PV247

Course organization

18.9.	Modern JS stack
25.9	React part 1
2.10	React part 2
16.10	Redux
23.10	Promises, router
30.10	Automated tests
6.11	Agile/Scrum
13.11	Lean software development
20.11	ITIL
27.11	Google Ventures sprint
4.12	Modern soft skills
11.12	DevOps

Kentico

Modern JS stack

Slavomír Moroz

Prerequisites

- Html
- CSS
- Git
- HTTP
  - Methods
  - Statuses
  - Query string
  - Headers
  - State (cookies, session, local storage)
- REST

JavaScript basics

- Prototypes
- Closures
- “this” & “arguments” context
- Undeclared vs undefined vs null
- Function / method invocation
- Constructor functions
- Curry (apply, call, bind)

- JavaScript The Good Parts by Douglas Crockford (first 60 pages)
- Functional programming in JavaScript (video)

ECMAScript

ECMAScript: A language standardized by ECMA International.
JavaScript: The commonly used name for implementations of the ECMAScript standard.

ECMAScript 5 (ES5): The 5th edition of ECMAScript, standardized in 2009. This standard has been implemented fairly completely in all modern browsers

ECMAScript 6 (ES6)/ ECMAScript 2015 (ES2015): The 6th edition of ECMAScript, standardized in 2015. This standard has been partially implemented in most modern browsers. To see the state of implementation by different browsers and tools, check out these compatibility tables.

ECMAScript 7 / 2016
ECMAScript 8 / 2017

ES6 new features

- Constants, let
- Default arguments
- String interpolation
- Property shorthands
- Spread operators
- Object & array destructuring
- Arrow functions (lambda expressions)
- Promises
- Classes & class inheritance

All features - http://es6-features.org

https://egghead.io/courses/learn-es6-ecmascript-2015

Jose Aguinaga Follow
Web Engineer. UX & Interactions Designer. Full-Stack Javascript Developer. Previously @numbrs, @plaidhq
Oct 3 · 12 min read

How it feels to learn JavaScript in 2016

https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f

React

Redux

node JS

ESLint

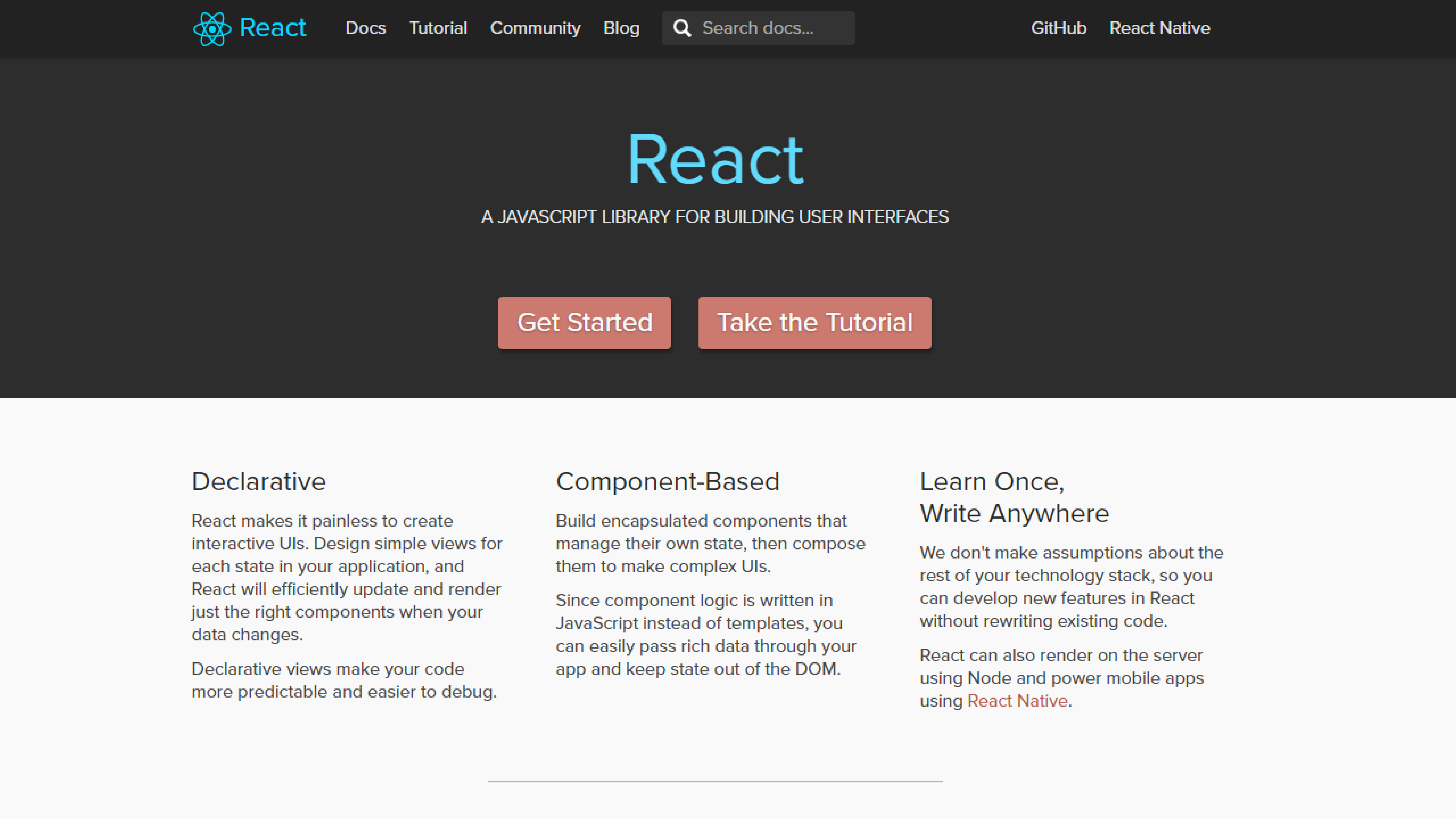
webpack
MODULE BUNDLER

BABEL

npm

BABEL

https://babeljs.io



React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

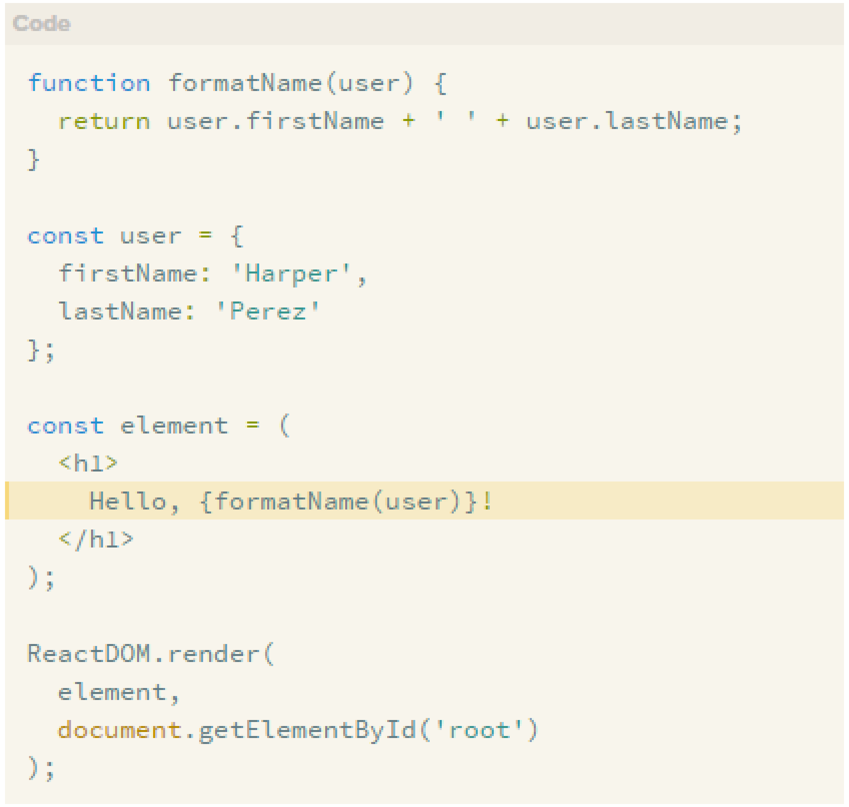
Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.



Code

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

https://facebook.github.io/react/docs/introducing-jsx.html

CommonJS (Node.js)

```
function myModule() {
  this.hello = function() {
    return 'hello!';
  }

  this.goodbye = function()
    return 'goodbye!';
  }
}

module.exports = myModule;
```

```
var myModule = require('myModule');

var myModuleInstance = new myModule();
myModuleInstance.hello(); // 'hello!'
myModuleInstance.goodbye(); // 'goodbye!'
```

AMD (Browser)

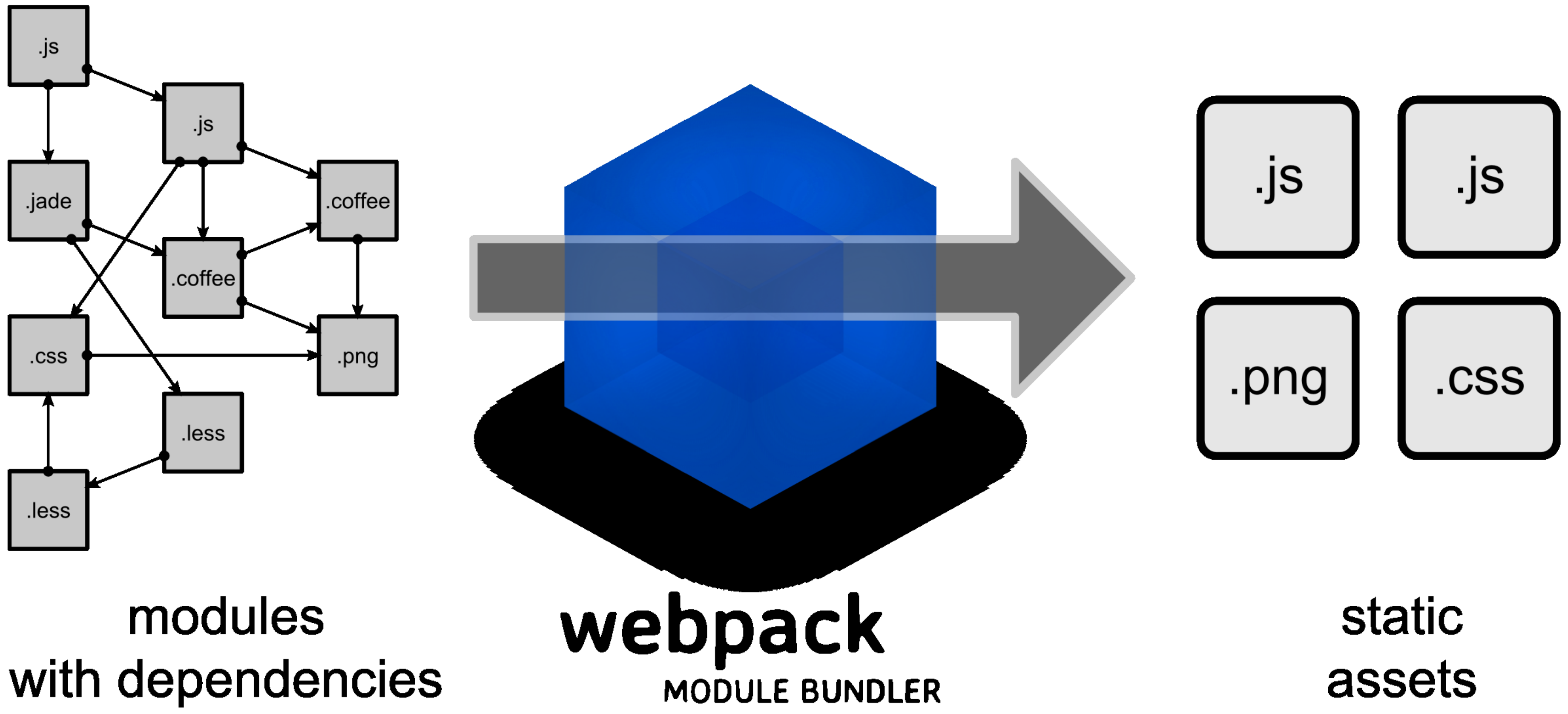
```
define([], function() {

  return {
    hello: function() {
      console.log('hello');
    },
    goodbye: function() {
      console.log('goodbye');
    }
  };
});
```

```
define(['myModule', 'myOtherModule'], function(myModule, myOtherModule) {
  console.log(myModule.hello());
});
```

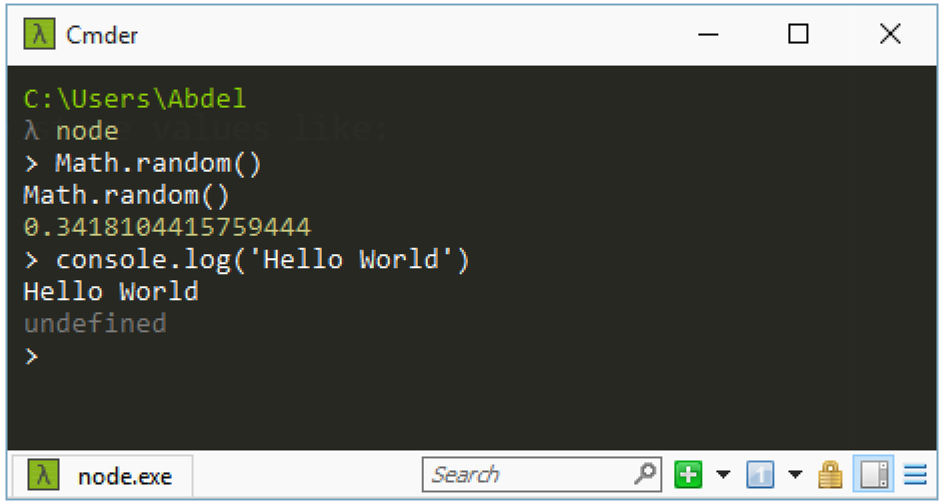
UMD

```
(function (root, factory) {
  if (typeof define === 'function' && define.amd) {
      // AMD
    define(['myModule', 'myOtherModule'], factory);
  } else if (typeof exports === 'object') {
      // CommonJS
    module.exports = factory(require('myModule'), require('myOtherModule'))
  } else {
    // Browser globals (Note: root is window)
    root.returnExports = factory(root.myModule, root.myOtherModule);
  }
}(this, function (myModule, myOtherModule) {
  // Methods
  function notHelloOrGoodbye(){}; // A private method
```



https://webpack.js.org/concepts/

- npm init
- npm install
- npm run ...



Cmder

```
C:\Users\Abdel
λ node
> Math.random()
Math.random()
0.3418104415759444
> console.log('Hello World')
Hello World
undefined
>
```

https://docs.npmjs.com/getting-started/what-is-npm

ESLint

tool for identifying and reporting on patterns found in ECMAScript/JavaScript code, with the goal of making code more consistent and avoiding bugs.

http://eslint.org/docs/user-guide/getting-started

Debug

When debugging in a browser, how to tell where the original code is?

- Source maps
  - Mapping between the original and the transformed source code.
  - Inline
    - Add the mapping data directly to the generated files.
    - Thanks to their speed, inline source maps are ideal for development.
  - Separate
    - Emit the mapping data to separate source map files and link the original source to them using a comment.
    - Given they make the bundles big, separate source maps are the preferable solution for production.

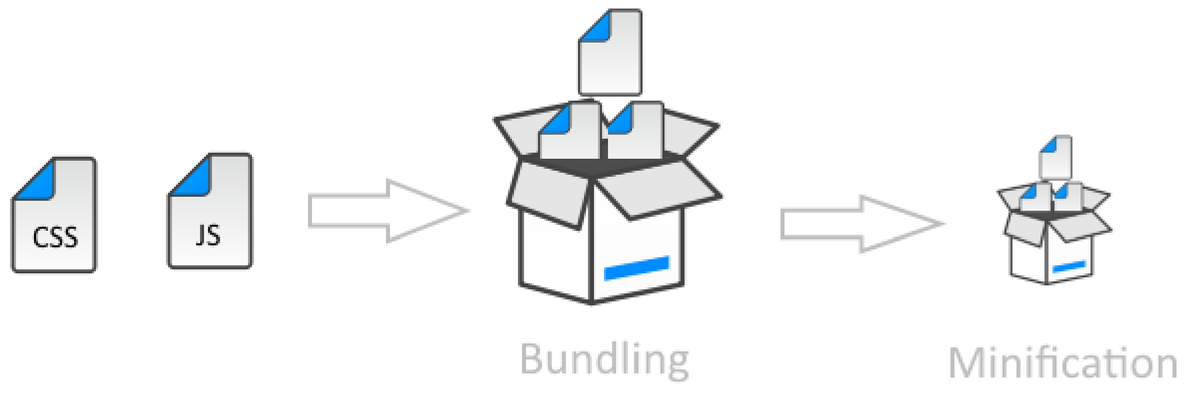
It's possible you don't want to generate a source map for your production bundle as this makes it effortless to inspect your application. By disabling them you are performing a sort of obfuscation.

[Chrome] Search in source files: CTRL + P

https://survivejs.com/webpack/building/source-maps
https://webpack.js.org/configuration/devtool/

Minification

- process of removing all characters that are not necessary from the Javascript source code



UglifyJS Webpack Plugin
https://github.com/webpack-contrib/uglifyjs-webpack-plugin

Resources

https://github.com/facebookincubator/create-react-app