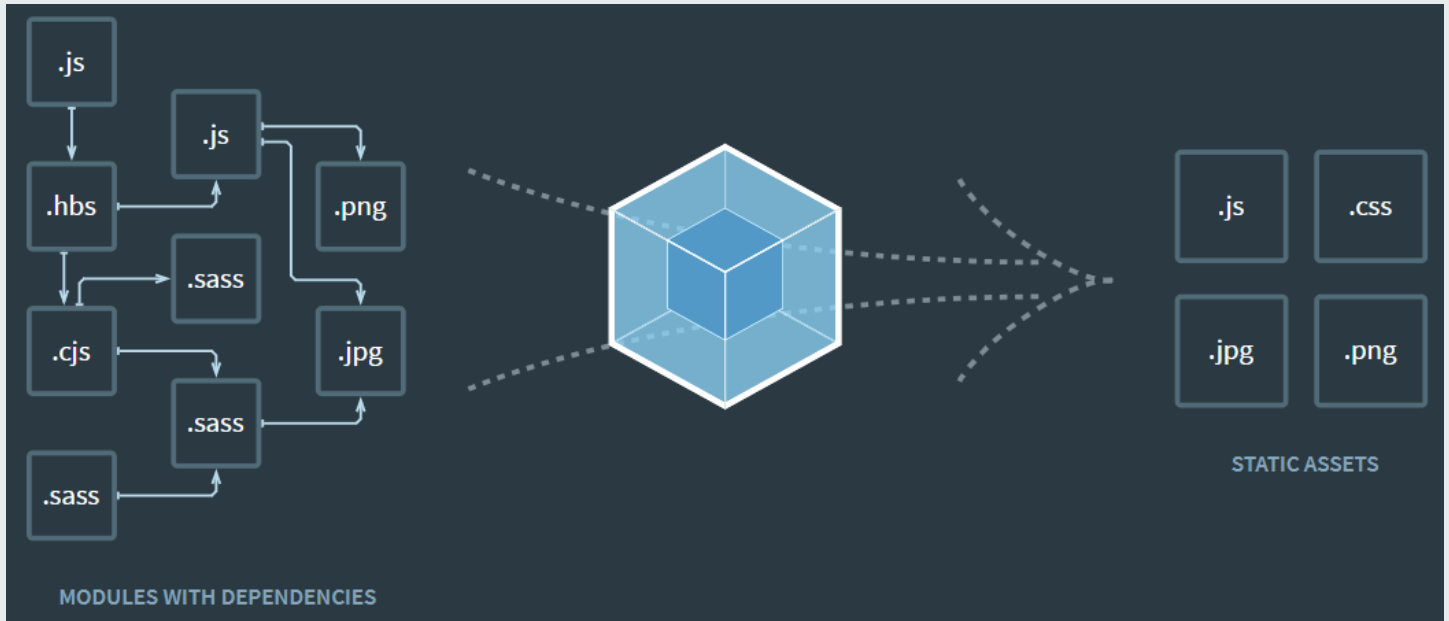
Comparison

	CDN (AMD)	ESM local build	AMD local build
No installation, configuration or local build required	X		
Fast download performance via CDN cache	X		
Easy installation via npm		X	(X)
Seamless integration with most modern frameworks and build tools		X	
Using API version 4.17 or earlier with a framework or build tools			X
Using Dojo 1 or RequireJS			X

Webpack

What is it?

- **Static module bundler**
- **Creates an internal dependency graph**
- **Open source**
- **Extendable**
- **Sometimes a bit tricky to use ;-)**



Source: https://webpack.js.org/

Entry

- **Defines where webpack should begin – Your «first» or «main» module**
- **Can be more than one (e.g. TS and SASS)**
- **Docs**

```
entry: {
  index: ["./src/css/main.scss","./src/index.ts"]
},
```

Output

- **Tell webpack where to emit the output to**
  - **Usually ./dist**
- **Docs**

```
output: {
  filename: "[name].[chunkhash].js",
  publicPath: "",
},
```

Loaders

- **Loaders enable webpack to load any types of files**
  - **By default, webpack only supports JS and JSON files**
- **This gives us a lot of flexibility, because we can use webpack to handle most of our ressources**
- **Docs**

```
module: {
  noParse: /assets\/libs\/widgets/,
  rules: [
    {
      test: /\.html$/i,
      loader: 'raw-loader',
      options: {
        esModule: false
      }
    },
    {
      test: /\.tsx?$/,
      use: [{
        loader: "ts-loader",
        options: {
          transpileOnly: true
        }
      }]
    }
  ]
}
```

Plugins

- Extend the functionality of webpack (cleanup, copy assets, …)
- Need to be loaded seperately
- @arcgis/webpack-plugin ;-)
- Docs

```
plugins: [
  new CleanWebpackPlugin({
    cleanAfterEveryBuildPatterns: ['dist']
  }),

  new ArcGISPlugin({
    locales: ['de', 'fr', 'en']
  })
]
```

Mode

- Enable/disable built in optimization
  - Production build
  - Development mode
- Can be passed as command line param

```
mode: 'production',
```

Webpack and CDN

Demo

- Turn the sample application with the migrated widgets into a webpack app
- Include the „old“ widgets as static library
  - Write some definition files for them

Webpack and Sass

Demo

- Include Sass into your webpack project
- Control which styles are imported from the JS API

Local Builds

Demo

- **Create a webpack application which includes the AMD version of the JS API**

Building apps with ES Modules

Rapid Prototyping

Sample (Not for production)

```
import ArcGISMap from "https://js.arcgis.com/4.18/@arcgis/core/Map.js";
import MapView from "https://js.arcgis.com/4.18/@arcgis/core/views/MapView.js";

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

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

Create a local build

Assets

- The API comes with a lot assets, which need to be included in the final product. Use the @arcgis/webpack-plugin or use a tool like ncp (cross-platform copy tool)

```
"scripts": {
    "start": "npm run copy && webpack serve --open --mode development",
    "build": "npm run copy && webpack --mode production",
    "copy": "npx ncp ./node_modules/@arcgis/core/assets ./src/assets"
  },
```

Migrate from ArcGIS Webpack to ES Modules

1. Instead of ‘arcgis-js-api’ install ‘@arcgis/core’ npm package
2. Change your imports from:

```
import Map from 'esri/Map';
import MapView from 'esri/views/MapView';
```

To:

```
// ES modules
import Map from '@arcgis/core/Map';
import MapView from '@arcgis/core/views/MapView';
```

3. Make sure the assets get copied
4. Don’t forget to tweak your tsconfig.json ;-)

Demo

- **Try to migrate the App created in 07_webpack_local_build to ES modules**

Esri in Germany and Switzerland

Company

As distributors, Esri Deutschland GmbH and Esri Schweiz AG sell Esri Inc. products. We fully support our users in every way - with the combined experience and expertise of 330 employees, we provide consulting and implementation services as well as training courses and technical support - since 1979.

esri Deutschland
THE SCIENCE OF WHERE

Esri Deutschland GmbH Kranzberg
Hamburg Office
Leipzig Office
Berlin Office
Hannover Office
Münster Office
Bonn Office
Cologne Office

esri Suisse
THE SCIENCE OF WHERE

Esri Schweiz AG Zürich
Nyon Office

esri Deutschland
THE SCIENCE OF WHERE

esri Suisse
THE SCIENCE OF WHERE