esri

Using TypeScript

with the ArcGIS API for JavaScript

Noah Sager | René Rubalcava

@Noashx

@odoenet

2021 ESRI
DEVELOPER SUMMIT

Agenda

- What is TypeScript?
- Why use TypeScript?
- Setup and First steps
- Live Action Demo
- Where can I get more info?

What is TypeScript?

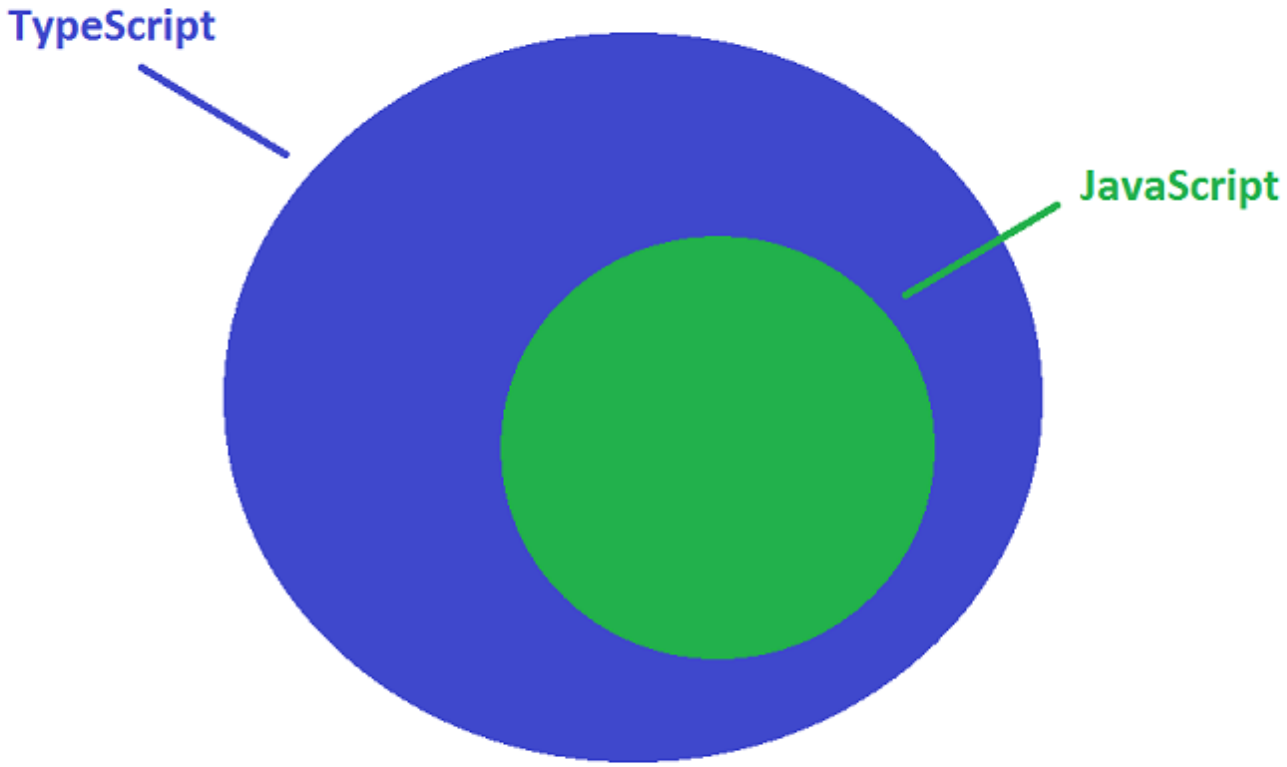


*figure not drawn to scale

Where do I begin?

Developer Setup

ArcGIS Developers | Documentation | Features | Pricing | Support | Search | Sign In

ArcGIS API for JavaScript | Overview | Sample Code | API Reference | Showcase | Blog

Find page...

- Overview
- Key Features
- Get started
- Install and setup
- Release notes
- Get the API
- Quick start
- FAQ
- Community
- Tutorials
- Core concepts
- Data visualization
- Building your UI
- Working with ArcGIS Online and Enterprise
- Developer tooling
  - Introduction to tooling
  - Build with ES modules
  - Build with AMD modules
  - Setting up TypeScript

TypeScript - Setting up your development environment

In order to take advantage of the Accessor and custom widget development, you will want to first learn how to set up your development environment to use TypeScript.

This guide provides some basic steps you can use to set up your TypeScript development environment. *This is not a TypeScript tutorial*. It is highly recommended that you review some of the tutorial material available.

Prerequisites

The recommended way to install TypeScript is via Node and npm. The package manager `npm` is used to install various libraries and tools.

Make sure to install TypeScript globally using `npm install -g typescript`. This will install a command line tool called `tsc` that will be used to compile your TypeScript code. You can then check you have TypeScript installed by using `tsc -v` and it should tell you the version of TypeScript you have installed.

As of version 4.16, the `tslib` runtime library that contains all of the TypeScript helper functions must be installed as well. This can be installed via `npm install --save tslib`. For additional information on this, please refer to the tslib documentation.

> *The demo application for this guide can be found here.*

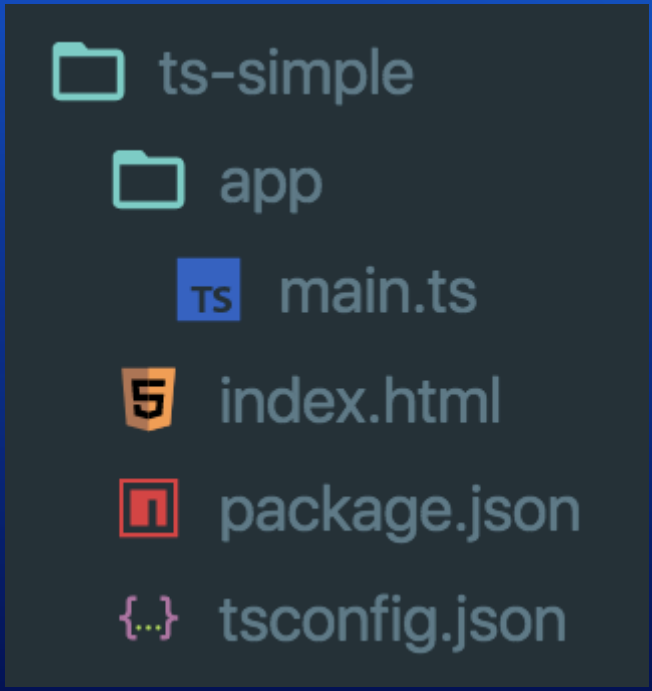
Folder structure

TypeScript - Setting up your development environment

Was this page helpful?

Yes No

Developer Setup



Benefits of TypeScript

TypeScript

JavaScript that scales.

- Easier for multiple people to work on
- Easier to refactor
- Easier to test
- Can help prevent technical debt

Why use TypeScript?

TypeScript adds *type* support to JavaScript

```
const url = "https://sampleserver6.arcgisonline.com/arcgis/rest/services/Notes/FeatureServer/0";

function createFeatureLayer(URL: string, legend: boolean) {
  const featureLayer = new FeatureLayer({
    url: URL,
    legendEnabled: legend
  });
  map.add(featureLayer);
}

createFeatureLayer(url, true);
```

Why use TypeScript?

Enhanced IDE support

```
const url = 12345;

function createFeatureLayer(URL: string, legend: boolean) {
  const featureLayer = new FeatureLayer({
    url: URL,
    legendEnabled: legend
  });
  map.add(featureLayer);
}

createFeatureLayer(url, true);
```

```
const url: 12345
Argument of type '12345' is not assignable to parameter of type 'string'. ts(2345)
Quick Fix...   Peek Problem
```

Why use TypeScript?

Makes use of the latest JavaScript features

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();
```

Why use TypeScript?

Makes use of the latest JavaScript features

```
async function importModule() {
  const esriModule = './utils.js';
  const module = await import(esriModule);
  module.doStuff();
}

importModule();
```

Setup and First steps

1. The recommended way to install TypeScript is via `node` and `npm`.
2. Make sure to install TypeScript globally:

```
npm install -g typescript
```

3. Install the ArcGIS API for JavaScript Typings:

```
npm install --save @types/arcgis-js-api
```

Essentials

TypeScript

```
npm install --save-dev typescript
```

JS API 4.x typings

```
npm install --save-dev @types/arcgis-js-api
```

JS API 3.x typings

```
npm install --save-dev @types/arcgis-js-api@3
```

Demo: Build a TypeScript app from scratch


WHERE

Imports

- Can use AMD or ESM build
- *Hint:* use the ESM build

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

Auto-cast

- Due to nature of types, auto-cast does not type-check
  - `get` and `set` must have the same type
- Auto-casting is supported in constructor signatures only
  - Still helps in lots of cases
  - For setting properties, need to import the relevant modules

Typing improvements

- Use of generics where possible `Collection<T>`
- Strictly type events (`mapView.on("mouse-wheel", ...)`))
- "Advanced" auto-casts like colors (`"red"`), screen sizes (`"5px"`) and basemaps `"streets"`

Demo Steps:

```
mkdir ts-demo && cd ts-demo
mkdir src && touch src/index.ts
npm init --yes && tsc --init
npm i @arcgis/core rollup ## other rollup plugins
```

index.html

```
<body>
  <div id="viewDiv"></div>
  <script src="index.js" type="module"></script>
</body>
```

tsconfig.json

```
{
    "compilerOptions": {
        "lib": ["ES2019", "DOM"],
        "sourceMap": true,
        "target": "ES2019",
        "noImplicitAny": true,
        "suppressImplicitAnyIndexErrors": true,
        "moduleResolution": "node"
    },
    "include": ["src/**/*"]
}
```

CSS

```
<style>
  html,
  body,
  #viewDiv {
    padding: 0;
    margin: 0;
    height: 100%;
    width: 100%;
  }
</style>
```

src/index.ts

imports

```
import WebMap from "@arcgis/core/WebMap";
import MapView from "@arcgis/core/views/MapView";
import LayerList from "@arcgis/core/widgets/LayerList";
```

src/index.ts

WebMap and MapView

```
const map = new WebMap({
  portalItem: {
    id: 'd5dda743788a4b0688fe48f43ae7beb9'
  }
});

// Add the map to a MapView
const view = new MapView({
  container: 'viewDiv',
  map
});
```

src/index.ts

LayerList

```
// Add a legend instance to the panel of a
// ListItem in a LayerList instance
const layerList = new LayerList({
  view,
  listItemCreatedFunction: (event: { item: __esri.ListItem }) =>
    const item = event.item;
    if (item.layer.type != 'group') {
      item.panel = {
        content: 'legend',
        open: true
      } as __esri.ListItemPanel;
    }
  }
});
view.ui.add(layerList, 'top-right');
```

Demo the build

**Tip: Hide .js and .jsmap files **

- Reduce clutter
- VSCode: Add below to user preferences in files.exclude

```
"**/*.js.map": true,
"**/*.js": {
    "when": "$(basename).ts"
  }
```

Tip: Debugging with source maps

- Enable source maps in browser dev tools
- Set breakpoints in .ts instead of .js

main.ts main.js ×

```
1  var __importDefault = (this && this.__importDefault) || f
2      return (mod && mod.__esModule) ? mod : { "default": m
3  };
4  define(["require", "exports", "esri/WebMap", "esri/views/
5      "use strict";
6      Object.defineProperty(exports, "__esModule", { value:
7      WebMap_1 = __importDefault(WebMap_1);
8      MapView_1 = __importDefault(MapView_1);
9      LayerList_1 = __importDefault(LayerList_1);
10     var map = new WebMap_1.default({
11         portalItem: {
12             id: "d5dda743788a4b0688fe48f43ae7beb9"
13         }
14     });
15     // Add the map to a MapView
```

Where can I get more info?

- SDK Documentation
- Esri-related training and webinars
- ArcGIS Blogs
- GeoNet, StackExchange, etc.

Using TypeScript with the ArcGIS API for JavaScript

Mapping and Visualization

October 16, 2018

Noah Sager,
Rene Rubalcava

esri

THE SCIENCE OF WHERE

Please provide your feedback for this session by clicking on the session survey link directly below the video.