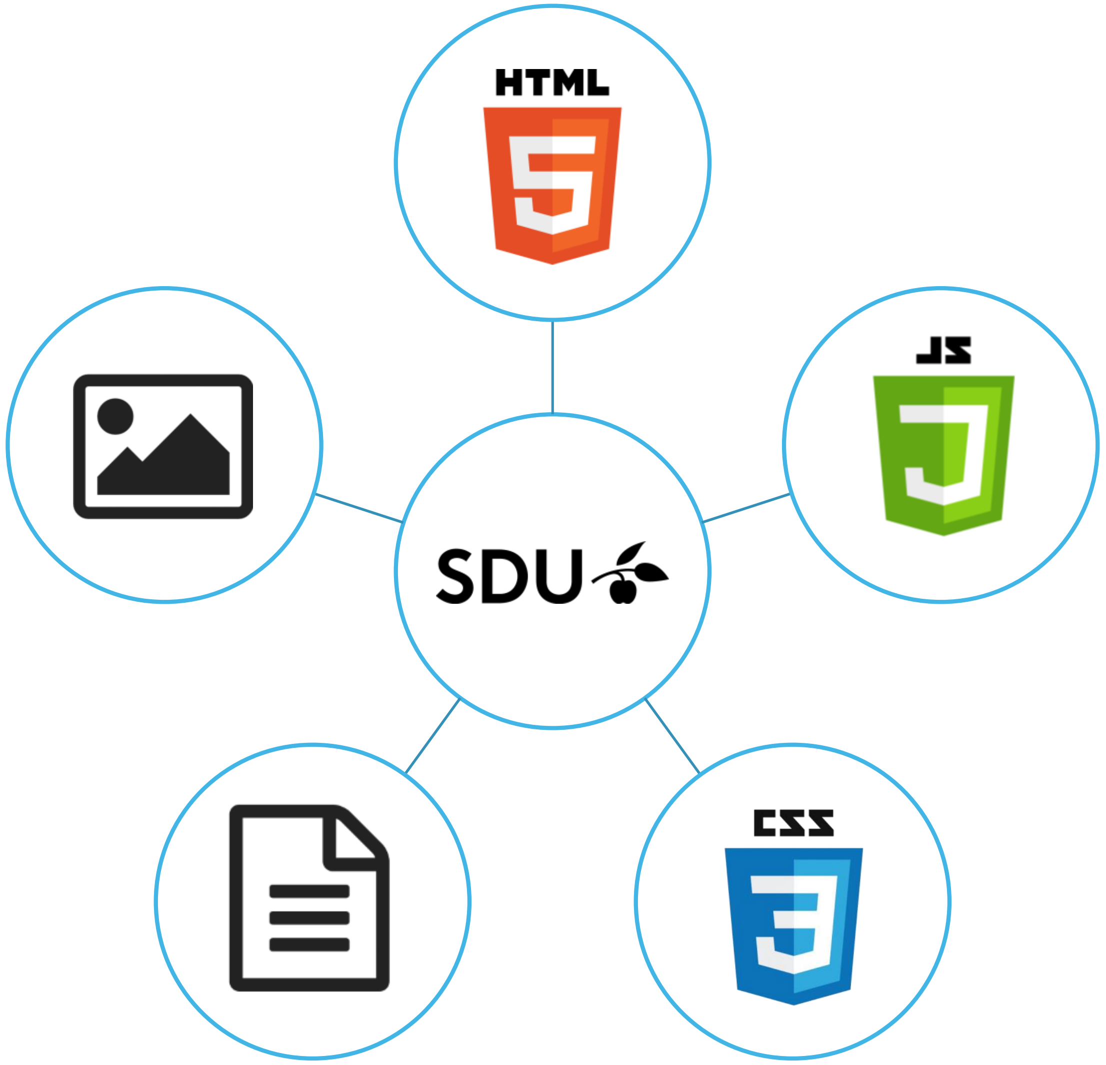
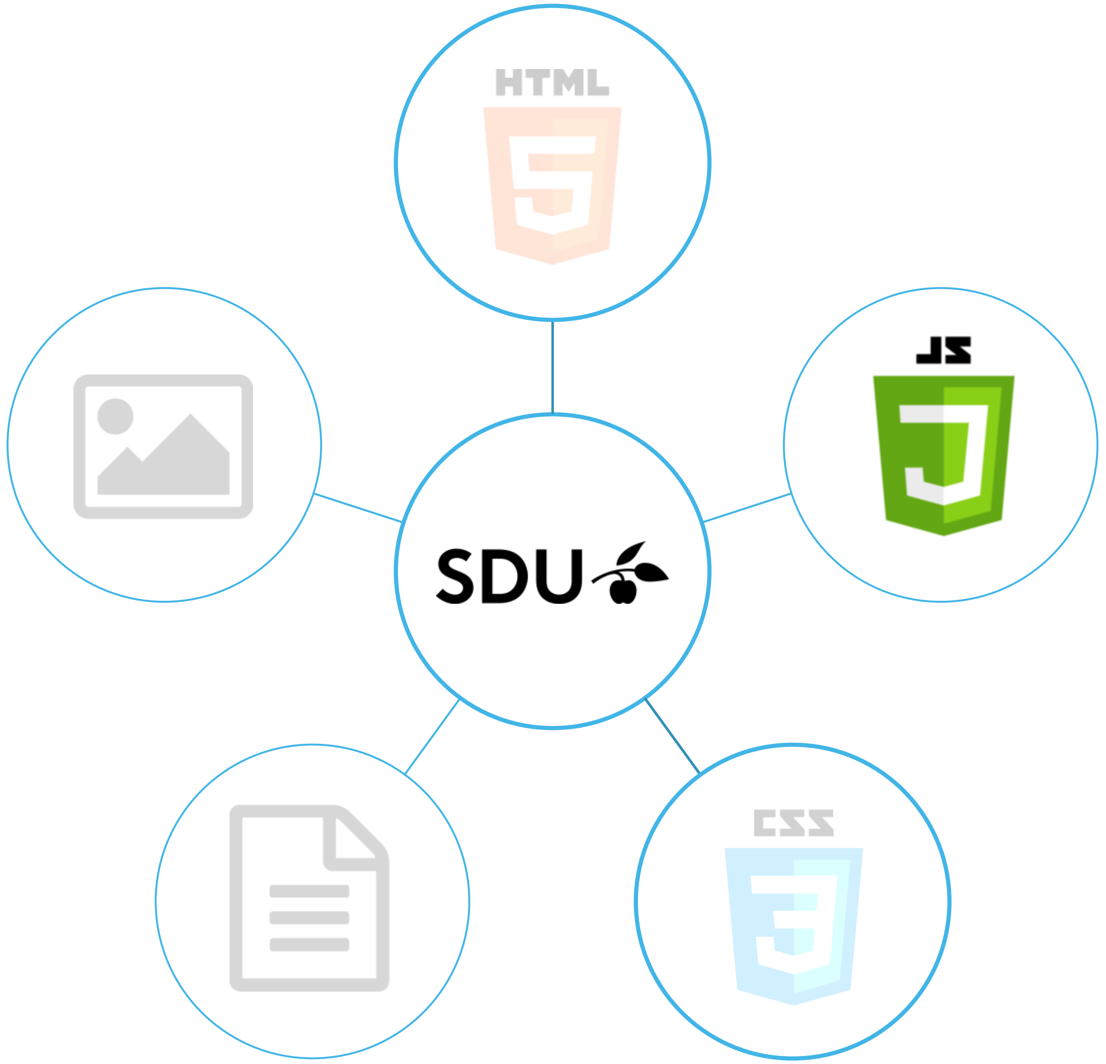
Advanced Javascript

20. marts 2018

What is a website?



What is a website?



HTML – HyperText Markup Language

Defines the structure of a webpage

CSS – Cascading Style Sheet

Defines the visual presentation of HTML elements

JS – Javascript

Handles user interaction and dynamic content

ES6 Introduction

A history lesson

1996 – Standardization in ECMA

1997 – ECMA-262 (ECMAScript)

1998 – ECMAScript 2

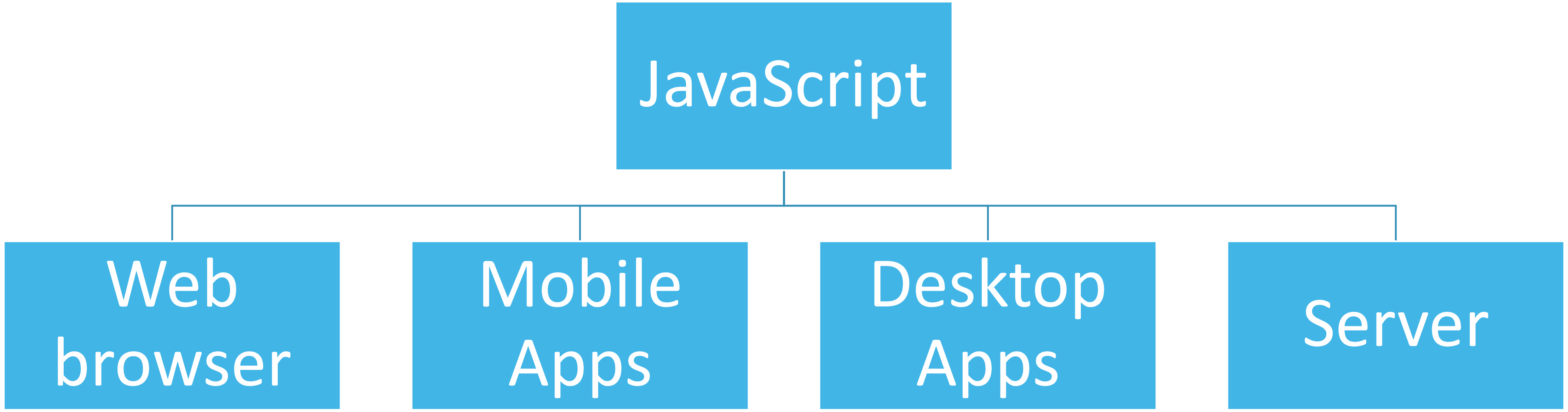
1999 – ECMAScript 3

2005 – Work started on ECMAScript 4, Microsoft and Yahoo opposed it,
ECMAScript 3.1 was the compromise

2009 – Opposing parties meet in Oslo and come to an agreement.
ES3.1 is renamed to ES5, which is what we primarily use today.

2015 – ES6 is released. From now on release will be named after the year it is release

JavaScript of today



Pros of JavaScript

- Easy syntax
- Functions are objects, i.e. easy to work with
- Independent from any of the huge organizations and companies
- The only native browser language
- It has a vast community
- A ton of libraries, frameworks and tools

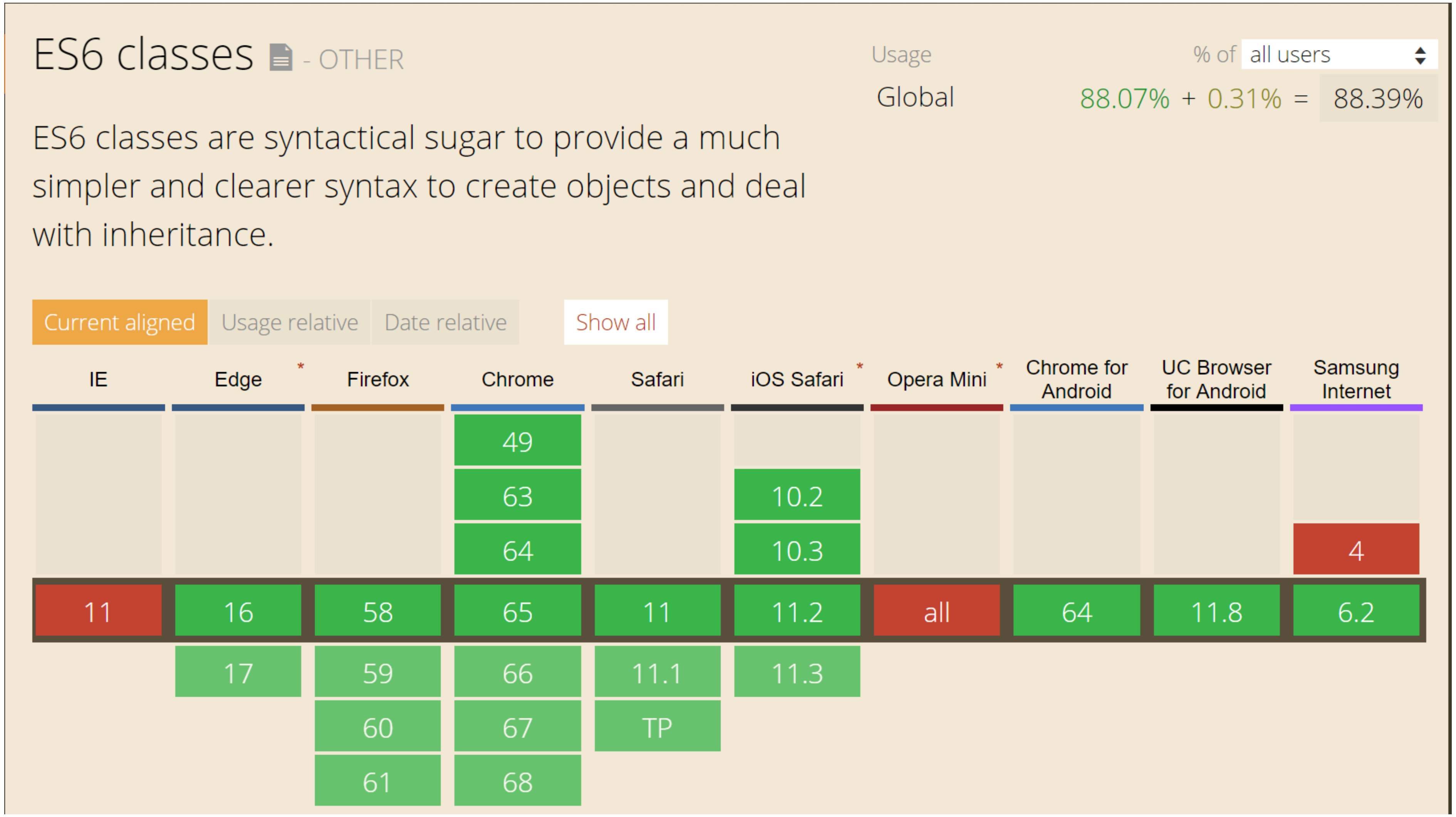
Cons of JavaScript

- There are not many clean code practices
- Rapid development and change, makes tools, frameworks and libraries obsolete fast
- Comparisons between Java-like languages and JavaScript is confusing

Main goals of ES6

- Fix major issues with ES5
- Backwards compatibility with ES5
- Modernized syntax
- Well suited for larger applications
- Extended feature set in the standard library

ES6 Adoption



ES6 classes - OTHER

Usage: Global

% of all users

88.07% + 0.31% = 88.39%

ES6 classes are syntactical sugar to provide a much simpler and clearer syntax to create objects and deal with inheritance.

Current aligned | Usage relative | Date relative | Show all

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			63		10.2				
			64		10.3				4
11	16	58	65	11	11.2	all	64	11.8	6.2
	17	59	66	11.1	11.3				
		60	67	TP					
		61	68						

ES6 Adoption (cont.)

Popular browsers of today is covering roughly 96% of the ES6 scope.

For a full compatibility table: https://kangax.github.io/compat-table/es6/

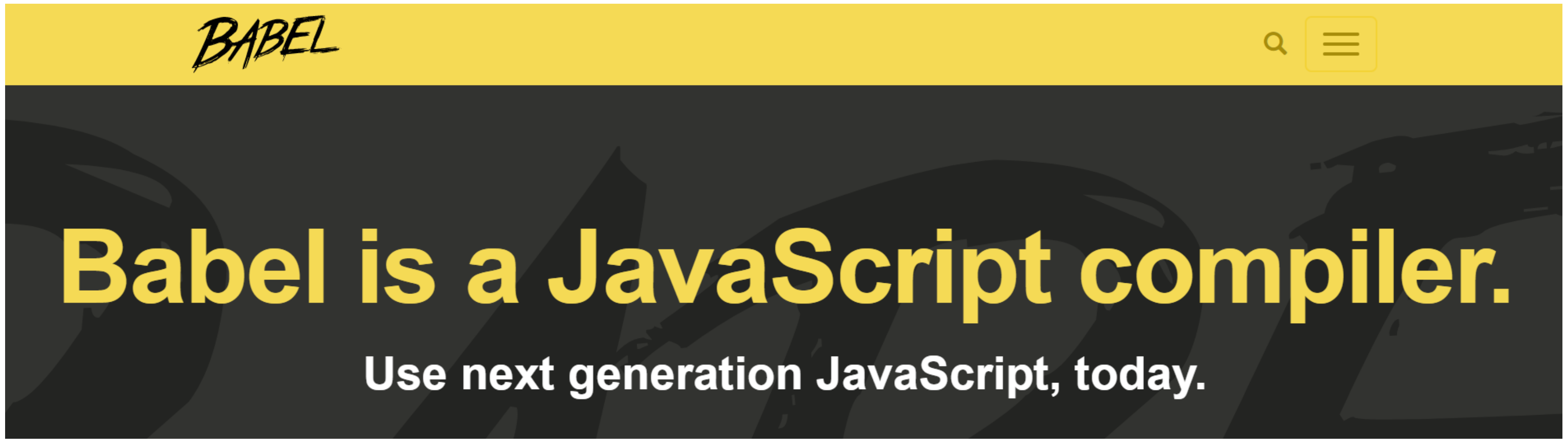
Using ES6 in an ES5 world

Even though there is a great adoption in the latest browsers, we as web developers must make our websites accessible from older browsers.

Either we refrain from using ES6 altogether

Or we use transpilers! ☺

Enter:


BABEL

Babel is a JavaScript compiler.

Use next generation JavaScript, today.

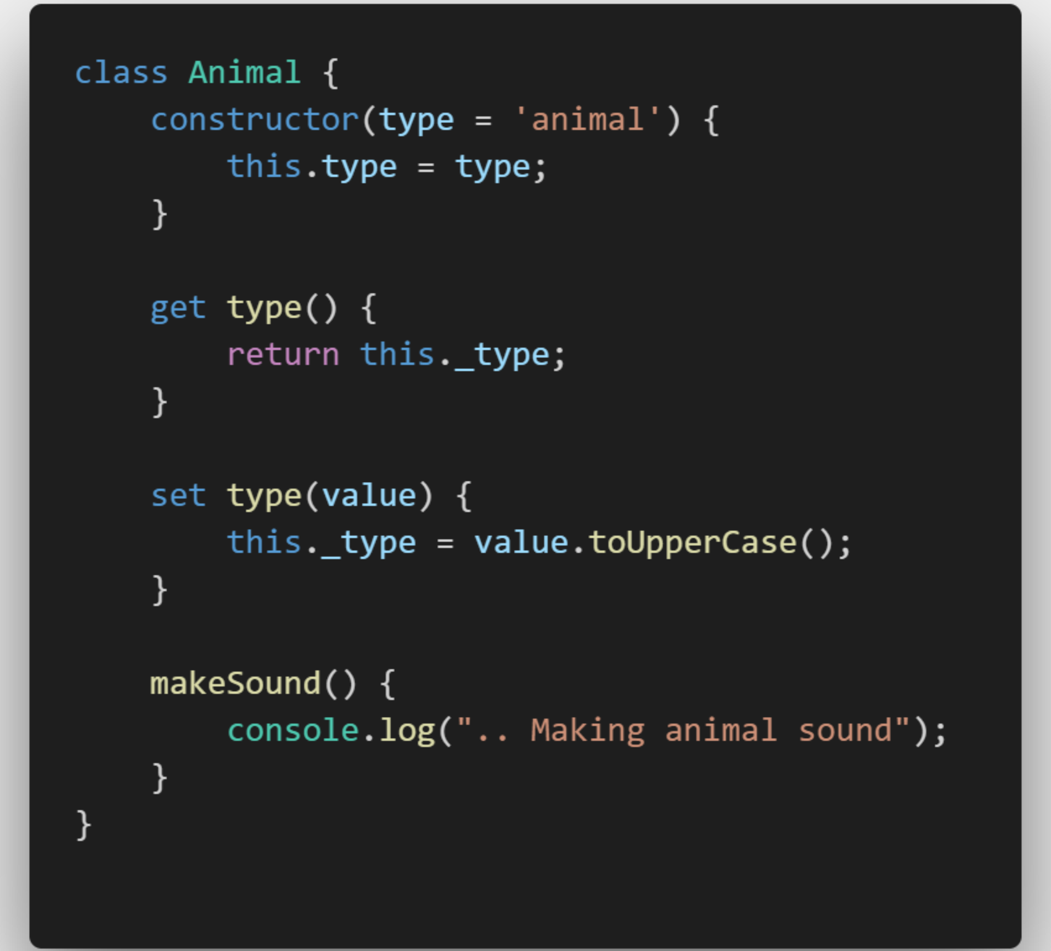
Babel - Usage

Command line: **babel es6.js -o es5.js**

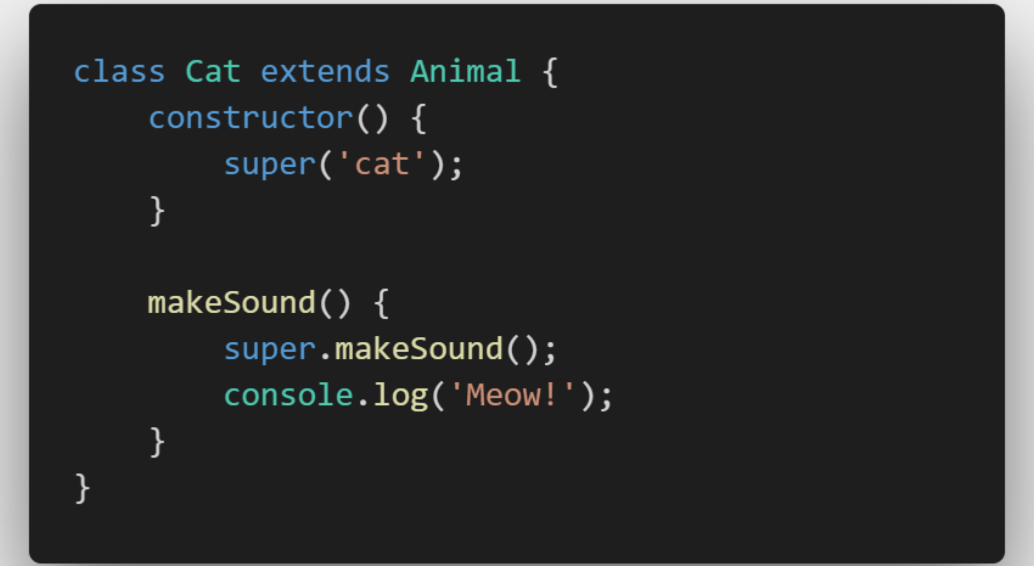
Online: https://babeljs.io/repl

Installation docs: https://babeljs.io/docs/setup/

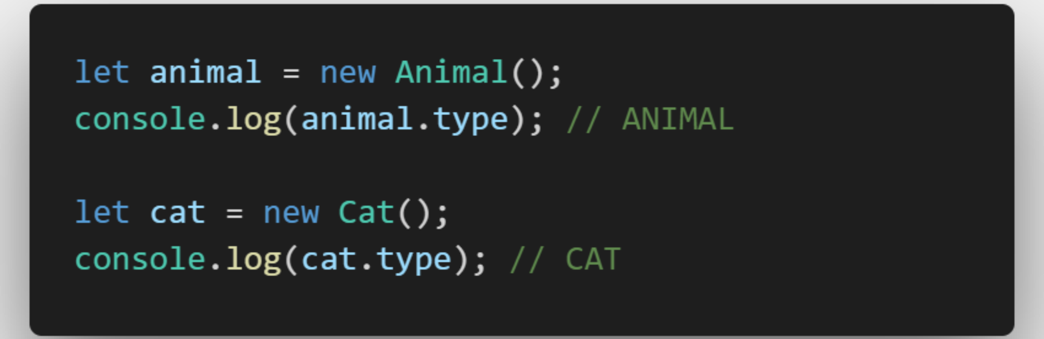
ES6 - Classes



```javascript
class Animal {
    constructor(type = 'animal') {
        this.type = type;
    }

    get type() {
        return this._type;
    }

    set type(value) {
        this._type = value.toUpperCase();
    }

    makeSound() {
        console.log(".. Making animal sound");
    }
}
```

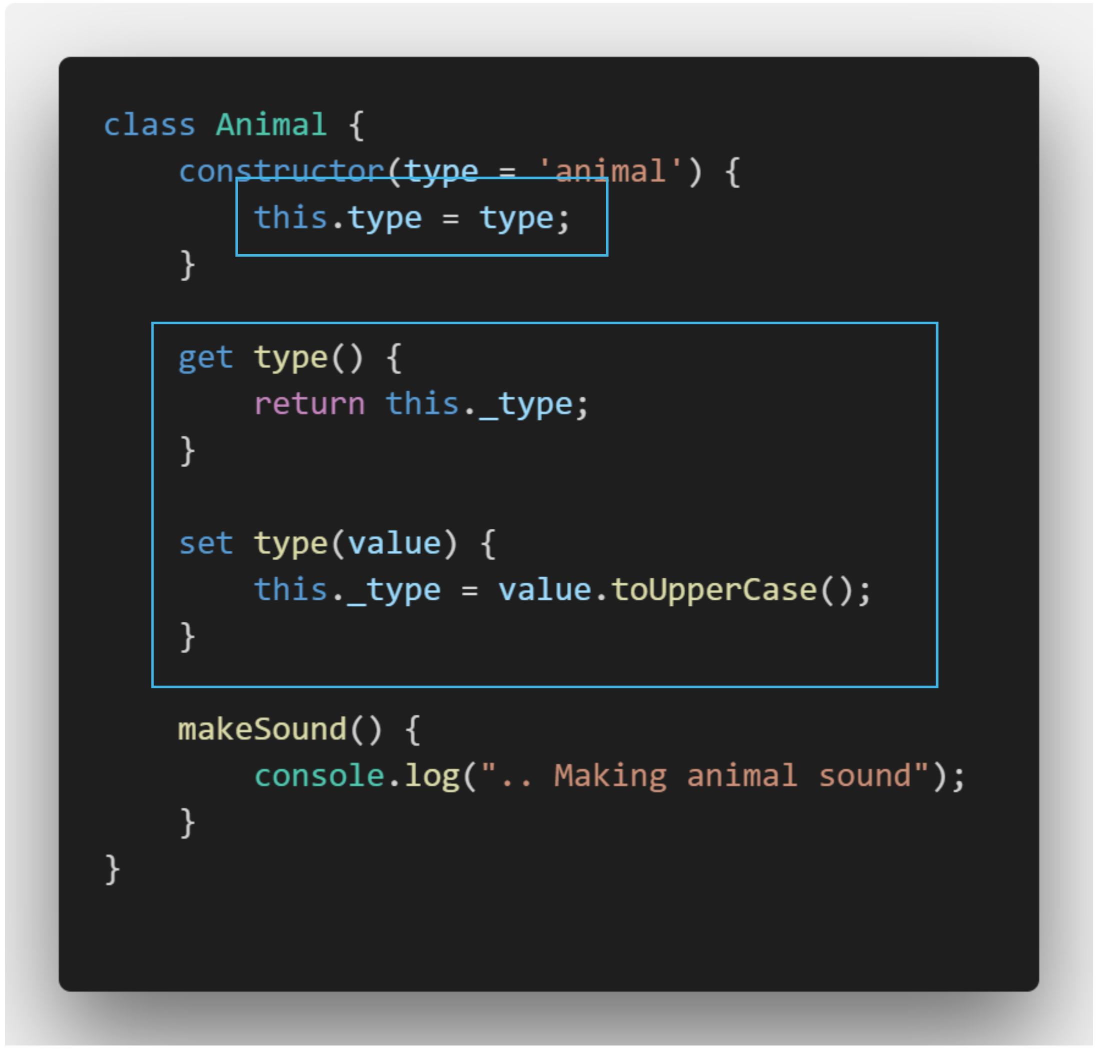


```javascript
class Cat extends Animal {
    constructor() {
        super('cat');
    }

    makeSound() {
        super.makeSound();
        console.log('Meow!');
    }
}
```



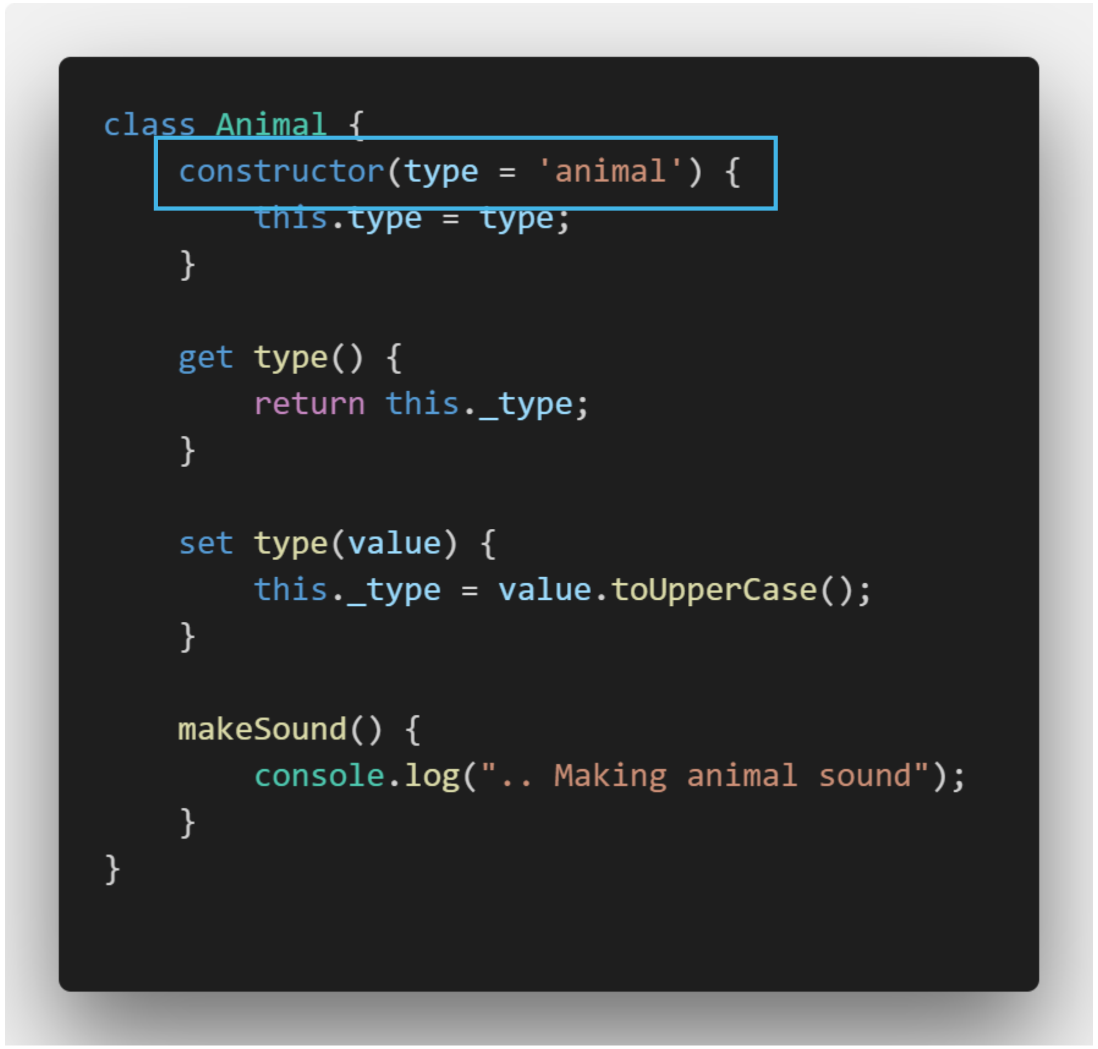
```javascript
let animal = new Animal();
console.log(animal.type); // ANIMAL

let cat = new Cat();
console.log(cat.type); // CAT
```

ES6 – Setters and Getters



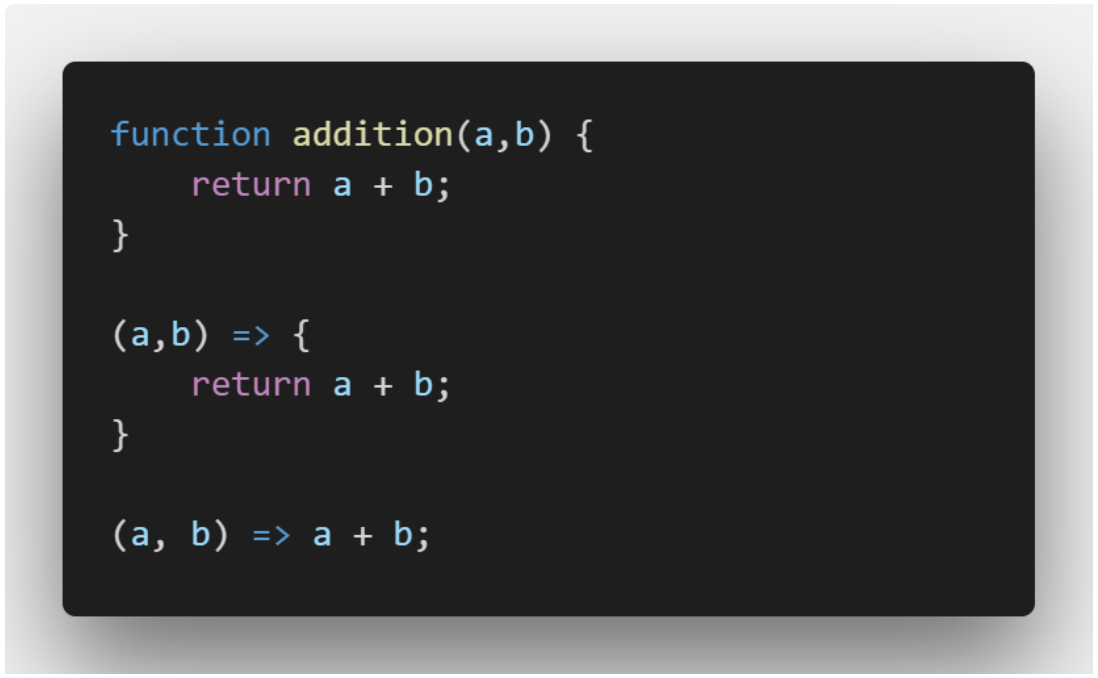
```
class Animal {
    constructor(type = 'animal') {
        this.type = type;
    }

    get type() {
        return this._type;
    }

    set type(value) {
        this._type = value.toUpperCase();
    }

    makeSound() {
        console.log(".. Making animal sound");
    }
}
```

ES6 – Default parameters



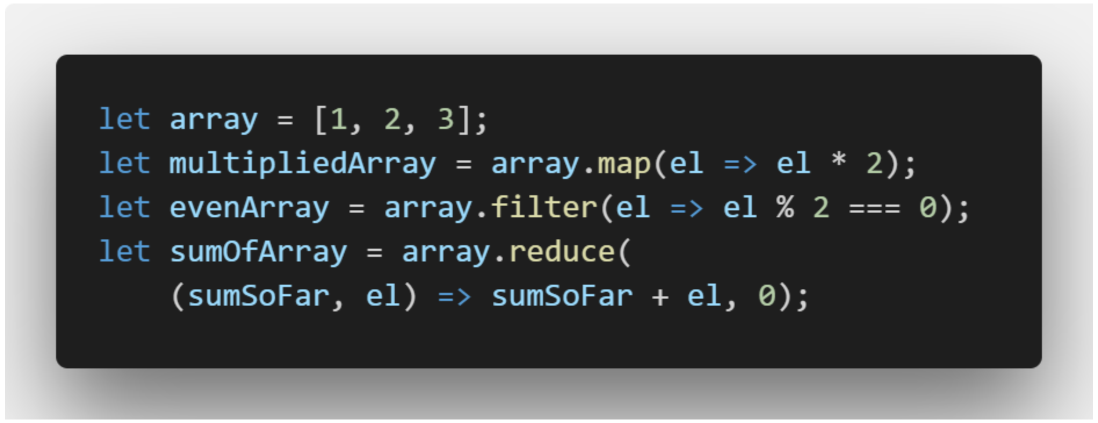
```
class Animal {
    constructor(type = 'animal') {
        this.type = type;
    }

    get type() {
        return this._type;
    }

    set type(value) {
        this._type = value.toUpperCase();
    }

    makeSound() {
        console.log(".. Making animal sound");
    }
}
```

ES6 – Arrow functions



```
function addition(a,b) {
    return a + b;
}

(a,b) => {
    return a + b;
}

(a, b) => a + b;
```

Examples:



```
let array = [1, 2, 3];
let multipliedArray = array.map(el => el * 2);
let evenArray = array.filter(el => el % 2 === 0);
let sumOfArray = array.reduce(
    (sumSoFar, el) => sumSoFar + el, 0);
```

ES6 – Async programming via Promise

A promise is an object, that holds the state of an async function

- Can be waiting, resolved or rejected

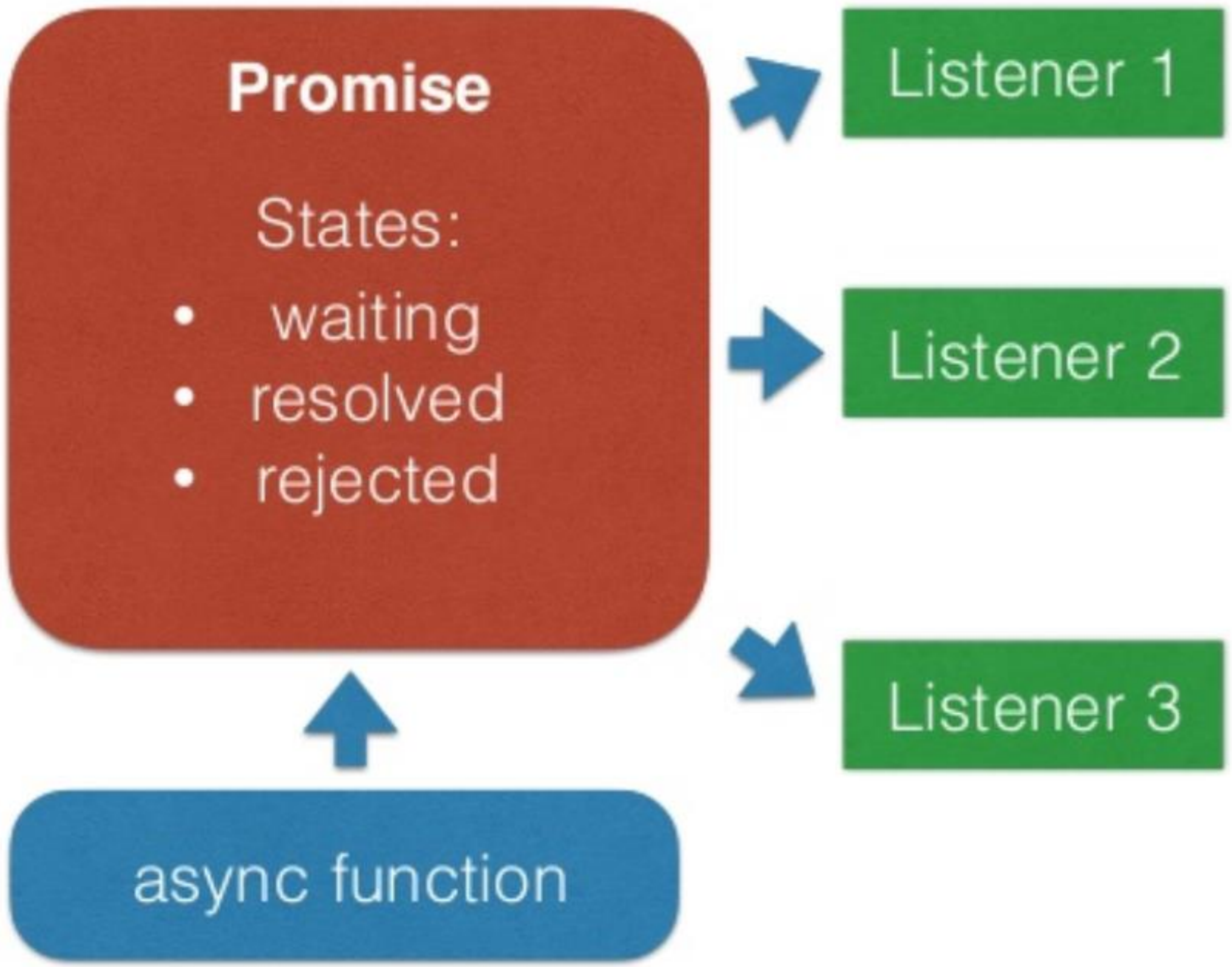
Allows us to return a promise and work with that, even when the async function is not done.

- Easier to read code

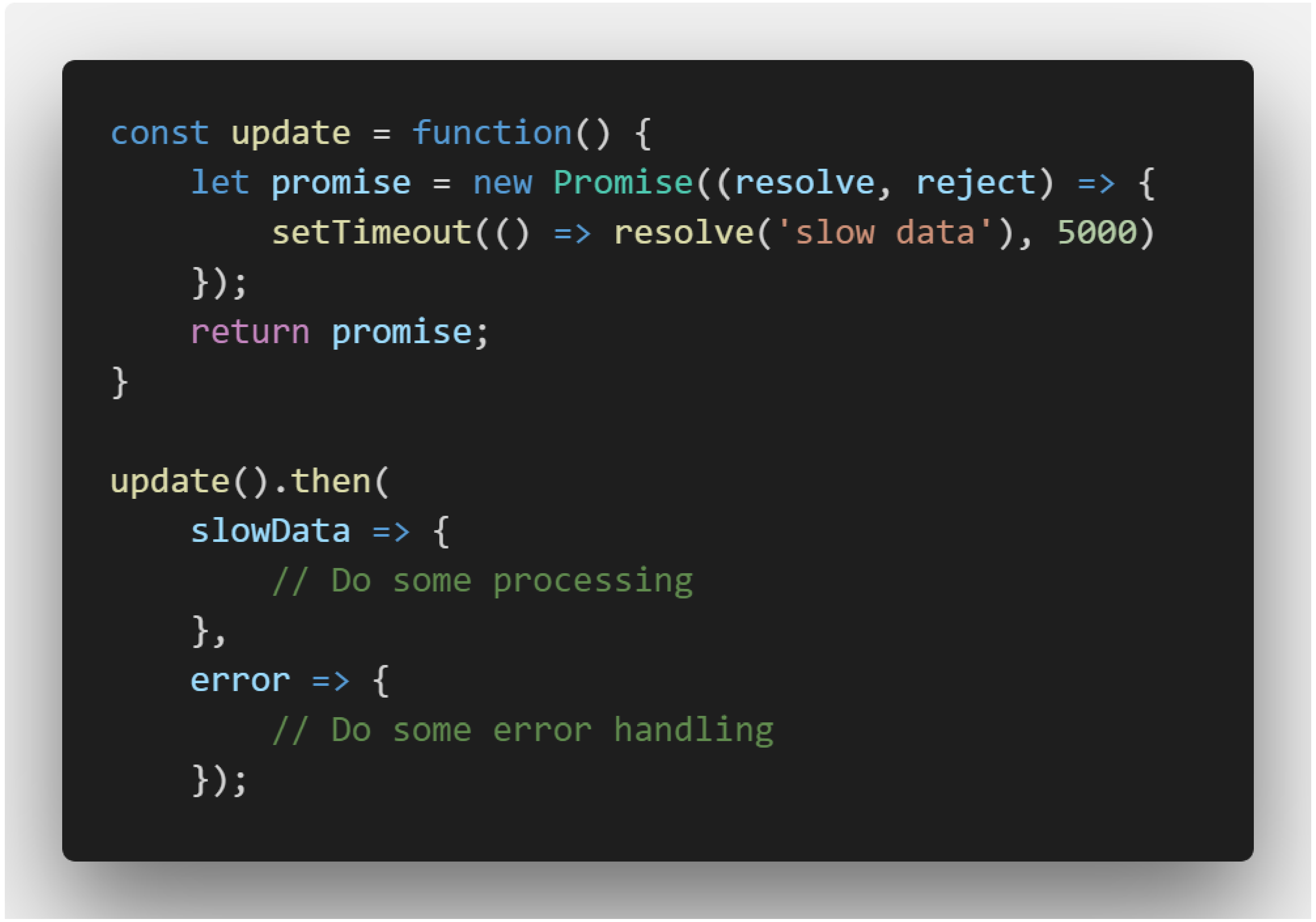
A Promise promises that it will be resolved

Uses observer pattern to populate the result

ES6 – Promise

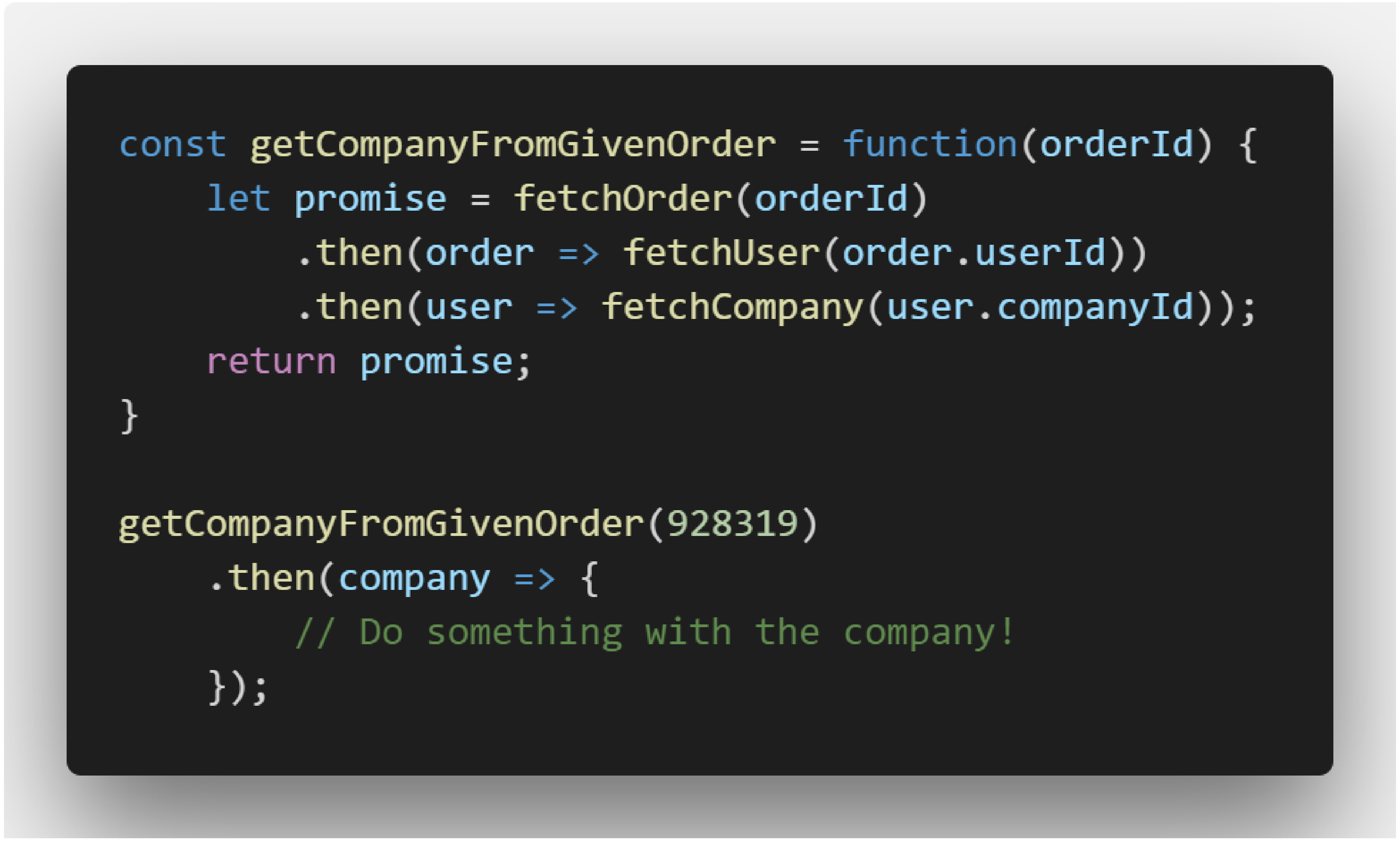

Promise
States:
• waiting
• resolved
• rejected
Listener 1
Listener 2
Listener 3
async function

ES6 – Promise (cont.)



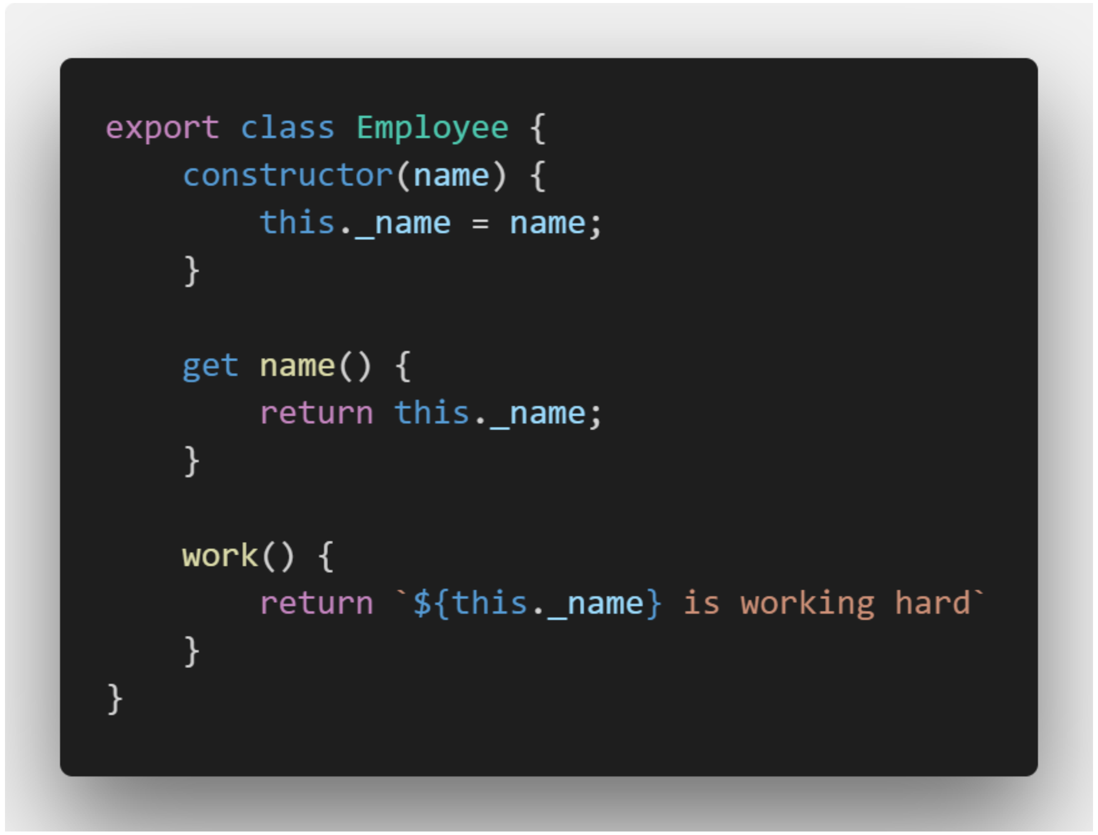
```
const update = function() {
    let promise = new Promise((resolve, reject) => {
        setTimeout(() => resolve('slow data'), 5000)
    });
    return promise;
}

update().then(
    slowData => {
        // Do some processing
    },
    error => {
        // Do some error handling
    });
```

ES6 – Promise chaining

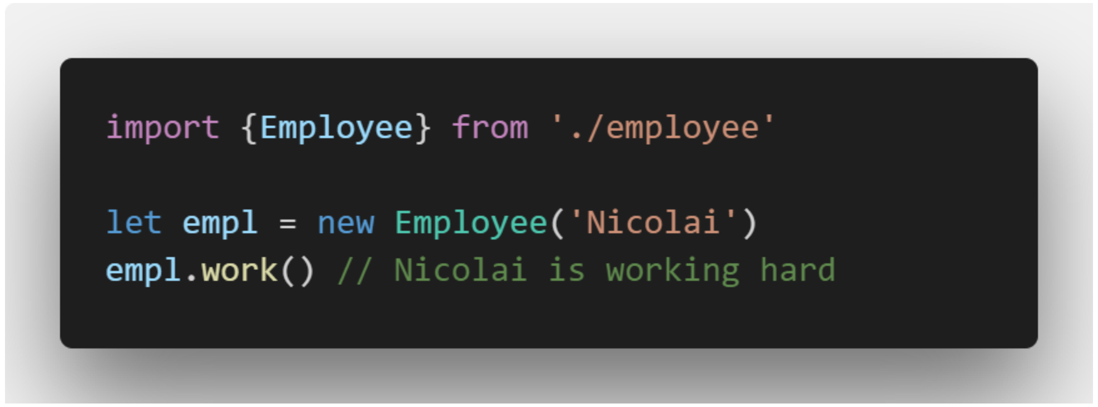


```javascript
const getCompanyFromGivenOrder = function(orderId) {
    let promise = fetchOrder(orderId)
        .then(order => fetchUser(order.userId))
        .then(user => fetchCompany(user.companyId));
    return promise;
}

getCompanyFromGivenOrder(928319)
    .then(company => {
        // Do something with the company!
    });
```

ES6 - Modules



```
export class Employee {
    constructor(name) {
        this._name = name;
    }

    get name() {
        return this._name;
    }

    work() {
        return `${this._name} is working hard`
    }
}
```



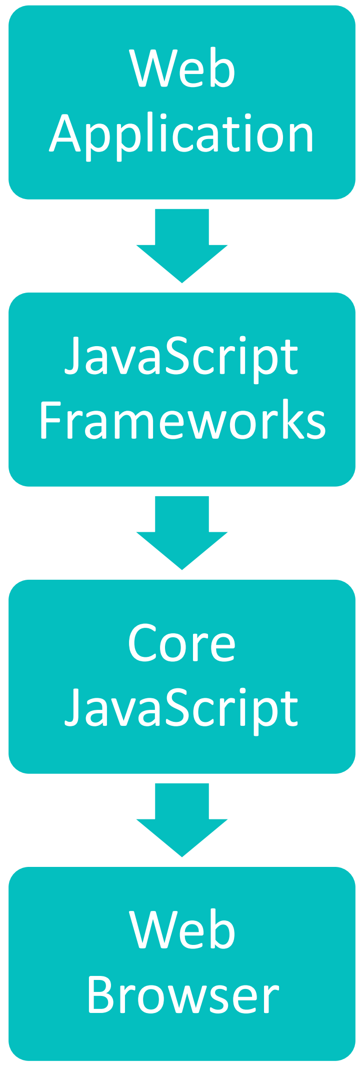
```
import {Employee} from './employee'

let empl = new Employee('Nicolai')
empl.work() // Nicolai is working hard
```

Native ES6 modules are not yet implemented

Use tools such as Browserify and Webpack

JavaScript Frameworks

Frameworks in the stack



Advantages

- Seamlessly handled cross browser issues
- Help speed up development
- Easy to learn and use
- Serves as a base for development of further libraries and frameworks

A few frameworks

ANGULARJS
by Google
node JS
METEOR
ext js
BACKBONE.JS
ember
Sencha
aurelia
Knockout.

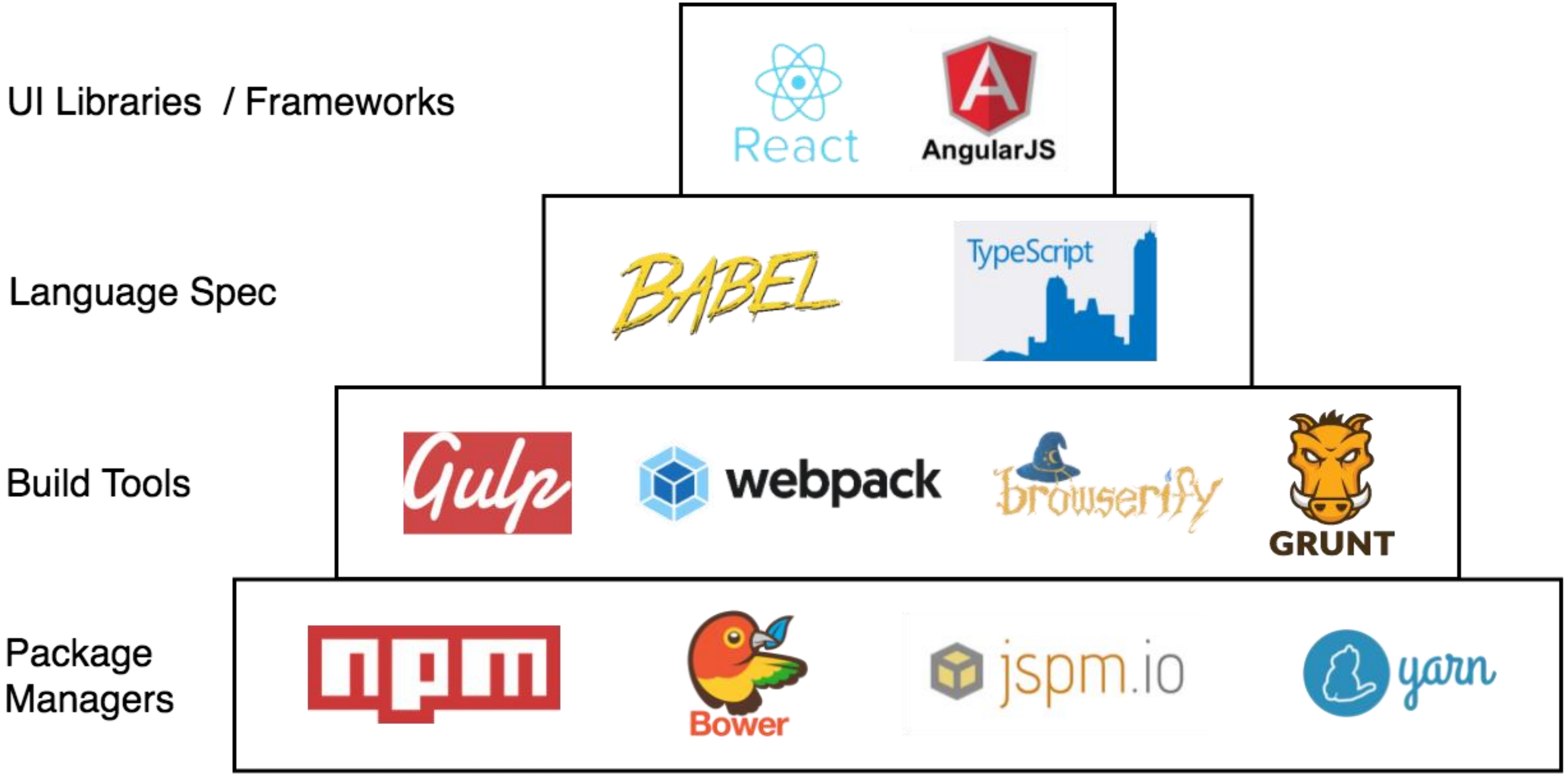
The choice

There are a lot of options when dealing with JavaScript Frameworks

This abundance of choices leads to confusion.

How do you pick the right library or framework?

The front-end stack



Core JavaScript engine

Every major browser vendor provides their own JavaScript engine

Varying degree of JavaScript support by different engines

Major engines:

- Chakra
- V8
- SpiderMonkey

DOM Libraries

- jQuery
- Mootools
- etc

Web App libraries

- Backbone JS
- Ember
- Angular
- React JS
- Vue JS

jQuery introduction

What is jQuery

Cross platform javascript library

Designed to simplify client side scripting

It is open source

It is the most widely deployed Javascript library (18.6% according to builtwith.com)

Intended to make navigating the DOM easier

Features

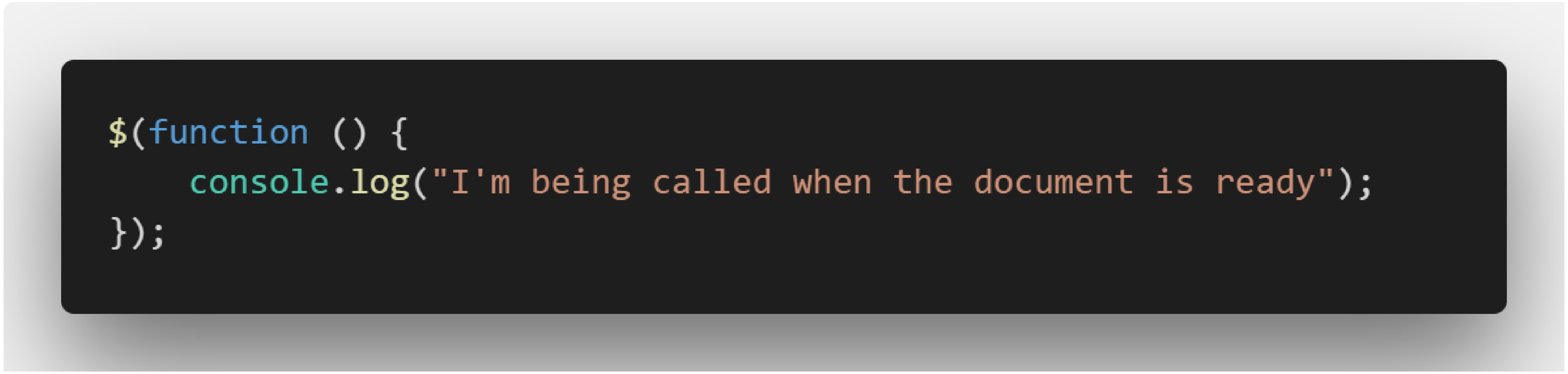
- DOM element selections using Sizzle
- DOM manipulation based on CSS selectors
- Events
- Effects and animations
- Ajax
- Deferred and Promise objects to control asynchronous processing
- JSON parsing
- Extensibility through plug-ins
- Utilities, such as feature detection
- Compatibility methods that are natively available in modern browsers, but need fall backs for older ones, such as inArray() and each()
- Multi-browser (not to be confused with cross-browser) support

Browser support

jQuery 2.x → current version – 1, i.e. the current version and the version before.

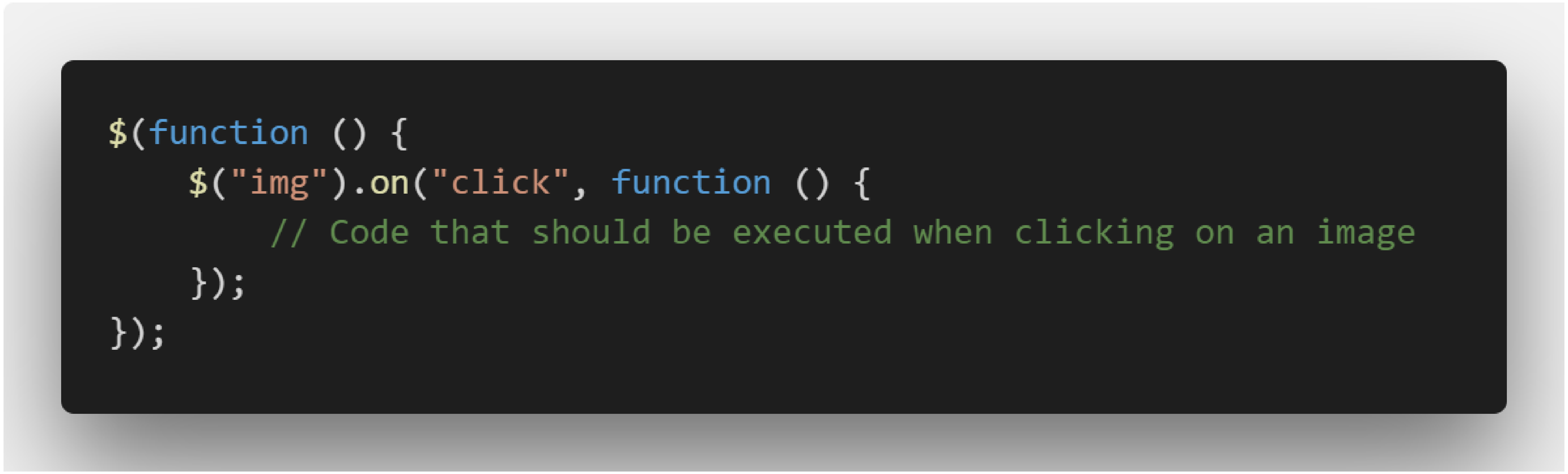
jQuery 1.x → Internet Explorer 6 an above.

Example



```
$(function () {
    console.log("I'm being called when the document is ready");
});
```

Events



```
$(function () {
    $("img").on("click", function () {
        // Code that should be executed when clicking on an image
    });
});
```

Chaining

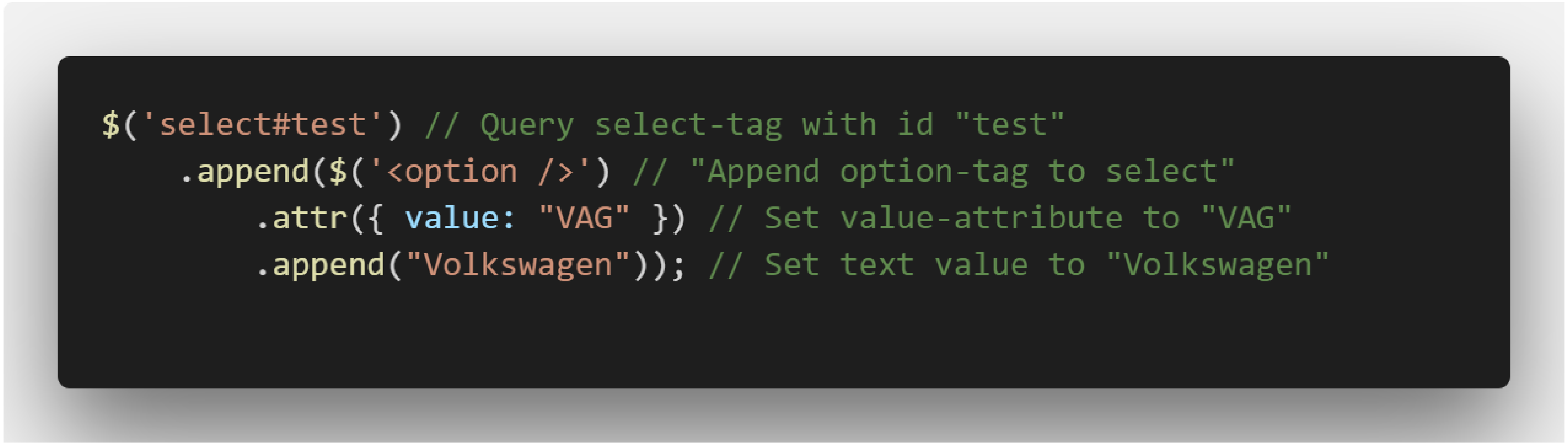
Almost all jQuery functions return a jQuery object and therefore chaining is possible.

```
$('div.test') // Find all div-tags with the class "test"
    .add('p.quote') // Finds all p-tags with "quote" class inside divs
    .addClass('blue'); // Adds the class "blue" to aforementioned p-tags and div-tags
```

Some functions return a pure value:

```
$('div.test').val() // Returns the raw value of the div-tags
```

Manipulating the DOM



```
$('select#test') // Query select-tag with id "test"
    .append($('<option />') // "Append option-tag to select"
        .attr({ value: "VAG" }) // Set value-attribute to "VAG"
        .append("Volkswagen")); // Set text value to "Volkswagen"
```

AJAX

```
$.ajax({
    type: 'POST', // GET, POST, UPDATE, DELETE etc.
    url: '/api/login', // URL to remote endpoint
    data: { // Data forwarded to remote endpoint
        username: 'nbo',
        password: 'lolno',
    },
}).done(function (msg) { // On succes, fire alert message
    alert('Data Saved: ' + msg);
}).fail(function (xmlHttpRequest, statusText, errorThrown) { // On failure
    alert('Your form submission failed.');
});
```

Exercise

Go to https://codepen.io/NicolaiOksen/pen/mxOePN

Click on **Fork** (top right corner)

- Create an event handler for **click, mouseenter** and **mouseleave**
  - What happens is up to you to decide
- Try selecting different elements
- Try manipulation the DOM with jQuery

20 minutes

React JS
An introduction

What is React

Javascript library for building user interfaces

Open source

Developed and maintained by Facebook and individual developers

Used on many different sites:

Netflix, Imgur, Bleacher Report, Feedly, Airbnb, SeatGeek, HelloSign, Walmart

Features

- Component based
- One way data flow
  - Properties are immutable
  - "properties flow down; actions flow up"
- Virtual DOM
  - In memory copy of the DOM
  - Allows React to find differences and update only required bits.
  - Much faster and efficient than jQuery
- JSX
  - More later
- Architecture beyond HTML
- React Native
  - Using React to develop native applications for iOS, Android and UWP

Browser support

Supports all major browser

From Internet Explorer 9 and above

Requires ECMAScript 5 support

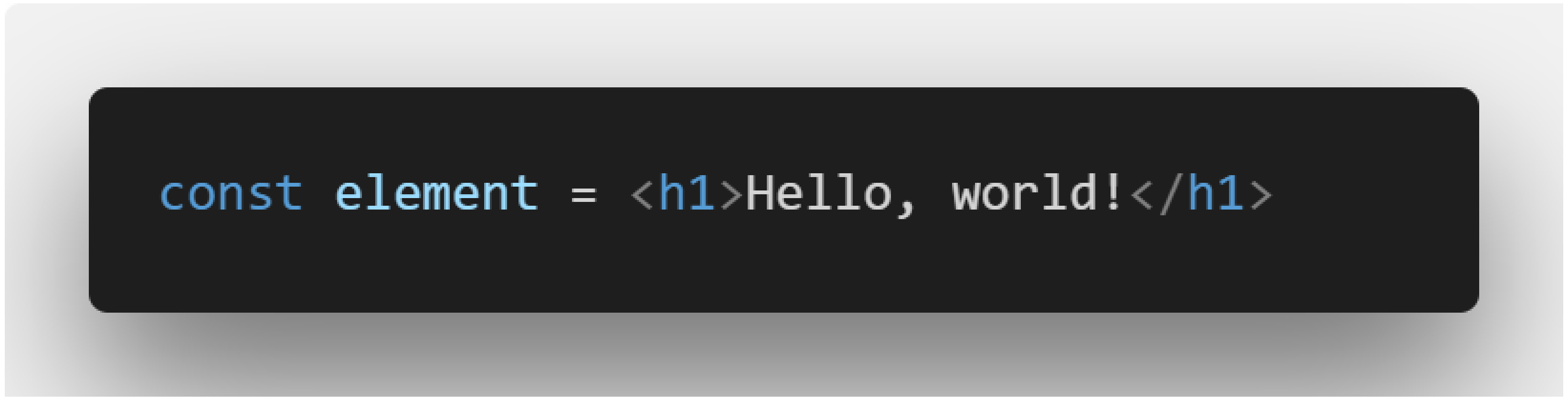
Works best with a proper build line and ES6

JSX

An extension to Javascript

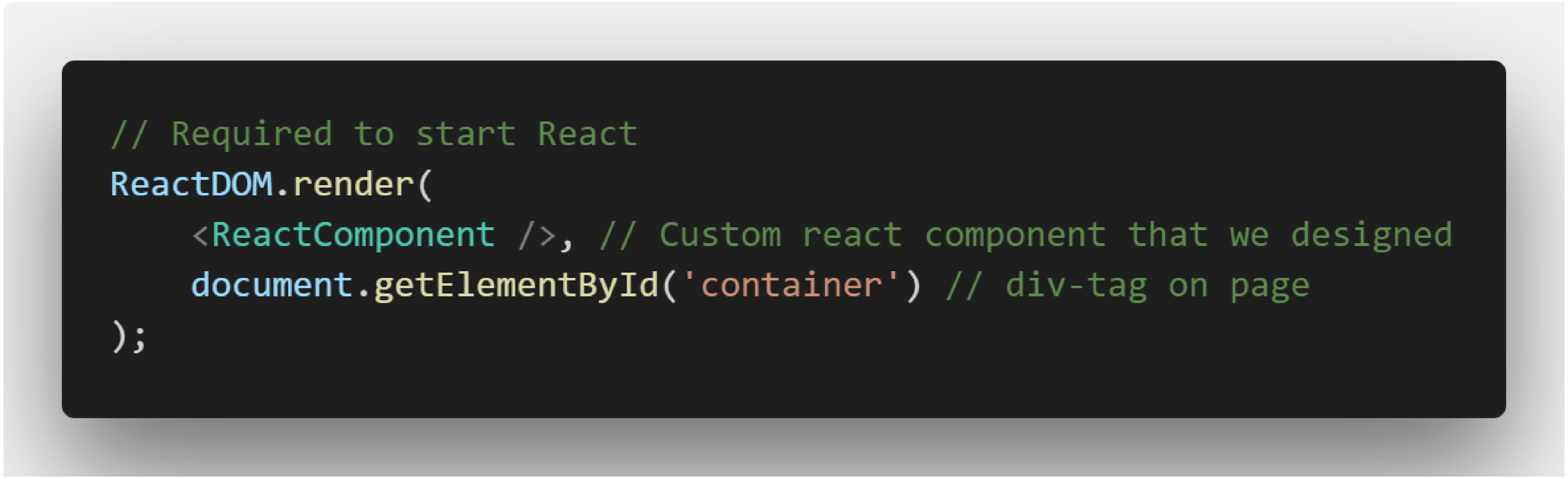
Allows using HTML-like markup in Javascript to create components

Example:



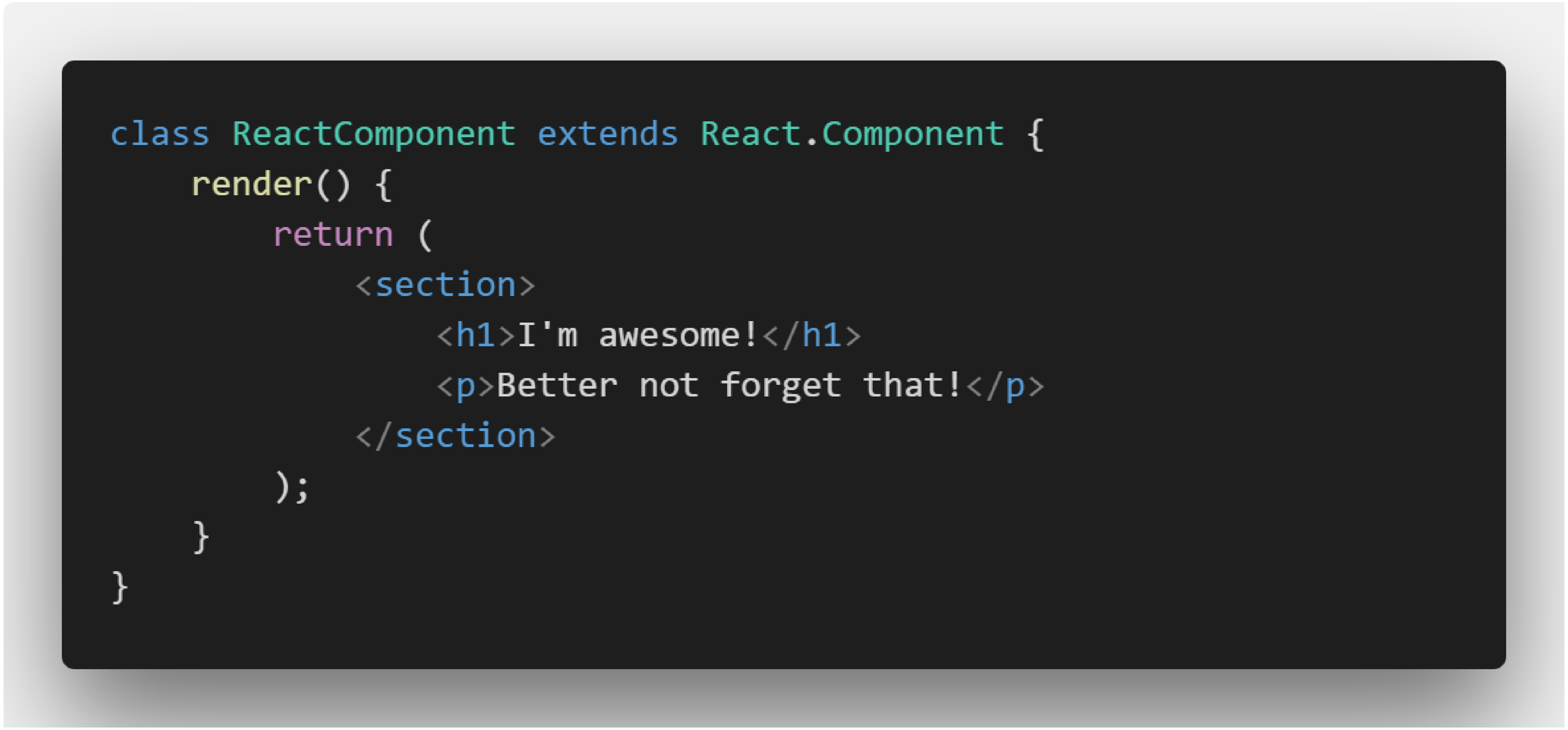
```
const element = <h1>Hello, world!</h1>
```

Using React



```
// Required to start React
ReactDOM.render(
    <ReactComponent />, // Custom react component that we designed
    document.getElementById('container') // div-tag on page
);
```

A Component in React


```
class ReactComponent extends React.Component {
    render() {
        return (
            <section>
                <h1>I'm awesome!</h1>
                <p>Better not forget that!</p>
            </section>
        );
    }
}
```

JSX is complex

React and JSX can be quite hard to grasp

Best way to learn is ”by doing”

Codecademy has a course on React

Vue JS
An Introduction

What is Vue JS

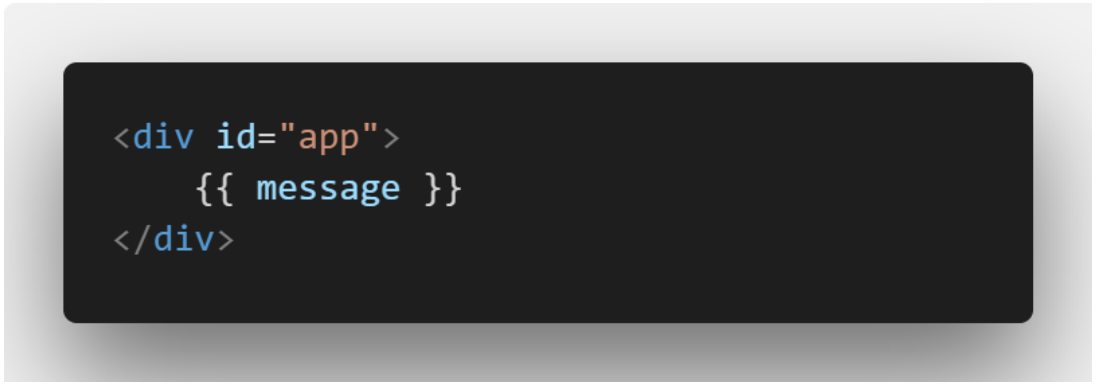
Open source

Progressive JavaScript Framework for building user interfaces

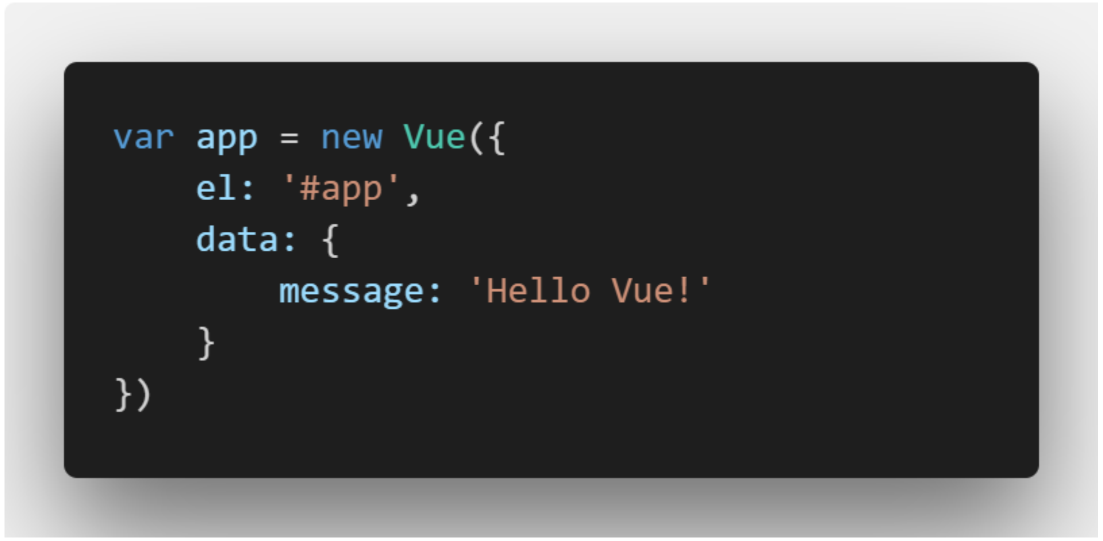
Component oriented

Declarative Rendering

At the core of Vue.js is a system that enables us to declaratively render data to the DOM using straightforward template syntax:



```
<div id="app">
    {{ message }}
</div>
```

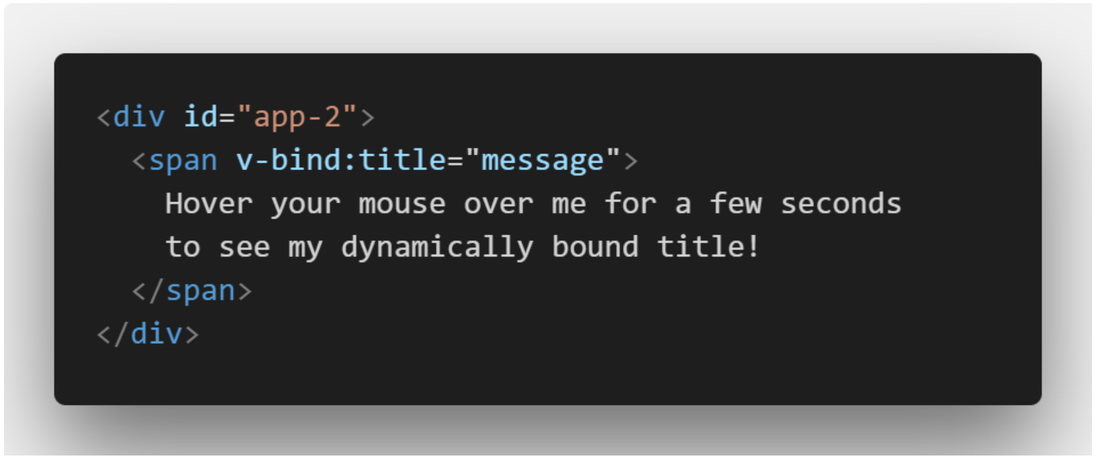


```
var app = new Vue({
    el: '#app',
    data: {
        message: 'Hello Vue!'
    }
})
```

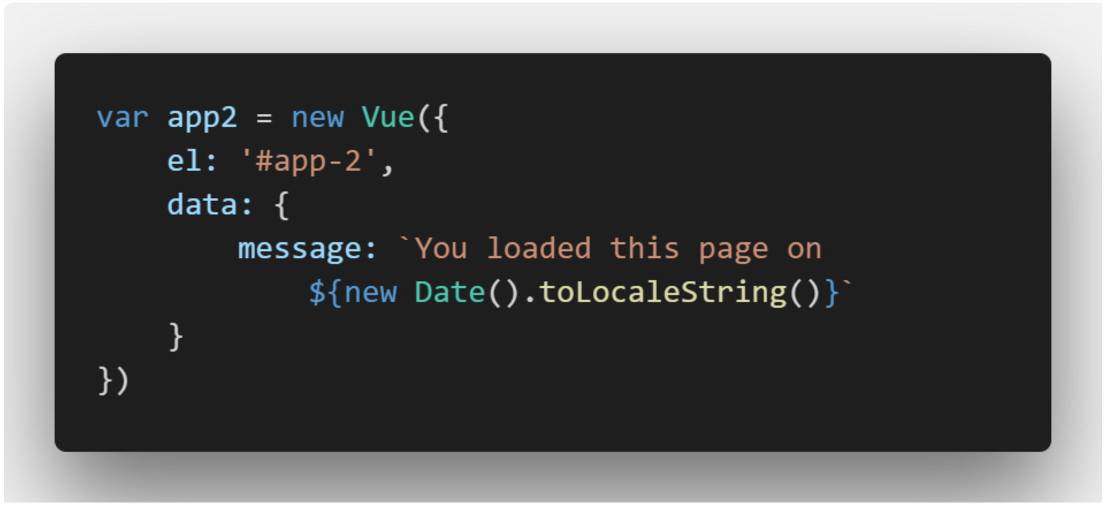
Binding attributes

With vue we can also populate attributes dynamically with some Vue magic

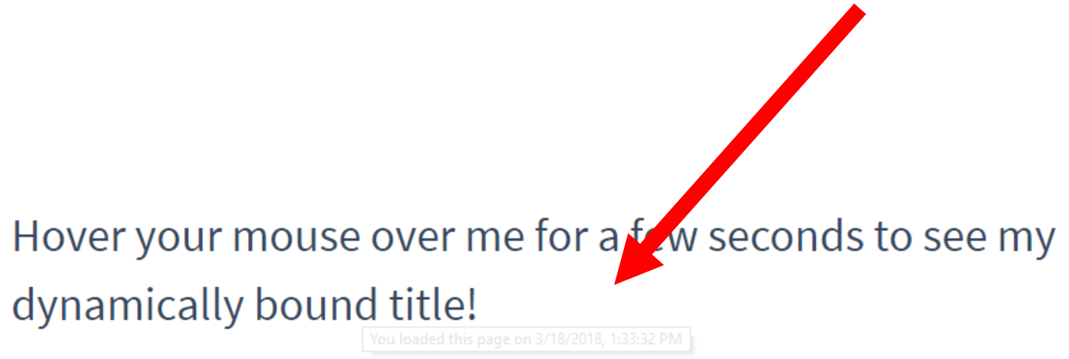
v-bind is a directive.



```
<div id="app-2">
  <span v-bind:title="message">
    Hover your mouse over me for a few seconds
    to see my dynamically bound title!
  </span>
</div>
```



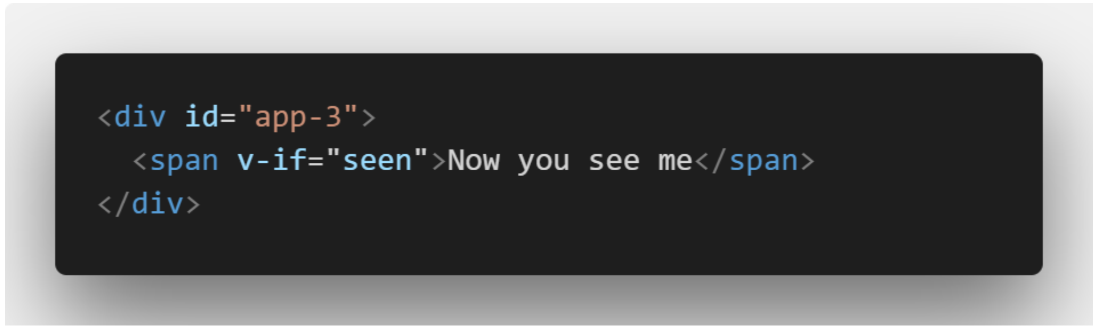
```
var app2 = new Vue({
    el: '#app-2',
    data: {
        message: `You loaded this page on
            ${new Date().toLocaleString()}`
    }
})
```



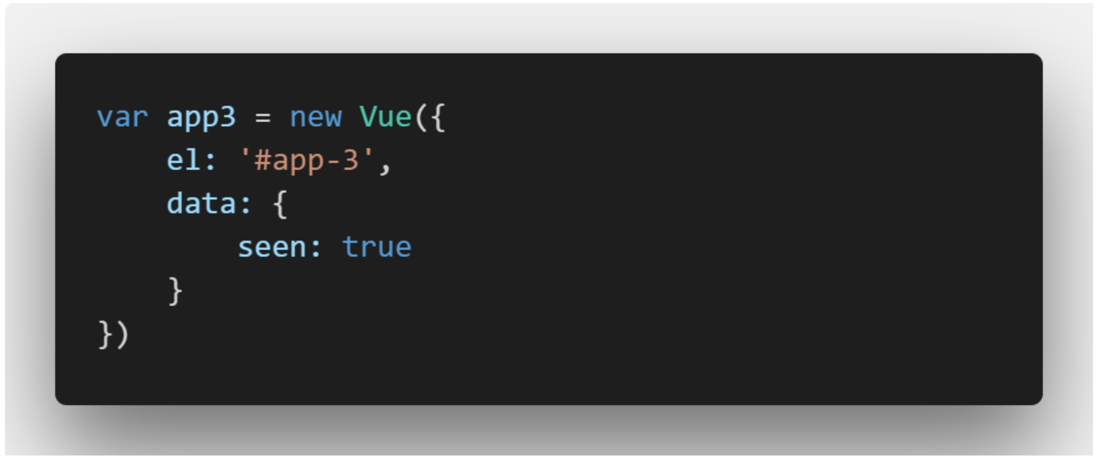
Hover your mouse over me for a few seconds to see my dynamically bound title!

You loaded this page on 3/18/2018, 1:33:32 PM

Conditionals



```
<div id="app-3">
  <span v-if="seen">Now you see me</span>
</div>
```



```
var app3 = new Vue({
    el: '#app-3',
    data: {
        seen: true
    }
})
```

Using conditionals it is easy to toggle the presence of a given element.

By going to your browsers console you can change the visibility:

app3.seen = false

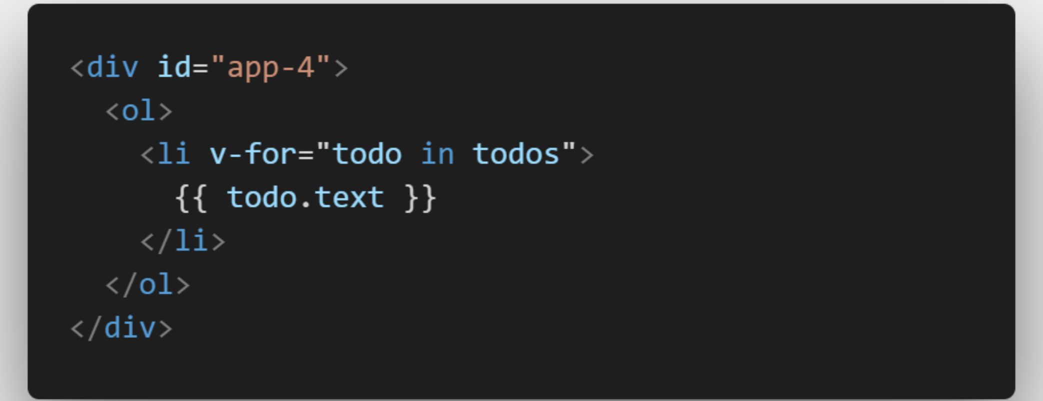
Loops

v-for is another directive, allowing use to loops over arrays and create multiple elements.

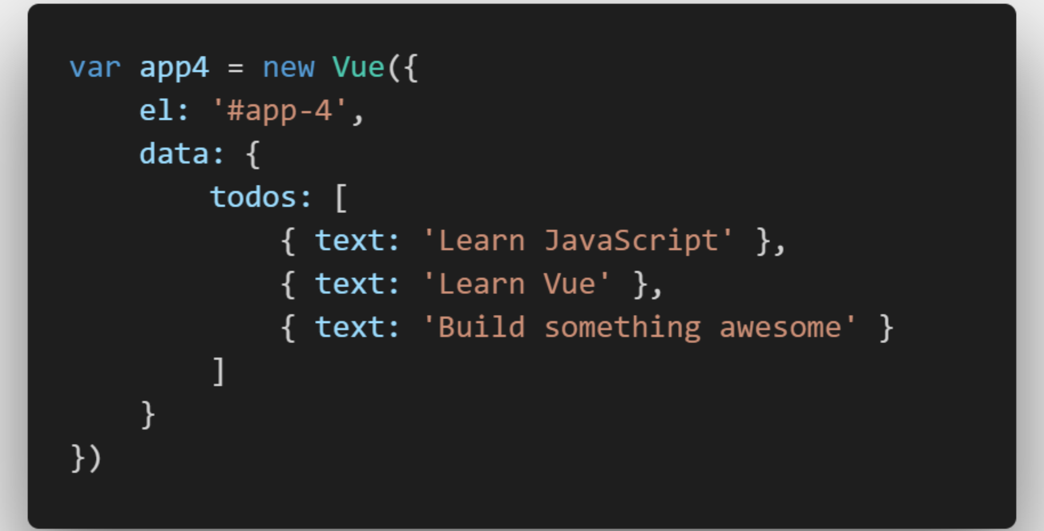
Result:

1. Learn JavaScript
2. Learn Vue
3. Build something awesome

In the console, try:
app4.todos.push({ text: 'New item' })



```html
<div id="app-4">
  <ol>
    <li v-for="todo in todos">
      {{ todo.text }}
    </li>
  </ol>
</div>
```



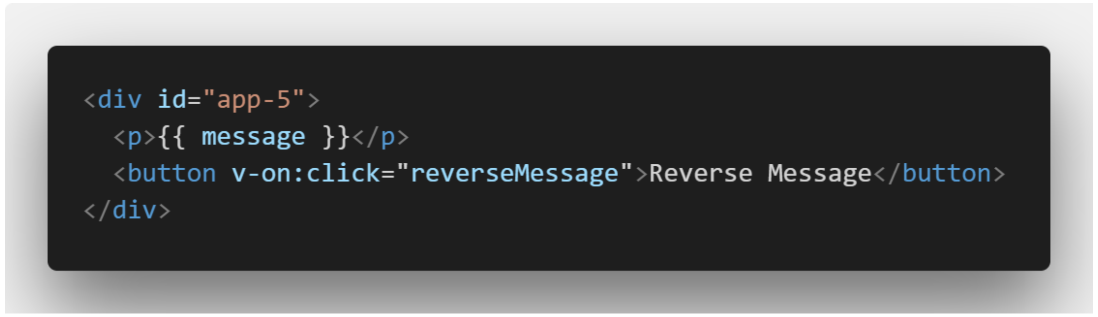
```js
var app4 = new Vue({
    el: '#app-4',
    data: {
        todos: [
            { text: 'Learn JavaScript' },
            { text: 'Learn Vue' },
            { text: 'Build something awesome' }
        ]
    }
})
```

Events

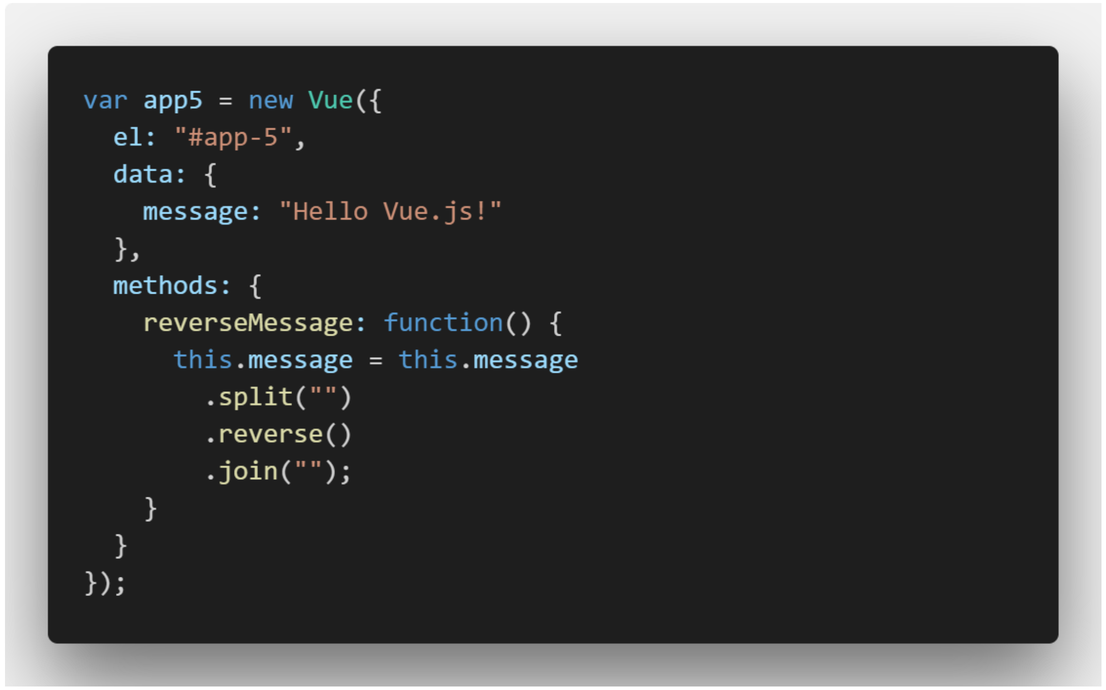
To let users interact with your app, we can use the **v-on** directive to attach event listeners that invoke methods on our Vue instances.

Manipulating the DOM, is all handled by Vue.

This lets us focus on code logic not the DOM



```html
<div id="app-5">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">Reverse Message</button>
</div>
```

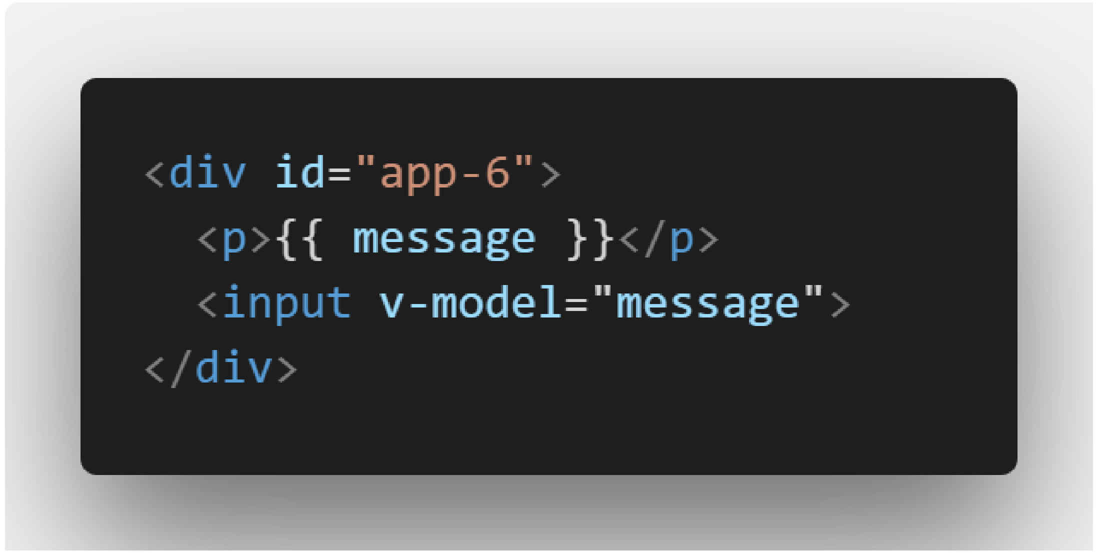


```javascript
var app5 = new Vue({
  el: "#app-5",
  data: {
    message: "Hello Vue.js!"
  },
  methods: {
    reverseMessage: function() {
      this.message = this.message
        .split("")
        .reverse()
        .join("");
    }
  }
});
```

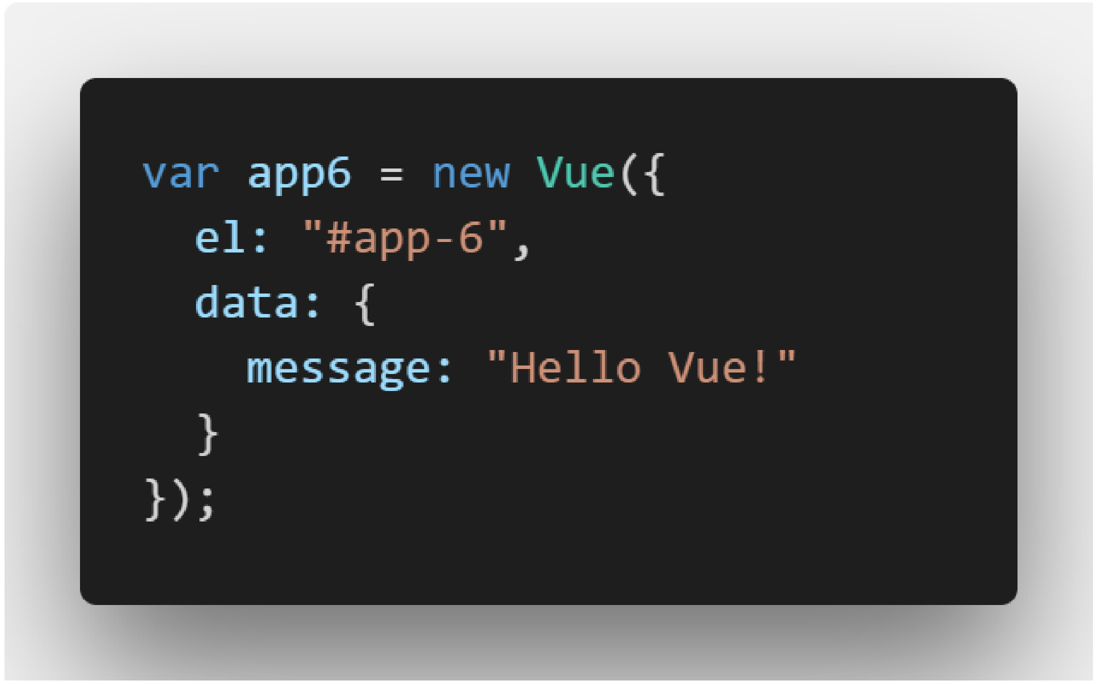
Two-way binding

By using **v-model** we allow our web app to use two way binding.

Try it out:
https://codepen.io/NicolaiOksen/pen/bvBVXr



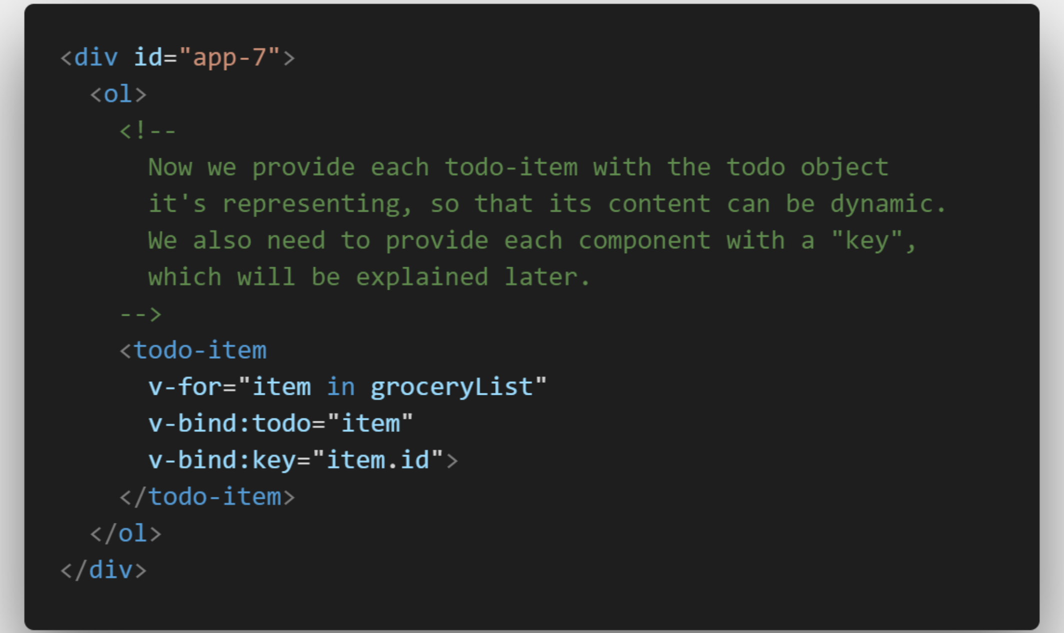
```
<div id="app-6">
  <p>{{ message }}</p>
  <input v-model="message">
</div>
```



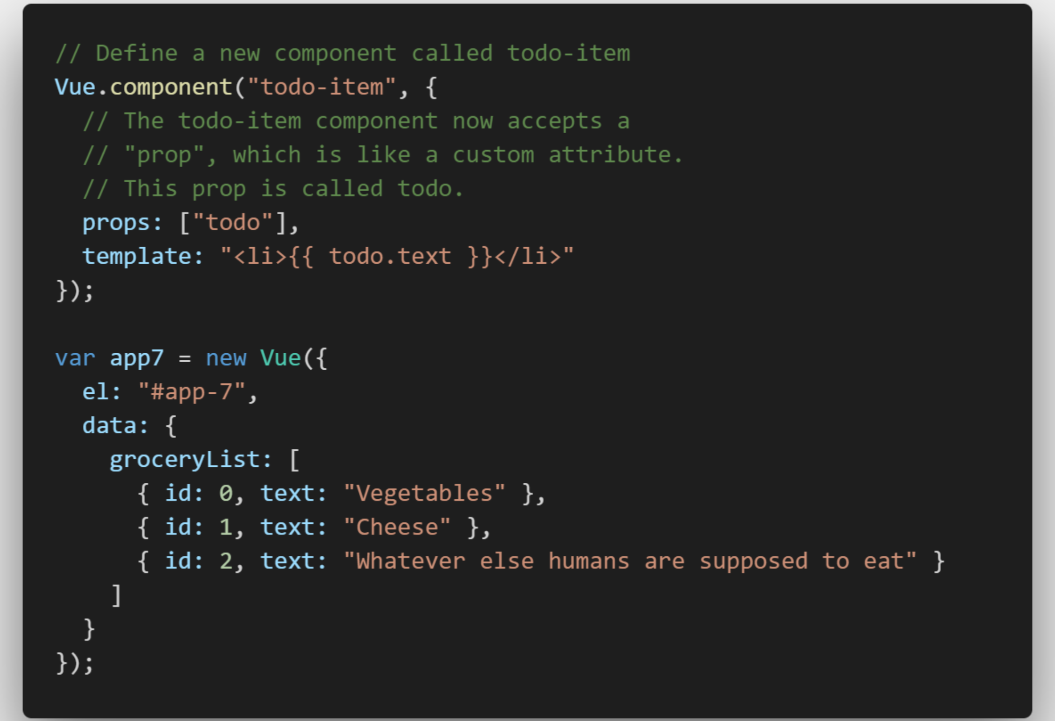
```
var app6 = new Vue({
  el: "#app-6",
  data: {
    message: "Hello Vue!"
  }
});
```

Components

A component is a small, self-contained and reusable piece of code.



```html
<div id="app-7">
  <ol>
    <!--
      Now we provide each todo-item with the todo object
      it's representing, so that its content can be dynamic.
      We also need to provide each component with a "key",
      which will be explained later.
    -->
    <todo-item
      v-for="item in groceryList"
      v-bind:todo="item"
      v-bind:key="item.id">
    </todo-item>
  </ol>
</div>
```



```javascript
// Define a new component called todo-item
Vue.component("todo-item", {
  // The todo-item component now accepts a
  // "prop", which is like a custom attribute.
  // This prop is called todo.
  props: ["todo"],
  template: "<li>{{ todo.text }}</li>"
});

var app7 = new Vue({
  el: "#app-7",
  data: {
    groceryList: [
      { id: 0, text: "Vegetables" },
      { id: 1, text: "Cheese" },
      { id: 2, text: "Whatever else humans are supposed to eat" }
    ]
  }
});
```

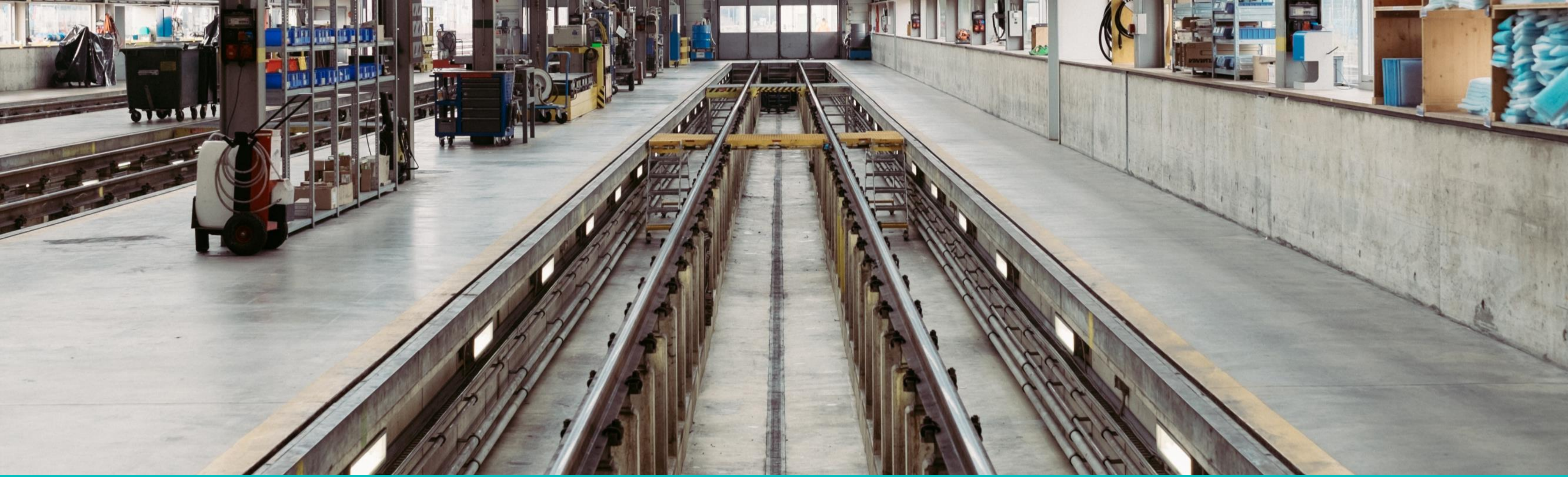
Exercise

Go to https://codepen.io/NicolaiOksen/pen/KoNdoX

Click on **Fork** (top right corner)

- Change HTML to get data from Vue
- Make an click event and change the background color of something

20 minutes



Web Assembly

What’s next?

As mentioned earlier, JavaScript is the only native language of browsers

That is not entirely true today…

Web Assembly is a way to compile a language to bytecode and execute directly in the browser

What is Web Assembly

Denoted as WASM

It is supported by roughly 72% of all global browser

An improvement to JavaScript:

- Implement your performance critical stuff in WASM and import it like a standard JavaScript module.

A new language:

- WebAssembly code defines an AST (Abstract Syntax Tree) represented in a binary format. You can author and debug in a text format so it's readable.

A browser improvement:

- Browsers will understand the binary format, which means we'll be able to compile binary bundles that compress smaller than the text JavaScript we use today. Smaller payloads mean faster delivery. Depending on compile-time optimization opportunities, WebAssembly bundles may run faster than JavaScript, too!

A Compile Target:

- A way for other languages to get first-class binary support across the entire web platform stack.

What will it be used for?

Threaded operations

Single Instruction Multiple Data – SIMD

In essence, parallel operations

WebAssembly fills in the gaps that would be awkward to fill with JavaScript.

A bit of history

June 2015 – WebAssembly was announced

March 2016 – Google, Microsoft and Mozilla preview WASM in their browsers

October 2016 – WebAssemly becomes a binary release candidate

March 2017 – Begins to be shipped on-by-default in major browsers

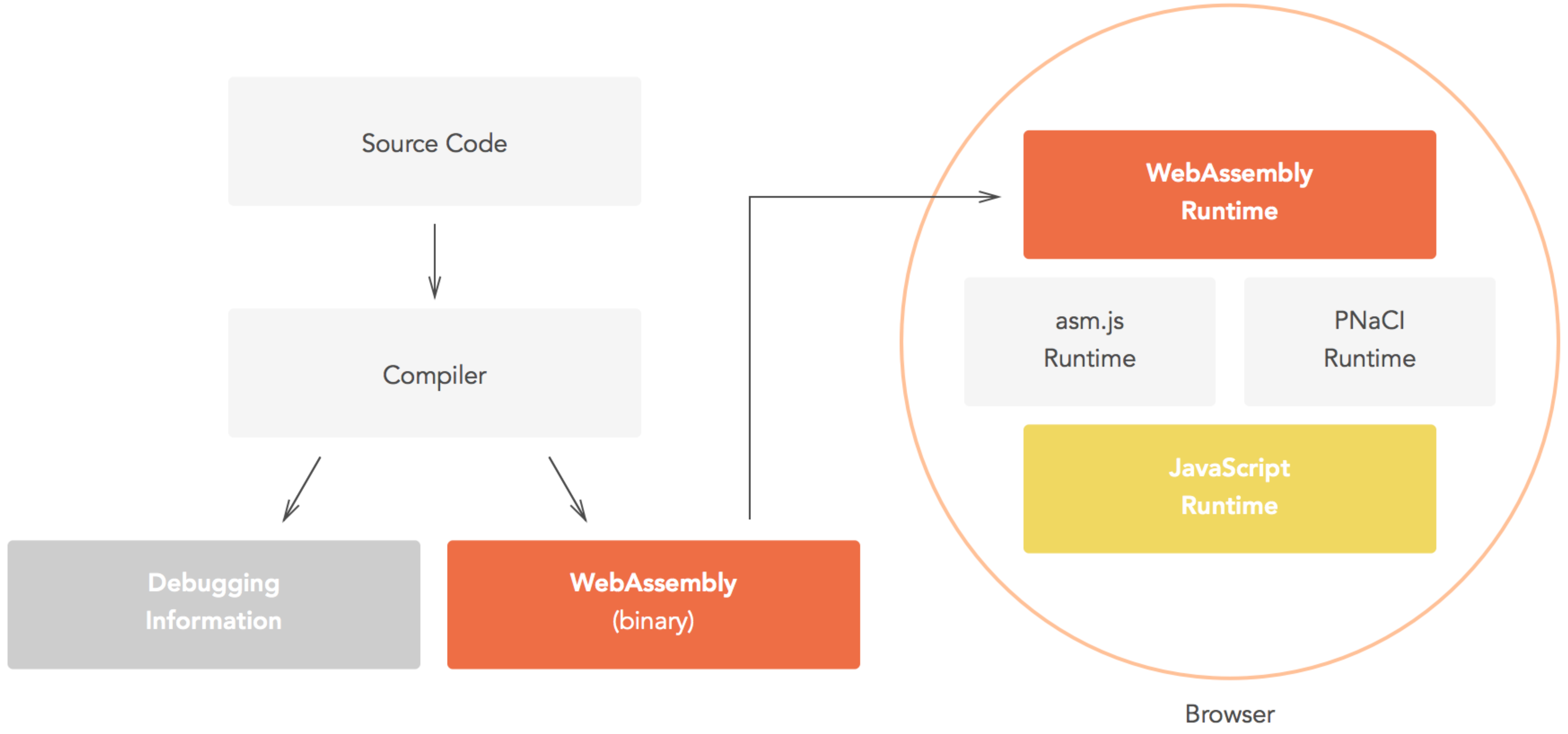
To binary

WebAssembly compiles into a Abstract Syntax Tree (AST)

In a compiler the AST keeps source location information and some typing information

WebAssembly gets compiled to binary AST from languages like Haskell, C++ or C

To binary… (cont.)



Advantages

You’re able to use other languages then JavaScript in the browser

In WASM you can author and debug in a text format, thereby it is readable

We talking about a new low-level language in the spirit of Assembler

Now you ask… Doesn’t ASM.js do this?

Well, no…

ASM.js is a low-level subset of JavaScript

It appears to have direct memory register access, but it is still parsed by JavaScript

Therefore you still have the overhead of JavaScript

WebAssembly is bypassing this by directly using the AST in a binary format (yay!)

The bottleneck that is JavaScript

JavaScript is restricted in its flexibility

WebAssembly works at the “bare metal” memory layer in the browser

WebAssembly is filling the holes left by JavaScript, in terms of controlling bit/byte level memory register control

JavaScript simply cannot get to that low-level.

But the awesome part… You are able to compile other languages for the browser!

JavaScript out, WebAssembly in?

WebAssembly is not a repleacement for JavaScript

JavaScript is still needed for accessing the DOM

They are to be used in collaboration with eachother

WebAssembly and C#

So, is it possible to use C# in the browser?

Yes it is! It is called Blazor! Take a look here: https://github.com/aspnet/blazor

Questions?