**Adding Polyfills in TypeScript App**

A polyfill is code that fills in functionality missing from the older browser. For example, Internet Explorer does not have Promise like other modern browsers. In this case, we can use a promise polyfill so that the code that uses Promise can work in IE.

There are two ways of adding polyfills.

1. Adding polyfill script from [Polyfill.io](https://polyfill.io/)

It adds ployfills at the page level. This is preferred because it decouples polyfills from the bundles. It makes the bundle size smaller. When we have multiple bundles and each is having its own polyfills, we need to add duplicate polyfills. The worst case scenario is that a polyfill from one bundle can override another polyfill from the other bundls. In this case, polyfill acts strangely and it is hard to debug like [this example](https://www.mydatahack.com/troubleshooting-promise-polyfill-not-working-as-expected).

<pre>

<script src="https://polyfill.io/v3/polyfill.min.js?Promise%2CPromise.prototype.finally"></script>

</pre>

1. Install polyfills and add them in the script tag

There are so many polyfills out there. My recommendation is [core-js](https://www.npmjs.com/package/core-js). It includes a bunch of polyfills and you can pick and choose. Core-js may not be enough if you are polyfilling functions on HTML elements such as scrollHeight or fetch. We can also install individual polyfills.

Here is the example.

<pre>

npm i --save-dev whatwg-fetch element-scroll-polyfill core-js @types/core-js @types/whatwg-fetch

</pre>

Once those polyfills are installed, we can add import statements at the top of the entry file such as main.ts.

<pre>

import 'core-js/features/promise';

import 'core-js/features/array/for-each';

import 'core-js/features/array/from';

import 'core-js/features/object/assign';

import 'core-js/features/object/values';

import 'core-js/features/string/includes';

import 'core-js/es/map';

import 'core-js/es/set';

import 'element-scroll-polyfill';

import 'whatwg-fetch';

</pre>

If you use the latest version of core-js, promise includes finally. If you decide to use promise-polyfill, you need to include finally. The same goes for older version of core-js, which is babel-core.

<pre>

npm i --save-dev promise @types/promise promise-polyfill @types/promise-polyfill @types/promise.prototype.finally promise.prototype.finally

</pre>

For TypeScript, the path is slightly different from the documentation. Adding these lines at the top of the entry file works fine.

<pre>

import 'promise-polyfill/src/polyfill';

import 'promise-polyfill/src/finally';

</pre>

On the final noe, with gulp and rollup, we need rollup-plugin-commonjs to transpile require used in core-js if you are getting require error.

<pre>

const commonjs = require('rollup-plugin-commonjs');

…

plugins: [

rollupTypescript({

cacheRoot: '.rollupcache',

tsconfigOverride: {

compilerOptions: {

removeComments: true,

}

}

}),

rollupNodeResolve({

customResolveOptions: {

moduleDirectory: 'node\_modules'

}

}),

commonjs(),

rollupUglify({

compress: {

drop\_console: false

}

})

]

</pre>