MODERN FRONT-END DEVELOPMENT WITH ANGULAR JS 2.0

History of the front-end, Typescript, Component Architecture, Observables.

JAVASCRIPT FRAMEWORKS

AnuglarJS, React, Backbone, TodoMVC, Ember, Polymer, Knockout, Aurelia, Spine, Brick, NuclearJS, Dojo, Matreshka

JAVASCRIPT-TARGETING LANGUAGES

Typescript, Dart, Coffeescript, asm.js, Coco, Uberscript, Caffeine, EmberScript, LiteScript, Flow, Latte JS

WHY?

TIMES HAVE CHANGED (THE WEB IS MATURING)

1. HYPER-TEXT (1963)
2. HTML ON WWW (1987)
3. AJAX (1999)
4. HTML5 “APP SANDBOX APIS” (2004)
5. WEB ASSEMBLY, COMPONENTS (NOW)
6. ???

THE BROWSER IS JUST A CLIENT

- Desktop application
- Smartphone app
- <u>Web app</u>
- Embedded device (IoT)

It's just another client.

WEB APP OR NATIVE APP?

- Communicates over HTTPS
- Runs in fullscreen
- Store data locally between runs
- Works offline
- Renders OpenGL graphics
- Uses background threads
- Can geo-locate using GPS
- Supports gamepads/joysticks
- Uses system speech recognition

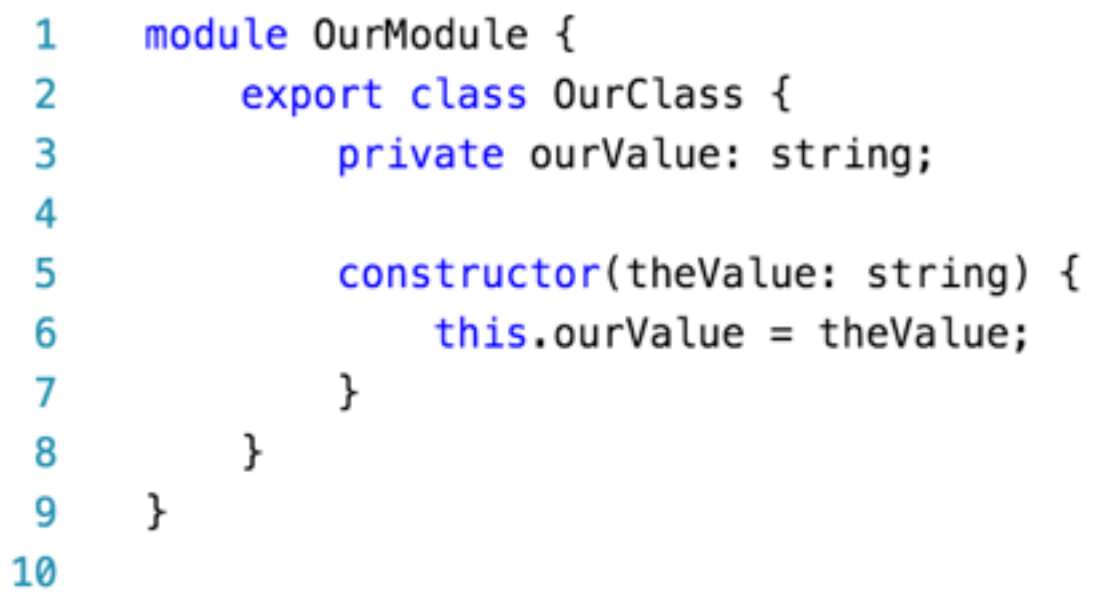
THE BROWSER ISN'T A PERFECT PLATFORM, BUT ANGULAR JS CAN HELP

BUT FIRST . . .

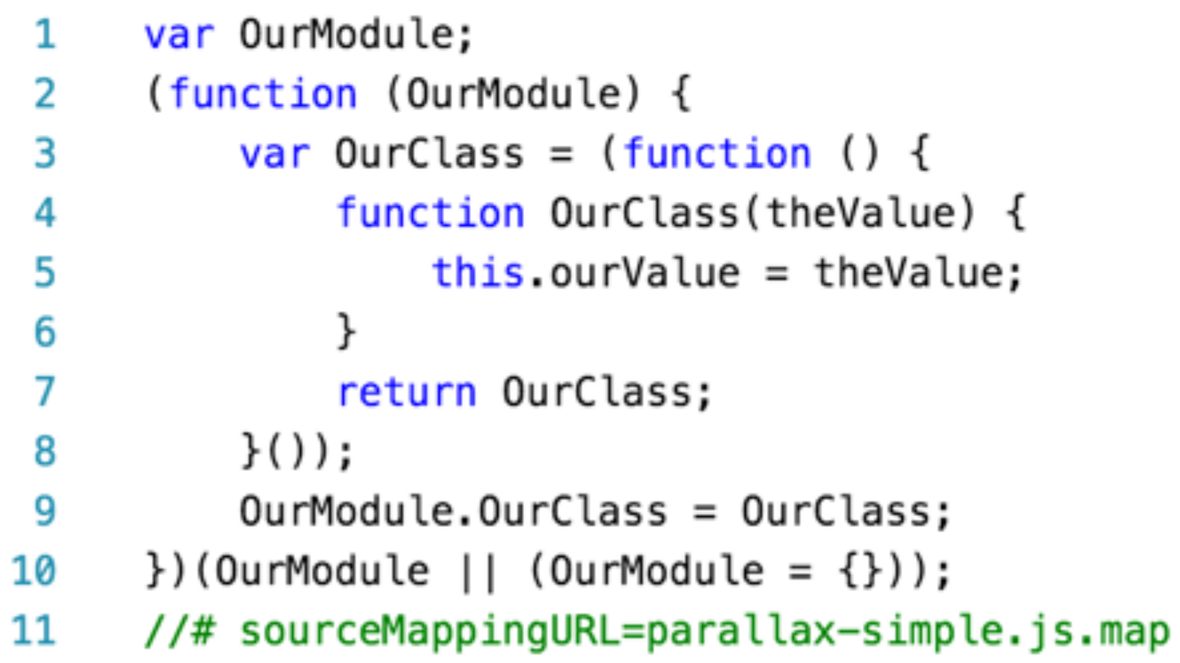
TYPESCRIPT

- Superset of Javascript (ES6) with optional typing
- Open source (Windows, Linux, OS X *nee* macOS)
- Passes code through — very little mangling
- Weakly-typed vs. strongly-typed (use both!)
- Recommended but not required for AngularJS 2.0

TYPESCRIPT COMPILER (TSC) OUTPUT



```typescript
module OurModule {
    export class OurClass {
        private ourValue: string;

        constructor(theValue: string) {
            this.ourValue = theValue;
        }
    }
}
```



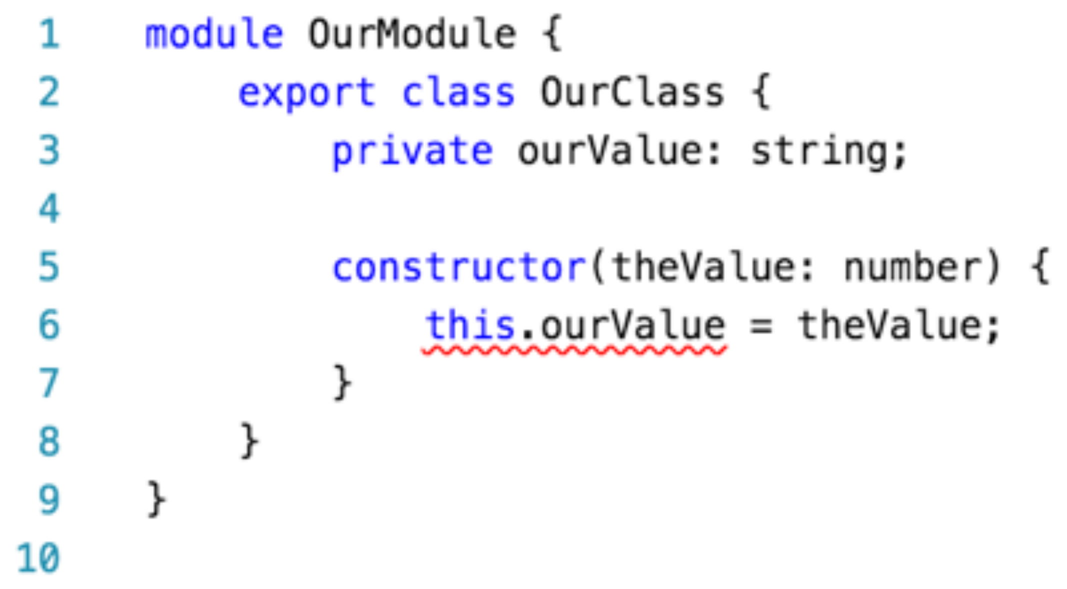
```javascript
var OurModule;
(function (OurModule) {
    var OurClass = (function () {
        function OurClass(theValue) {
            this.ourValue = theValue;
        }
        return OurClass;
    }());
    OurModule.OurClass = OurClass;
})(OurModule || (OurModule = {}));
//# sourceMappingURL=parallax-simple.js.map
```

Typescript

Outputted Javascript

```
tsc --sourcemap --target ES5 parallax-simple.ts
```

TYPESCRIPT COMPILER (TSC) OUTPUT



```
module OurModule {
    export class OurClass {
        private ourValue: string;

        constructor(theValue: number) {
            this.ourValue = theValue;
        }
    }
}
```

```
parallax-simple.ts(6,13): error TS2322: Type 'number' is not assignable to type 'string'.
```

- *Type-checking done at compile time but errors allowed (Javascript allows it)*
- *Reduces errors while developing*
- *Helps editors provide assistance (VS Code, WebStorm, others)*

ANGULAR JS 2.0

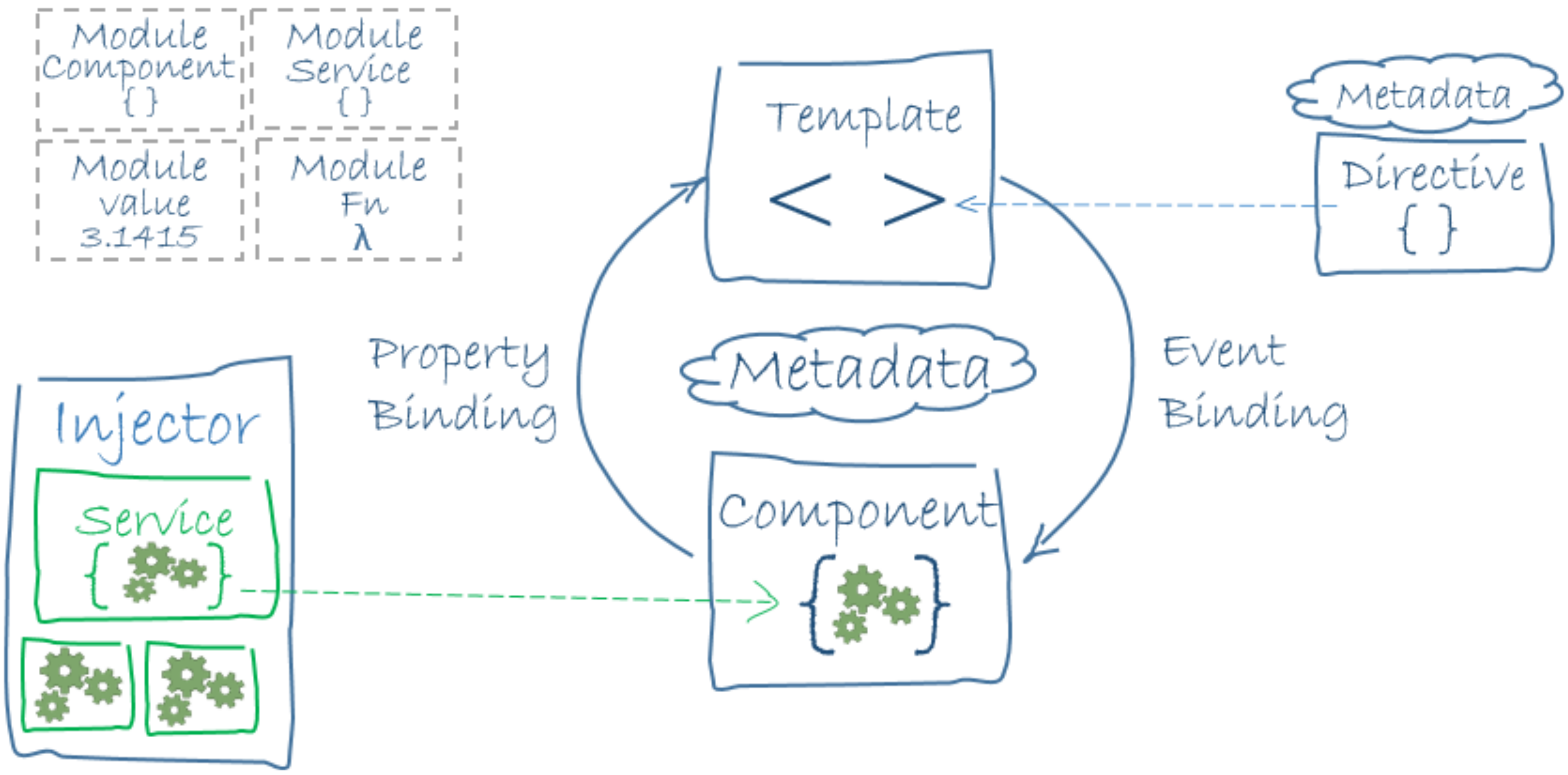
ANGULAR JS 2.0

- Open source, developed by Google + community
- Application framework for web and native applications
- Encourages code reuse through modules
- Not backwards-compatible with AngularJS 1.x but there is a migration path
- Typescript preferred
- Much faster than Angular JS 1.x
- Cleaner architecture than Angular JS 1.x

ANGULAR JS 2.0 ARCHITECTURE

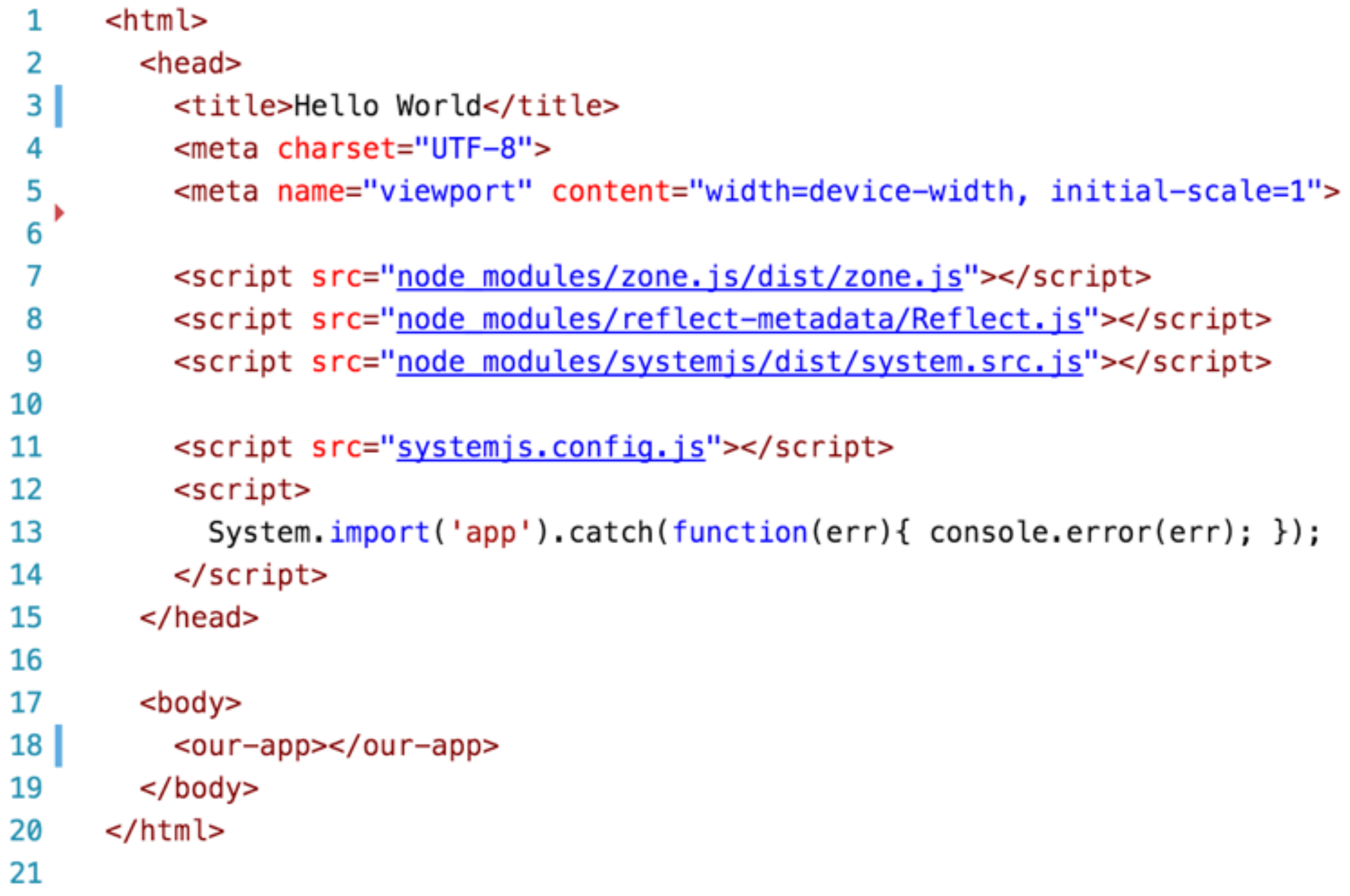
- **Component**-based design.
- Everything is a "**directive**" in one of three types: component (class with a view template), structural (alters DOM e.g. ng*If, ngSwitch, ngFor*), or attributive (affects style or behavior, e.g. ngClass).
- Uses **dependency injection**.
- **Services** are any Javascript class you register for injection. Preferably contain discrete, unique groupings of application logic.
- Miscellany: pipes, lifecycle hooks, routers, animation, and more.

ANGULAR JS 2.0 ARCHITECTURE



Credit: https://angular.io/resources/images/devguide/architecture/overview2.png

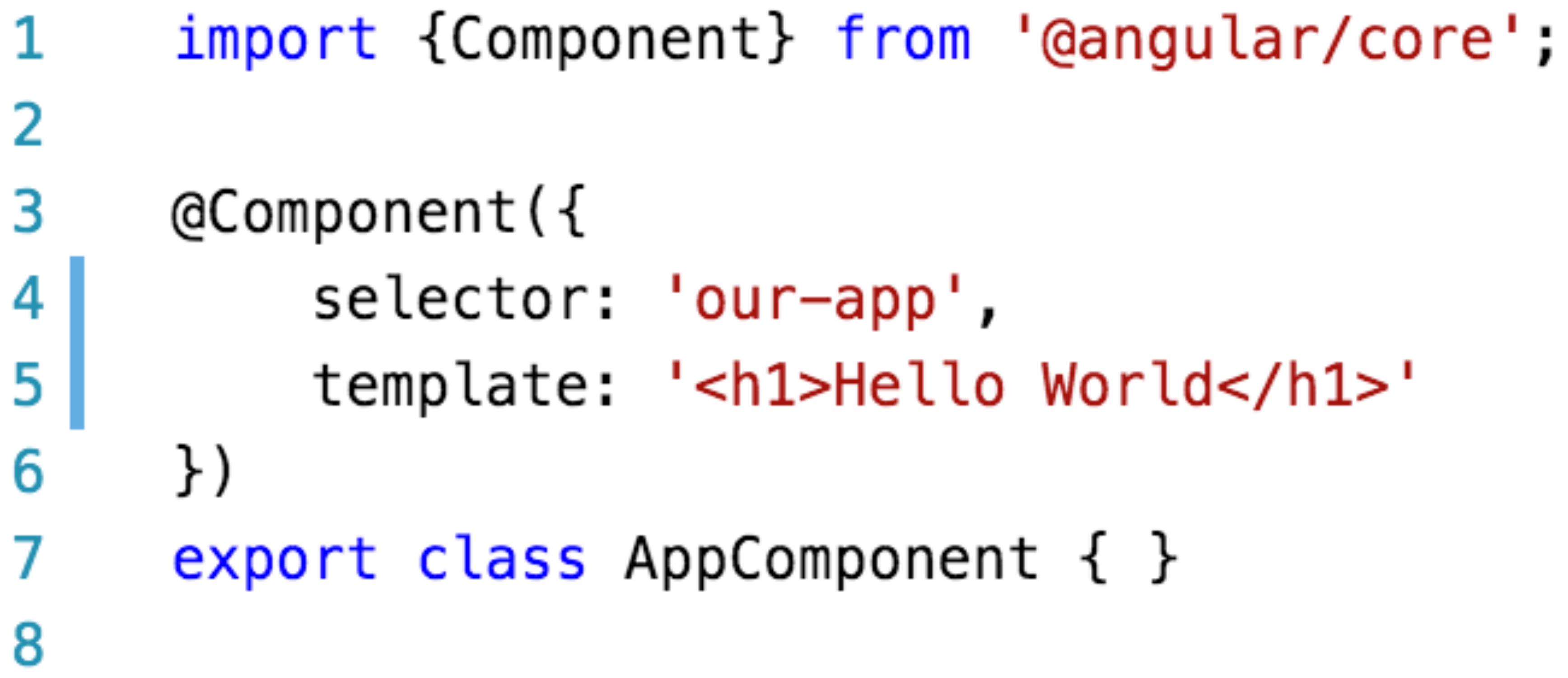
ANGULAR JS 2.0 HELLO WORLD



```html
<html>
  <head>
    <title>Hello World</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <script src="node_modules/zone.js/dist/zone.js"></script>
    <script src="node_modules/reflect-metadata/Reflect.js"></script>
    <script src="node_modules/systemjs/dist/system.src.js"></script>

    <script src="systemjs.config.js"></script>
    <script>
      System.import('app').catch(function(err){ console.error(err); });
    </script>
  </head>

  <body>
    <our-app></our-app>
  </body>
</html>
```

From https://angular.io/docs/ts/latest/quickstart.html

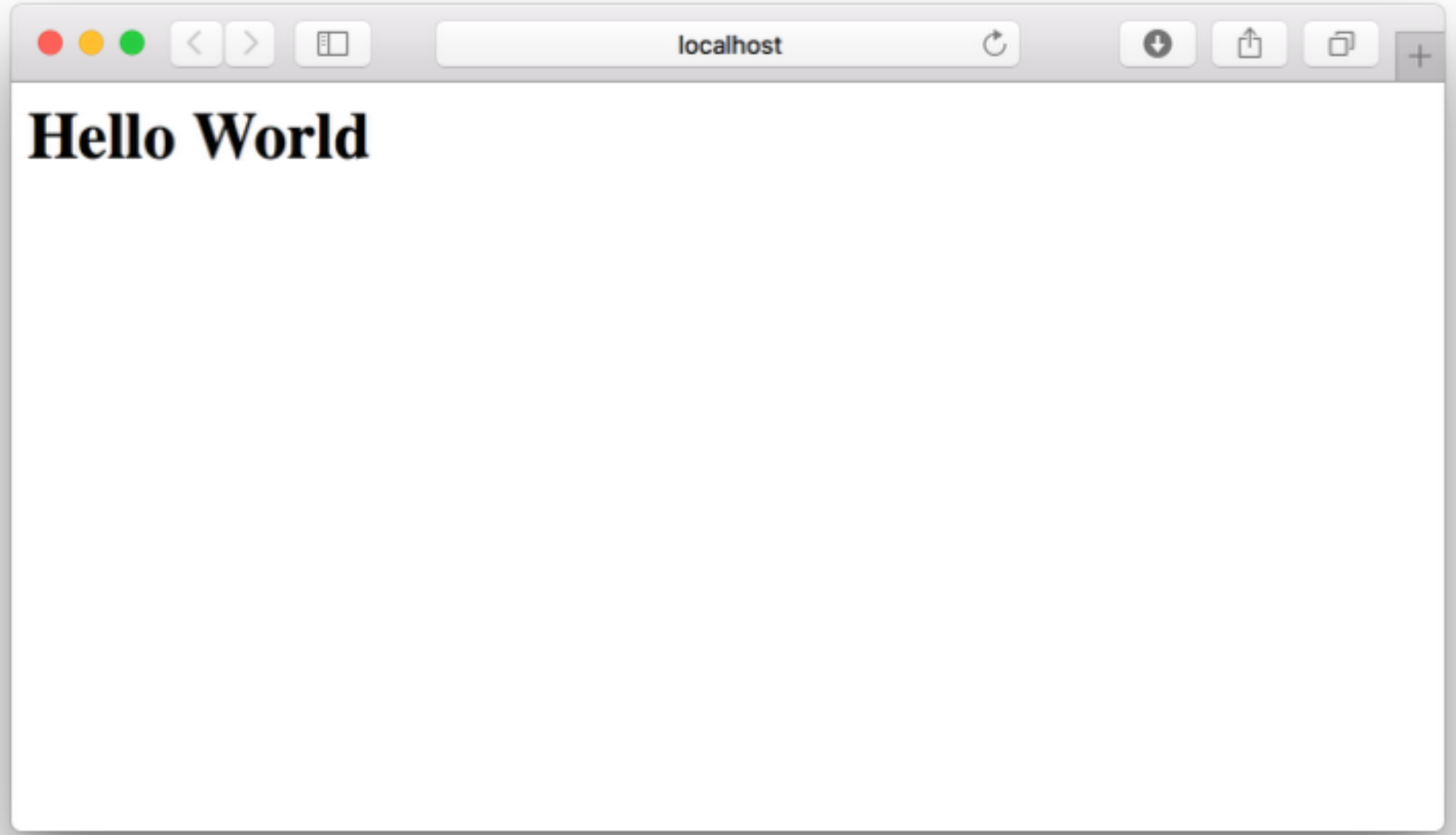
ANGULAR JS 2.0 HELLO WORLD



```
import {Component} from '@angular/core';

@Component({
    selector: 'our-app',
    template: '<h1>Hello World</h1>'
})
export class AppComponent { }

```

From https://angular.io/docs/ts/latest/quickstart.html

ANGULAR JS 2.0 HELLO WORLD

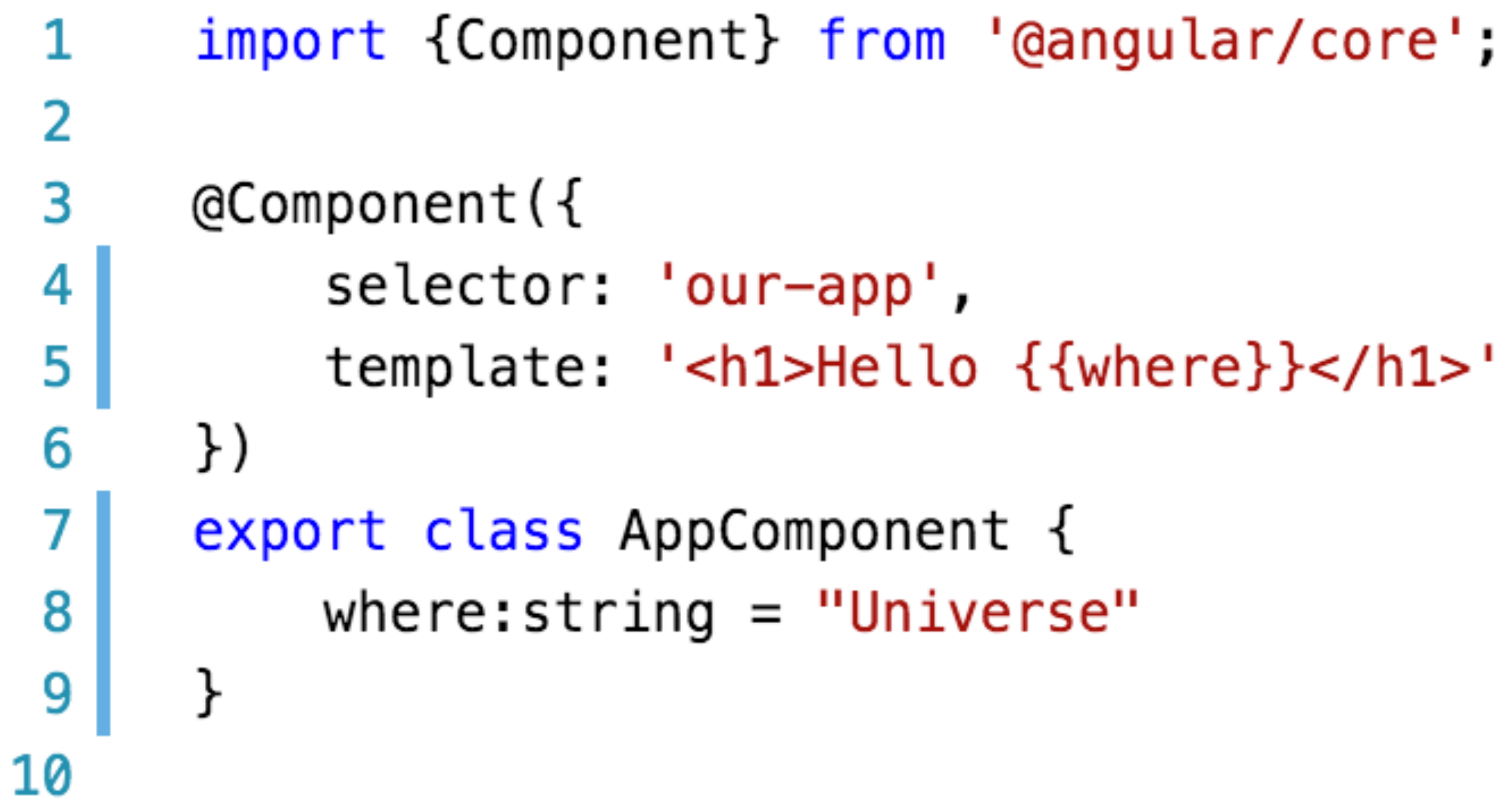


Hello World

WOW!

DATA BINDING VIA INTERPOLATION

ANGULAR JS 2.0 HELLO WORLD



```
import {Component} from '@angular/core';

@Component({
    selector: 'our-app',
    template: '<h1>Hello {{where}}</h1>'
})
export class AppComponent {
    where:string = "Universe"
}
```

ANGULAR JS 2.0 HELLO WORLD

```
import {Component} from '@angular/core';

@Component({
    selector: 'our-app',
    template: '<h1 *ngFor="let where of locations">Hello {{where}}</h1>'
})
export class AppComponent {
    locations = ['World', 'Universe', 'Parallel Dimension'];
}

```

BASIC EVENT HANDLING

ANGULAR JS 2.0 HELLO WORLD

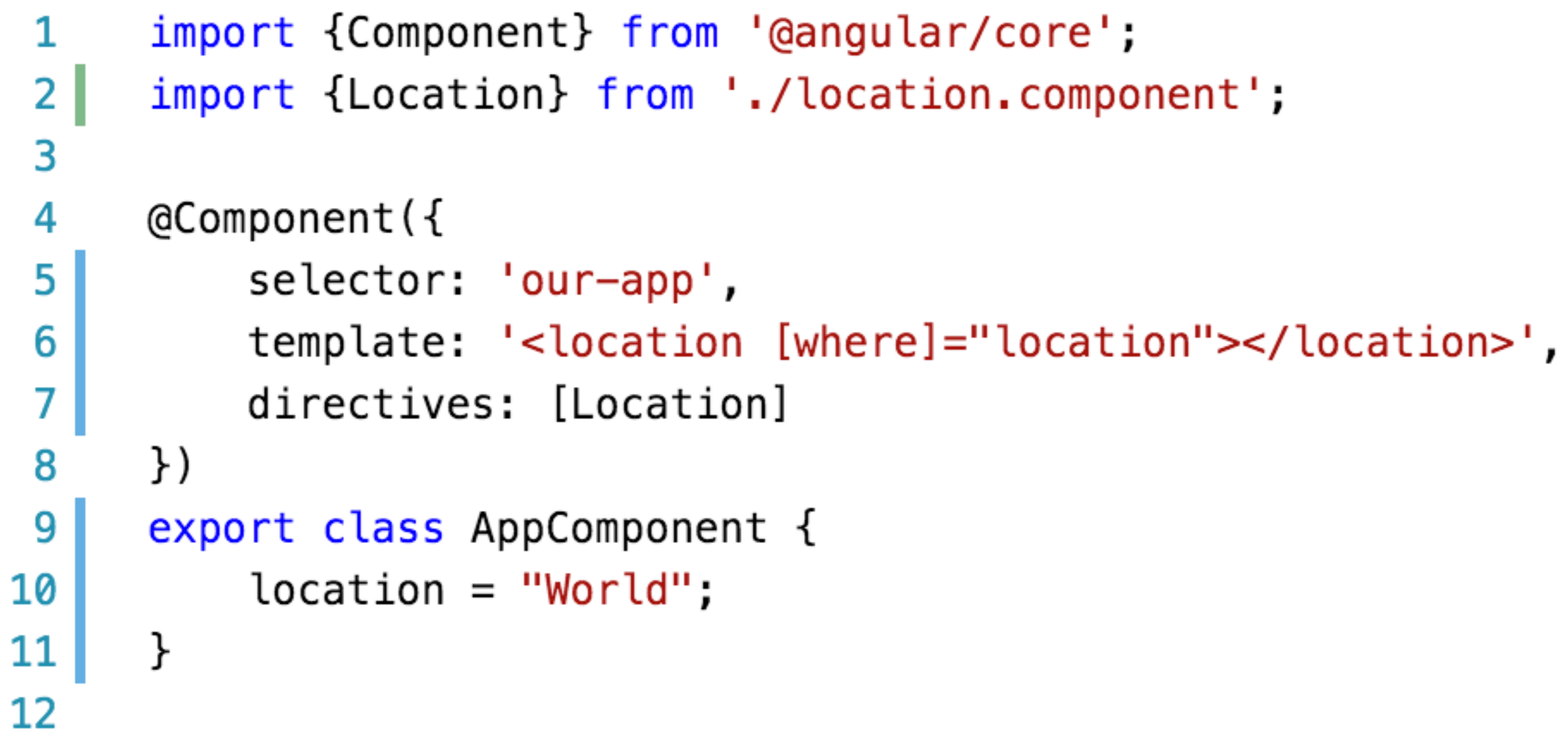
```
import {Component} from '@angular/core';

@Component({
    selector: 'our-app',
    template: '<span *ngFor="let where of locs" (click)="whichLoc(where)">Hello {{where}}<br /></span>'
})
export class AppComponent {
    locs = ['World', 'Universe', 'Parallel Dimension'];

    whichLoc(location:any) {
        console.log(location);
    }
}
```

COMPONENTS IN COMPONENTS

ANGULAR JS 2.0 HELLO WORLD

```
import {Component} from '@angular/core';

@Component({
    selector: 'location',
    template: '<h1>Hello {{where}}</h1>',
    inputs: ['where']
})
export class Location {
    public where:string;
}

```

ANGULAR JS 2.0 HELLO WORLD



```
import {Component} from '@angular/core';
import {Location} from './location.component';

@Component({
    selector: 'our-app',
    template: '<location [where]="location"></location>',
    directives: [Location]
})
export class AppComponent {
    location = "World";
}
```

JAVASCRIPT PATTERNS

- Javascript at Netscape (1995)
- Callback culture with jQuery (2006)
- Promises (2010)
- Observables (2012)

OBSERVABLES ARE COOL

OBSERVABLES ARE LAZY PROMISES WHICH CAN BE CANCELLED AND RE-TRIED

PROMISE VS OBSERVABLE

```
var promise = new Promise((resolve) => {
  setTimeout(() => {
    resolve(42);
  },  500);
  console.log('promise started');
});

promise.then(x => console.log(x));
```

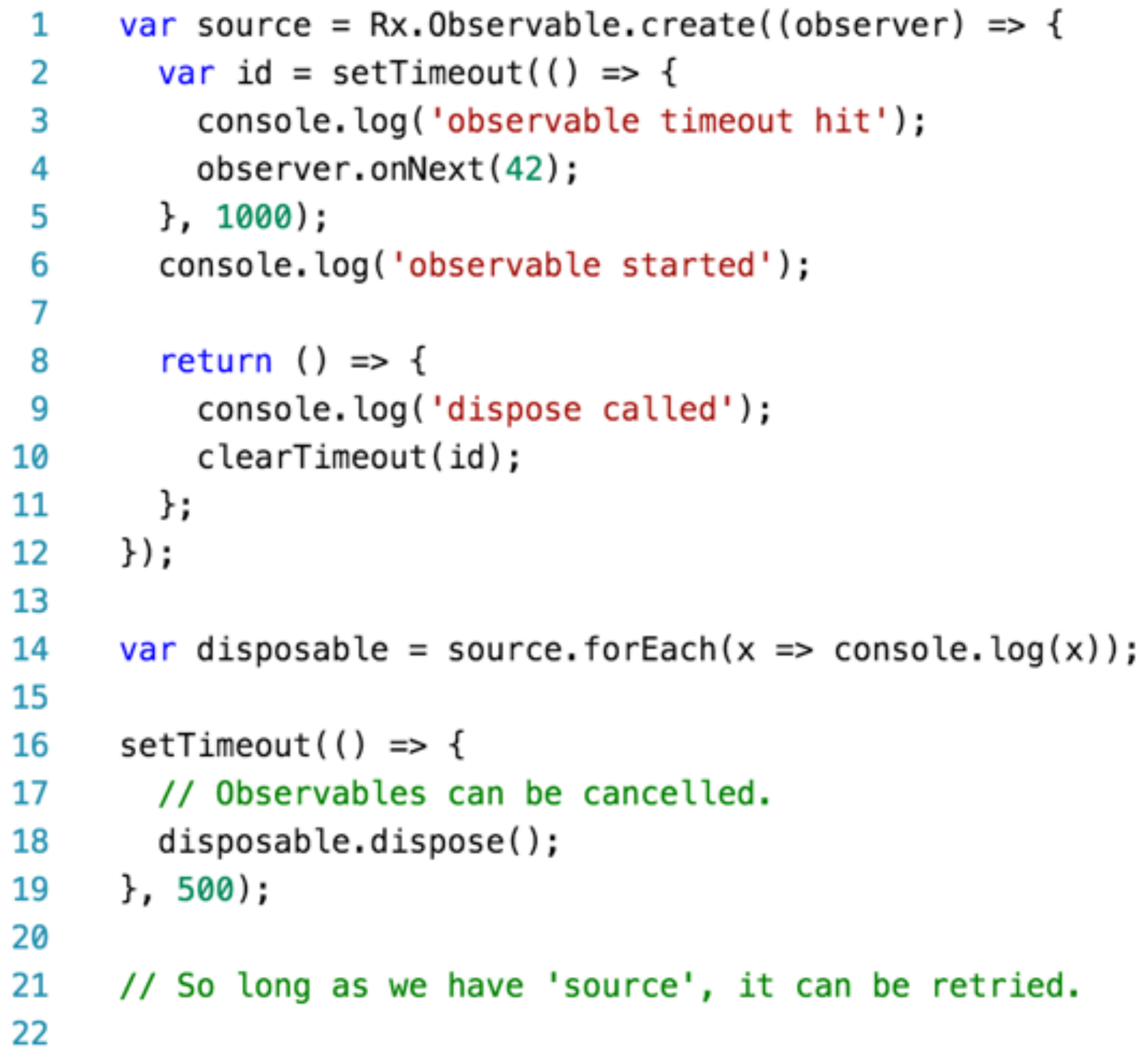
Promise

```
var source = Rx.Observable.create((observer) => {
  setTimeout(() => {
    observer.onNext(42);
  }, 500);
  console.log('observable started');
});

source.forEach(x => console.log(x));

```

Observable

From https://egghead.io/lessons/rxjs-rxjs-observables-vs-promises

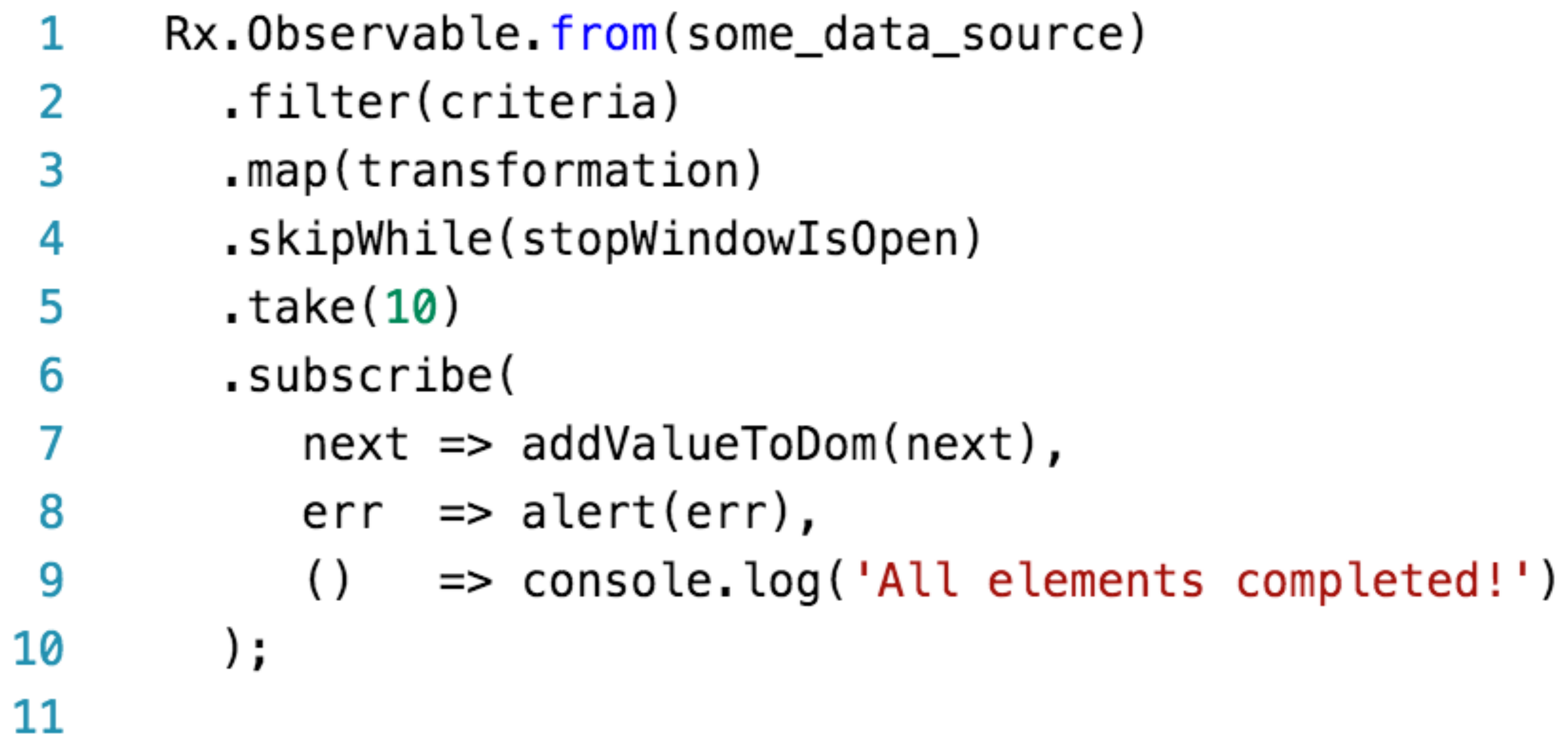
PROMISE VS OBSERVABLE



```
var source = Rx.Observable.create((observer) => {
  var id = setTimeout(() => {
    console.log('observable timeout hit');
    observer.onNext(42);
  }, 1000);
  console.log('observable started');

  return () => {
    console.log('dispose called');
    clearTimeout(id);
  };
});

var disposable = source.forEach(x => console.log(x));

setTimeout(() => {
  // Observables can be cancelled.
  disposable.dispose();
}, 500);

// So long as we have 'source', it can be retried.

```

From https://egghead.io/lessons/rxjs-rxjs-observables-vs-promises

PROMISE VS OBSERVABLE



```javascript
Rx.Observable.from(some_data_source)
  .filter(criteria)
  .map(transformation)
  .skipWhile(stopWindowIsOpen)
  .take(10)
  .subscribe(
     next => addValueToDom(next),
     err  => alert(err),
     ()   => console.log('All elements completed!')
  );
```

Officially included in ES7

From https://www.quora.com/What-is-the-most-useful-thing-you-can-do-with-RxJS-observables

RESOURCES

- https://angular.io
- http://typescriptlang.org
- https://code.visualstudio.com
- https://egghead.io