academeez

@angular/universal - SSR

8. shimming

shim

academeez

- It’s better to understand what shim is with an example…
- Let’s say we have an API that is only available in the browser and not in node.
- Keeping our code isomorphic if we run the same code on node our app will crash.
- We can create a similar API for node.
- A shim allows us expend our API so different environments can work in a similar way.

```
window.setTimeout(() => {
  console.log('this code will fail in node')
}, 1000);
```

```
(<any>global).window = {
  setTimeout: global.setTimeout
}

window.setTimeout(() => {
  console.log('this code can work with shim')
}, 1000);
```

polyfill

academeez

- A polyfill is a shim
- It implements an API which will be supported in the future
- For example if we are using Javascript ES8 and we are using a new javascript feature which is not supported yet in IE, we can create a polyfill which will implement this feature in IE browsers.
- A polyfill is a temporary solution that allows our code to run in different browsers.
- Usually you will not write it yourself since the community already created the popular polyfills.
- In angular the polyfills are arranged in the file **polyfills.ts** where you will have to comment our polyfills if you need to support a certain browser
- The comments in that file will help you decide which poyfill you need.

Shim for better isomorphic code

academeez

- branching your code with PLATFORM_ID will create a harder to maintain SSR environment
- It’s recommended to use shim and not branching
- if we have a code which is not supported for a platform, we can create a shim for that unsupported code.
- Let’s give a few examples which will help you create shims in your angular application and by doing so create isomorphic code.

shim service

academeez

- This case is useful when we use a service that works only on one platform
- If we have a service which only works on the browser we can use the DI to inject a different service in the **AppServerModule**
- If we have a service that can only be used in Node.js we can use the DI to provide a different service in the **AppModule**

shim component / directive/ pipe

academeez

- In this case we are using a module which provided us with component or directives or pipes which are only supported in one platform
- You can only include that module in the browser or the node will crash
- you can add an alternate root browser module and add the problematic module to the browser module
- In the node root module we will create shims for the components / directive / pipes

Summary

academeez

- Using shim we can avoid the branching in our code and create a separate logic for node and the browser
- we can avoid using the PLATFORM_ID
- Another technique we can use to load different logic in the browser and node is dynamic module loading

academeez

Thank You

Next Lesson: 9. Dynamic modules and dynamic components