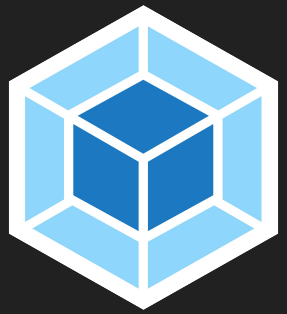
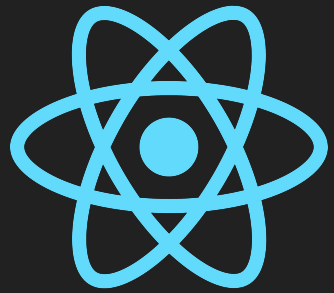
Development Cycle

For

React





By Maarten van Ittersum

Who am I?

Maarten van Ittersum

maarten@v-ittersum.nl

/dezemand

What we’re discussing today

1. What is React
2. How to develop with React
3. Live action example
4. Post-development

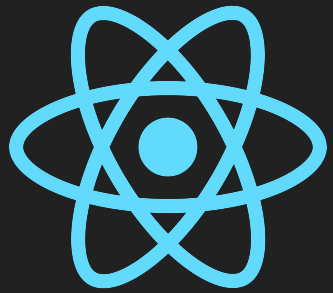
Code, presentation and everything else is available at:

/**dezemand/tutorial-react**

Issues and PRs are welcome!

What is **React**

- Created by **Facebook** in **2011**
- Front-end Library
- Divide the front-end up in **components**
- XML in JavaScript with **JSX**
- Popular in both **Big Tech** and **startups**



Documentation available at
reactjs.org

React has changed...

```
var MyComponent = React.createClass({
  componentDidMount() {
    console.log("I have mounted!");
  },
  componentWillUnmount() {
    console.log("I will unmount now, bye!");
  },
  render() {
    return (
      <div>
        <h1>Hello world!</h1>
        <p>How is everyone doing today?</p>
      </div>
    );
  }
});
```

React.createClass
Introduced at first release
Deprecated in version 15.5.0 (April 2017)

```
class MyComponent extends React.Component {
  componentDidMount() {
    console.log("I have mounted!");
  }
  componentWillUnmount() {
    console.log("I will unmount now, bye!");
  }
  render() {
    return (
      <div>
        <h1>Hello world!</h1>
        <p>How is everyone doing today?</p>
      </div>
    );
  }
}
```

React.Component (ES6 class)
Introduced in version 0.13.0 (March 2015)
Still used!

...for the better

```
const MyComponent = () => {
  useEffect(() => {
    console.log("I have mounted!");
    return () => {
      console.log("I will unmount now, bye!");
    };
  }, []);

  return (
    <div>
      <h1>Hello world!</h1>
      <p>How is everyone doing today?</p>
    </div>
  );
};
```

My preference

Functional components
Introduced in version 0.14 (October 2015)
Extended with hooks in version 16.8.0 (February 2019)

So... how can we use it?

create-react-app ?

Pros

- **Easy** to use
- Almost **no setting up** required!
- Run the command and start developing

Cons

- **No control** over your configurations
- **Big** and **bloated**
- You're going to **'eject'** sooner or later anyway

Interested anyway?
Check out **create-react-app.dev**

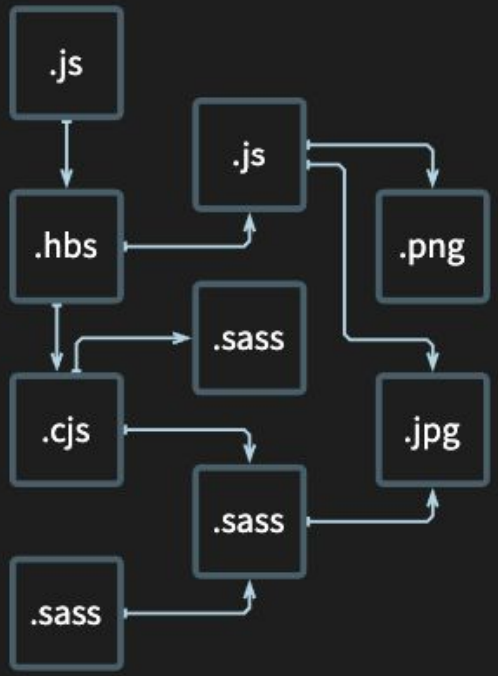
How it’s done

Translate & Compile

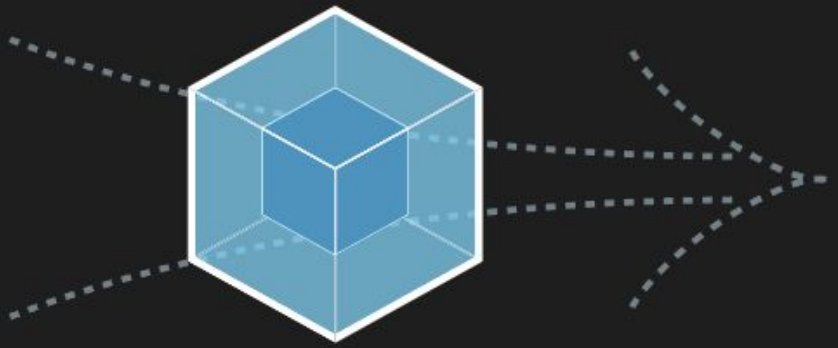
Bundle with dependencies

Present browser with bundles

Bundling with **Webpack**



MODULES WITH DEPENDENCIES



STATIC ASSETS

Not just JavaScript

Bundling with **Webpack**

- Created in **2012**
- **Bundles** and **transforms** your source code into static web assets
- Runs on **Node.JS**
- Fully configurable to process any type of file using **loaders**
- Processes only used files using a **dependency graph**
- Has a development server

Documentation available at
webpack.js.org

Transpile JavaScript with **Babel**

- Create **readable JavaScript** for **every browser** (of choosing)
- Use **new** and **experimental** ECMA features in your code
- Transform **JSX** into normal JavaScript
- Integrated with **Webpack** using the **babel-loader**

Documentation available at
babeljs.io

Simple **configuration**

webpack.config.js

Your entry file

Your output configuration

Your rules on how you want your files to be processed

The rule's regular expression

Which loaders to use for this rule (there can be more than one!)

The loader's options

```
module.exports = {
  entry: "./src/entry.jsx",

  output: {
    path: "build",
    filename: "bundle.js"
  },

  module: {
    rules: [
      {
        test: /\.jsx?$/,

        use: [
          {
            loader: "babel-loader",
            options: {
              // Options for babel
            }
          }
        ]
      }
    ]
  }
}
```

Not just **JavaScript**

- Add **CSS** with **style-loader** and **css-loader**
- **Inline** small files with **url-loader** (Check out the `data:` protocol)
- Add bigger files with **file-loader**
- Automatically fill **HTML** files with **HTML Webpack Plugin**

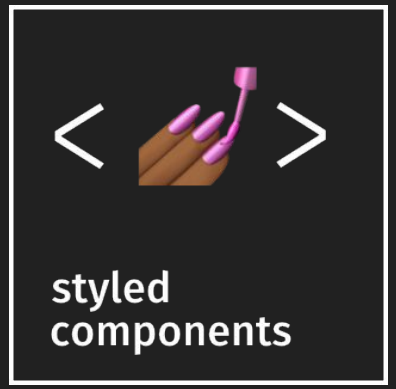
```
rules: [
  {
    test: /\.jsx?$/,
    use: ["babel-loader"]
  },
  {
    test: /\.svg?$/,
    use: [
      "@svgr/webpack",
      "url-loader"
    ]
  },
  {
    test: /\.css$/,
    use: [
      "style-loader",
      "css-loader"
    ]
  },
  {
    test: /\.scss$/,
    use: [
      "style-loader",
      "css-loader",
      "resolve-url-loader",
      "sass-loader"
    ]
  },
  {
    test: [/\.jpe?g$/, /\.png$/, /\.[ot]tf$/],
    use: [
      {
        loader: "url-loader",
        options: {
          limit: 10000
        }
      }
    ]
  }{
```

Styling with React

Simple CSS bundle

CSS MODULES

I will be using this one



Styling with React

Preprocessors

I will be using this one

And this one!

Example

MyComponent.jsx

```jsx
import React from "react";
import styles from "./MyComponent.scss";

export const MyComponent = () => {
  return (
    <div className={styles.container}>
      <h1 className={styles.title}>
        Hello world!
      </h1>
    </div>
  );
};
```

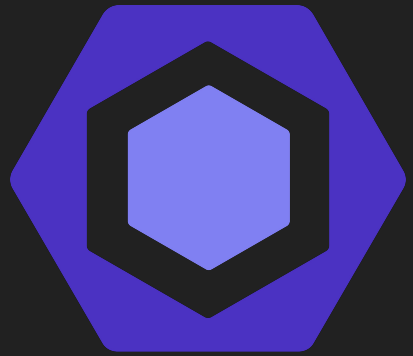
MyComponent.scss

```scss
$background: #ff0000;
$font: Montserrat, sans-serif;

.container {
  width: 500px;
  height: 200px;
  background-color: $background;
}

.title {
  font-family: $font;
  color: #fff;
}
```

Improving **Code Quality**

Linting with **ESLint**

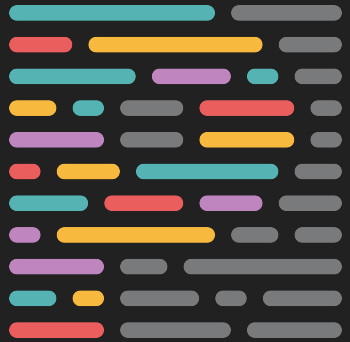
- Analyses your JavaScript code for **errors** and **flaws** (linting)
- **Integrated** in today's most popular **IDEs** (VSCode, WebStorm, etc.)
- Has a **plugin** for **Webpack**
- Can analyse **JSX** for style and accessibility errors



Documentation available at **eslint.org**

Code formatting with **Prettier**

- **Cleans up** your code
- Keeps your code in the **same style** everywhere
- Manages your **indenting**
- **Configurable** to your own liking



Documentation available at
prettier.io

Before we continue,

any questions?