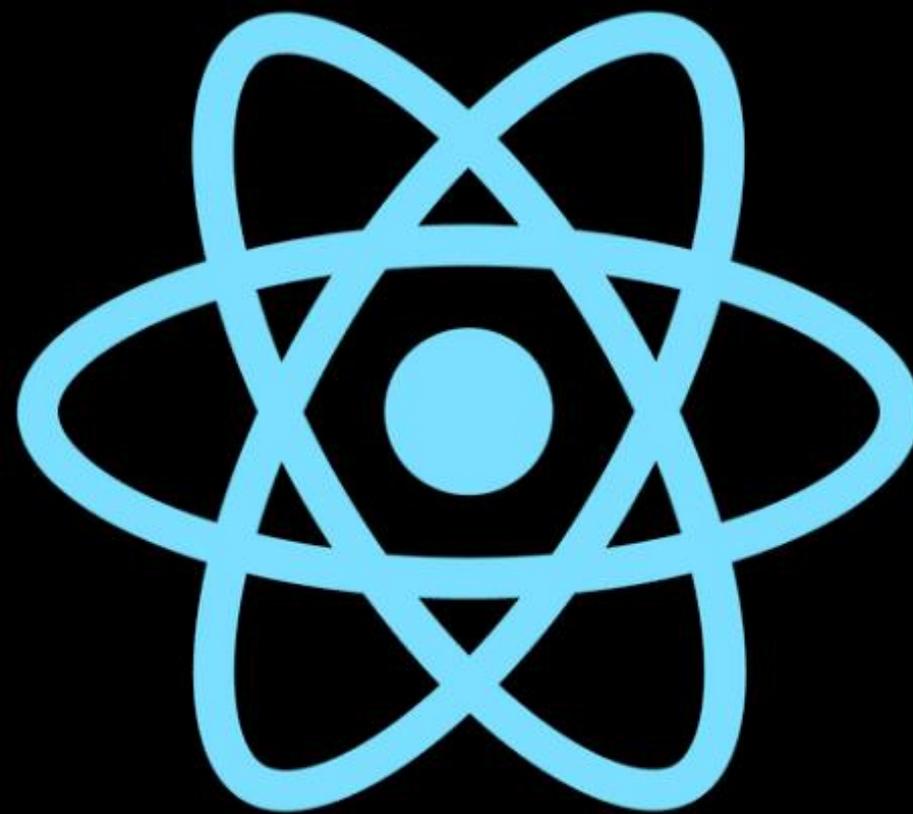


REACT

Laboratorium programowania: Frameworki
aplikacji internetowych



React JS

What is it and how to use it?

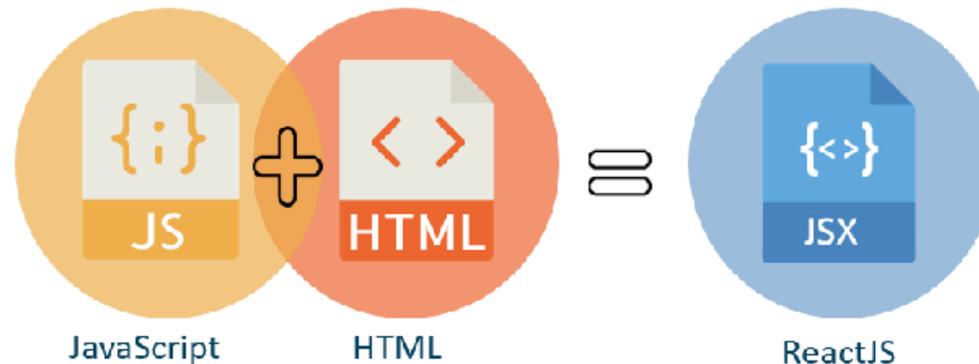
- React is a JavaScript library for building user interfaces
- It is used to build single-page applications
- Uses component based architecture – allows us to create reusable UI components
- To use it we should understand:
 - What a component is
 - What is a component state
 - What is a property for a component

Components

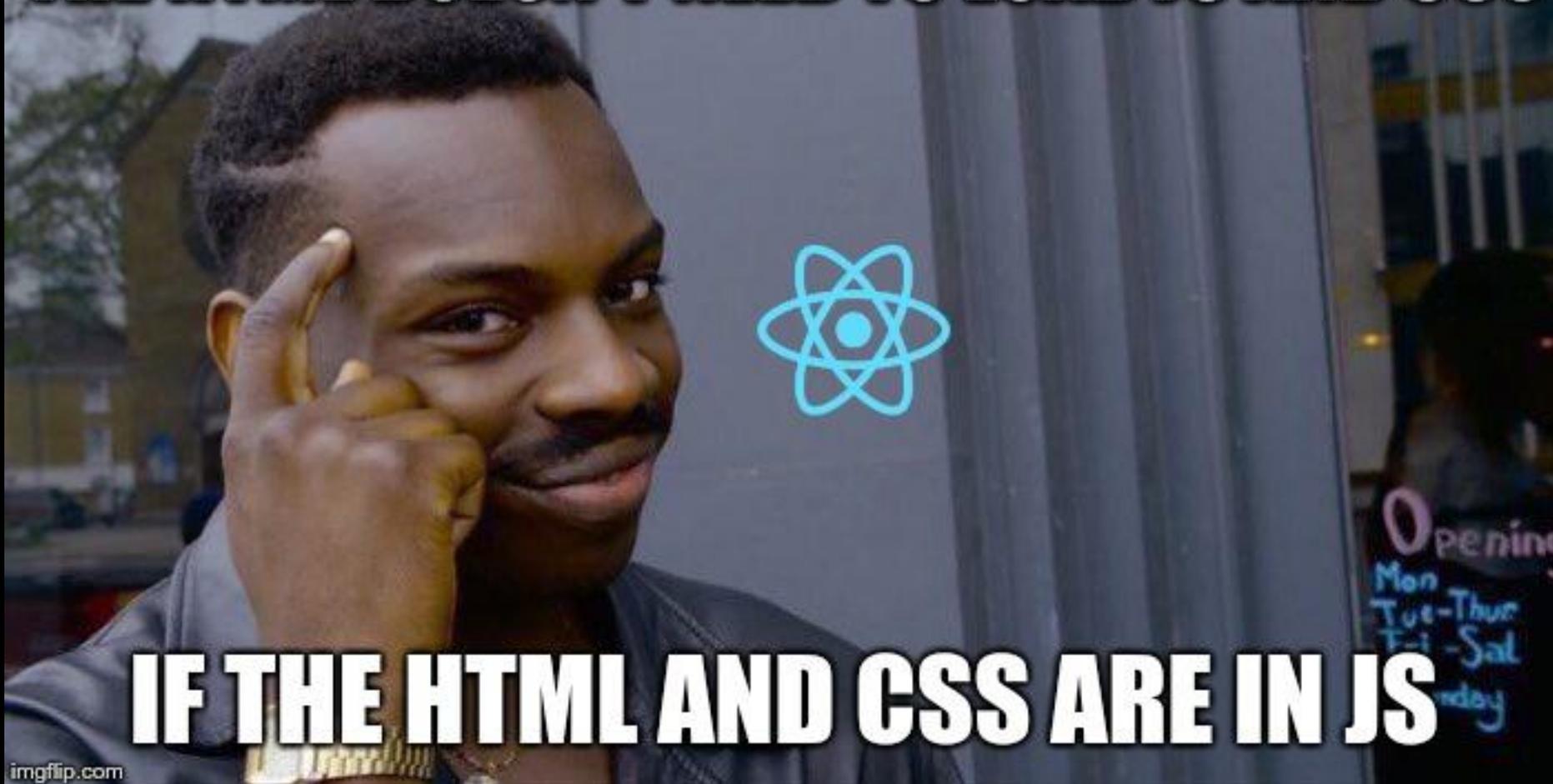
- Component is a class or function that returns an HTML , thus we have two types of components:
 - FUNCTIONAL Components – JS function that may or may not receive data as parameters
 - CLASS Components – JS class which can work with other components
- Component can be e.g. a button or a heading section
- It is especially useful if an element should be used repeatedly on a page

JSX

- React distinguishes JSX language.
- JSX is an extension of the JavaScript language and is translated into regular JavaScript at runtime.
- It converts HTML tags into react elements and makes it easier to write and add HTML in React.
- It allows us to write HTML directly within the JavaScript code!



THE HTML DOESN'T NEED TO LOAD JS AND CSS



IF THE HTML AND CSS ARE IN JS

imgflip.com

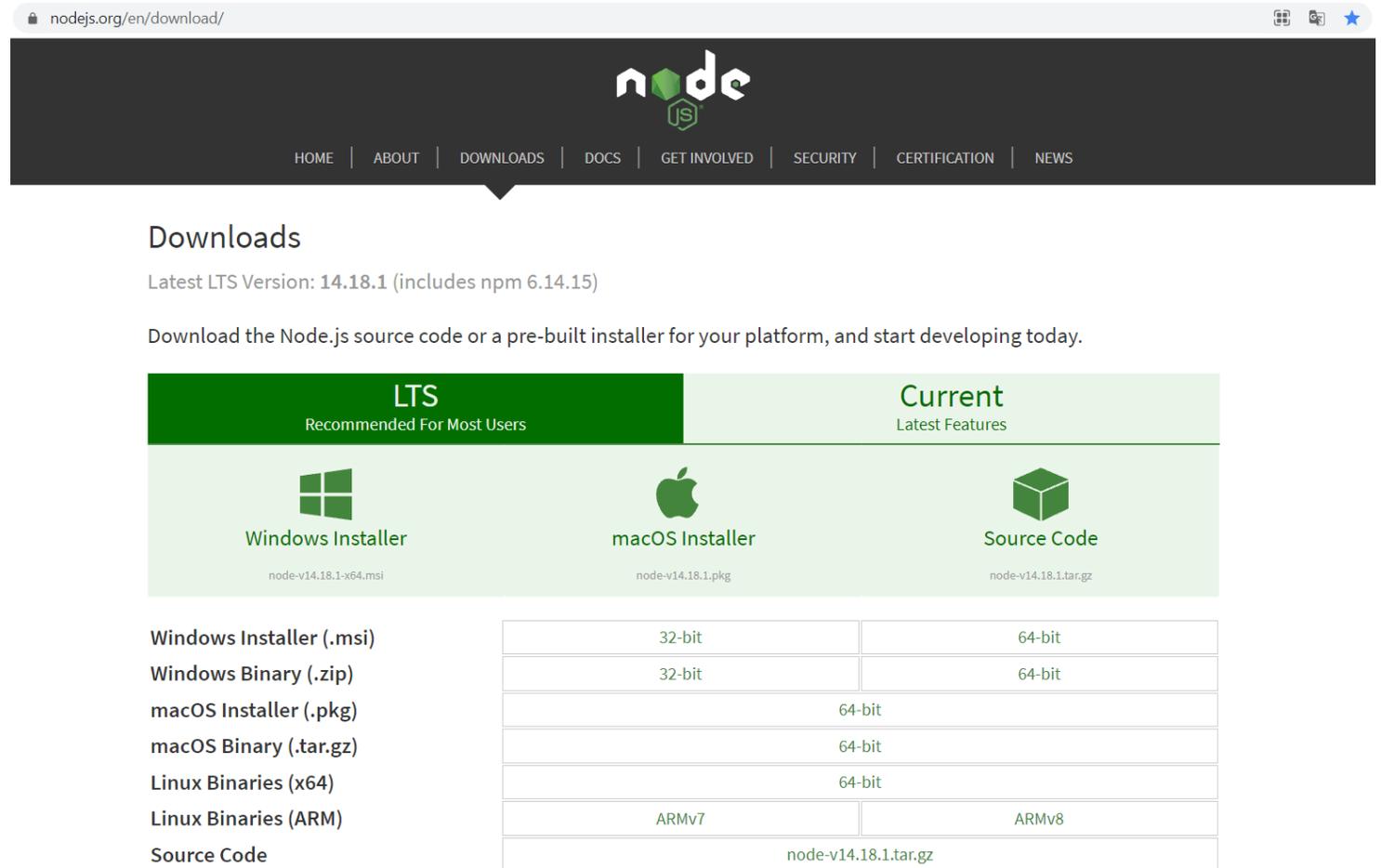
<https://imgflip.com/i/2kuh6f>

State and props

- State is one of the most important things in all components
- Whenever a state changes, a component is reloaded. React only changes what needs to be changed.
- There is a special field that holds the state of an element
- Thanks to state, we can know e.g. if a button is clicked

- Props are everything that is passed to component
- It is used when we want to e.g pass a value from one component to another

What should I install to use React?



The screenshot shows the Node.js download page. At the top, the Node.js logo is centered, with a navigation menu below it containing links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. The main heading is "Downloads", followed by the text "Latest LTS Version: 14.18.1 (includes npm 6.14.15)". Below this is a paragraph: "Download the Node.js source code or a pre-built installer for your platform, and start developing today." The page is divided into two main sections: "LTS Recommended For Most Users" (highlighted in dark green) and "Current Latest Features" (light green). Under the LTS section, there are three options: "Windows Installer" (node-v14.18.1-x64.msi), "macOS Installer" (node-v14.18.1.pkg), and "Source Code" (node-v14.18.1.tar.gz). Under the Current section, there is a table of download links for various architectures.

nodejs.org/en/download/

node

HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Downloads

Latest LTS Version: 14.18.1 (includes npm 6.14.15)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS
Recommended For Most Users

Current
Latest Features


Windows Installer
node-v14.18.1-x64.msi


macOS Installer
node-v14.18.1.pkg


Source Code
node-v14.18.1.tar.gz

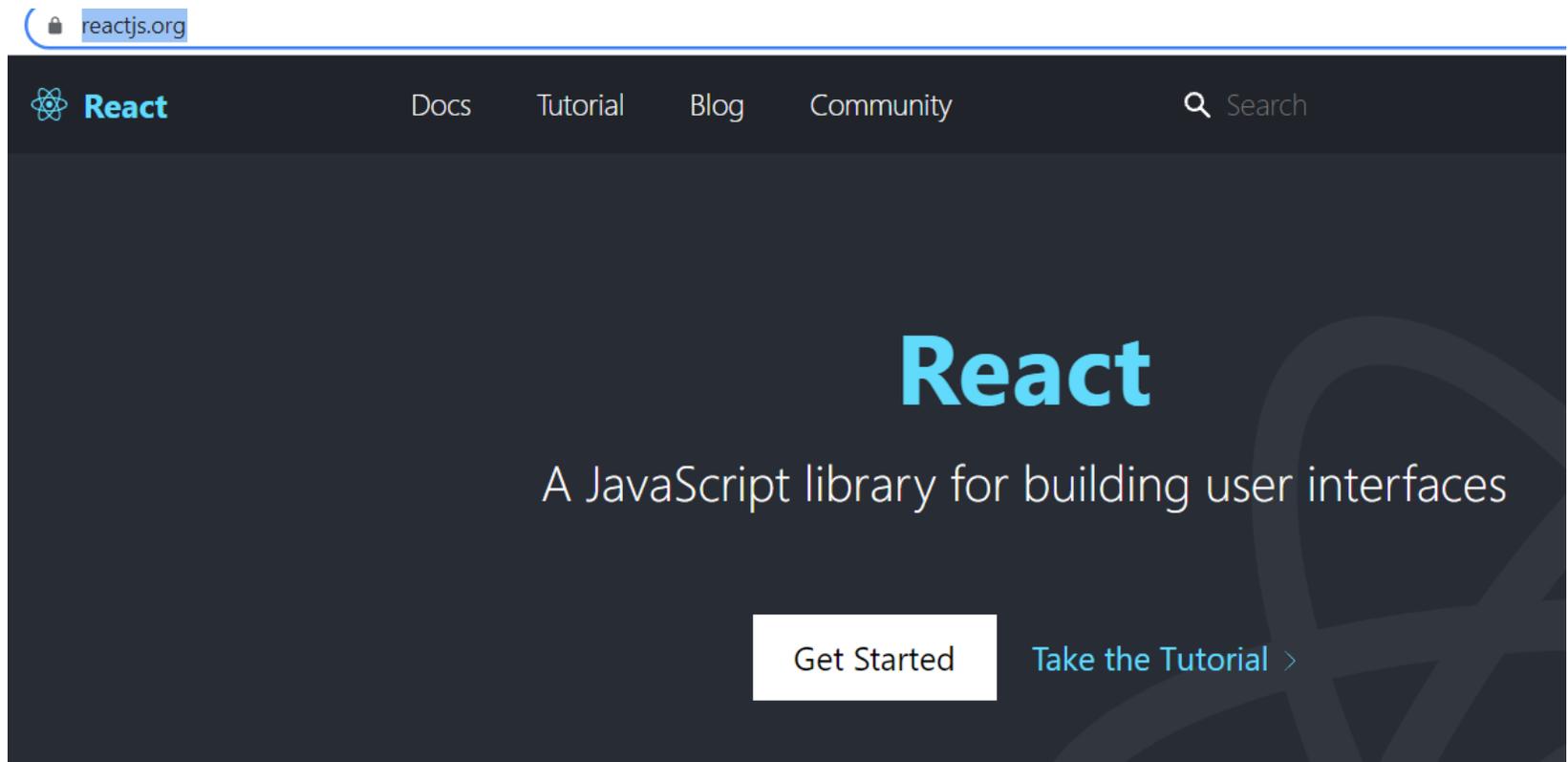
32-bit	64-bit
32-bit	64-bit
64-bit	
64-bit	
64-bit	
ARMv7	ARMv8
node-v14.18.1.tar.gz	

Windows Installer (.msi)
Windows Binary (.zip)
macOS Installer (.pkg)
macOS Binary (.tar.gz)
Linux Binaries (x64)
Linux Binaries (ARM)
Source Code

How to start using it?

The simplest way to get the general overview

- The simplest way is to write React code directly in HTML
- Visit <https://reactjs.org/> and click „Get started”



There is a starter template

Try React

React has been designed from the start for gradual adoption, and **you can use as little or as much React as you need**. Whether you want to get a taste of React, add some interactivity to a simple HTML page, or start a complex React-powered app, the links in this section will help you get started.

Online Playgrounds

If you're interested in playing around with React, you can use an online code playground. Try a Hello World template on [CodePen](#), [CodeSandbox](#), or [Stackblitz](#).

If you prefer to use your own text editor, you can also [download this HTML file](#), edit it, and open it from the local filesystem in your browser. It does a slow runtime code transformation, so we'd only recommend using this for simple demos.

Add React to a Website

You can [add React to an HTML page in one minute](#). You can then either gradually expand its presence, or keep it contained to a few dynamic widgets.

Copy everything to the plain HTML file, but focus on the 3 scripts below (it's the React core):

```
← → ↻ raw.githubusercontent.com/reactjs/reactjs.org/main/static/html/single-file-example.html

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Hello World</title>
    <script src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <!-- Don't use this in production: -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>
    <div id="root"></div>
    <script type="text/babel">

      ReactDOM.render(
        <h1>Hello, world!</h1>,
        document.getElementById('root')
      );

    </script>
    <!--
      Note: this page is a great way to try React but it's not suitable for production.
      It slowly compiles JSX with Babel in the browser and uses a large development build of React.

      Read this section for a production-ready setup with JSX:
      https://reactjs.org/docs/add-react-to-a-website.html#add-jsx-to-a-project

      In a larger project, you can use an integrated toolchain that includes JSX instead:
      https://reactjs.org/docs/create-a-new-react-app.html

      You can also use React without JSX, in which case you can remove Babel:
      https://reactjs.org/docs/react-without-jsx.html
    -->
  </body>
</html>
```

What are those scripts for?

- The first script is a React core
- The second (ReactDOM) is a library responsible for rendering application
- The third (Babel) is needed to use JSX

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Hello World</title>
    <script src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <!-- Don't use this in production: -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>
    <div id="root"></div>
    <script type="text/babel">

      ReactDOM.render(
        <h1>Hello, world!</h1>,
        document.getElementById('root')
      );
    </script>
    <!--
      Note: this page is a great way to try React but it's not suitable for production.
      It slowly compiles JSX with Babel in the browser and uses a large development build of React.

      Read this section for a production-ready setup with JSX:
      https://reactjs.org/docs/add-react-to-a-website.html#add-jsx-to-a-project
    -->
  </body>
</html>
```

Note also that:

- There is a div tag with id= "root" – it is a place where we put our React element
- ReactDOM.render – with the HTML code (h1 that should be placed into div-root) and document.getElementById('root') to locate the right place
- h1 is in fact a JSX (thus Babel is needed)
- type = "text/babel " is needed

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Hello World</title>
    <script src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <!-- Don't use this in production: -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>
    <div id="root"></div>
    <script type="text/babel">

      ReactDOM.render(
        <h1>Hello, world!</h1>,
        document.getElementById('root')
      );
    </script>
    <!--
    Note: this page is a great way to try React but it's not suitable for production.
    It slowly compiles JSX with Babel in the browser and uses a large development build of React.

    Read this section for a production-ready setup with JSX:
    https://reactjs.org/docs/add-react-to-a-website.html#add-jsx-to-a-project
  -->
  </body>
</html>
```

Note also that:

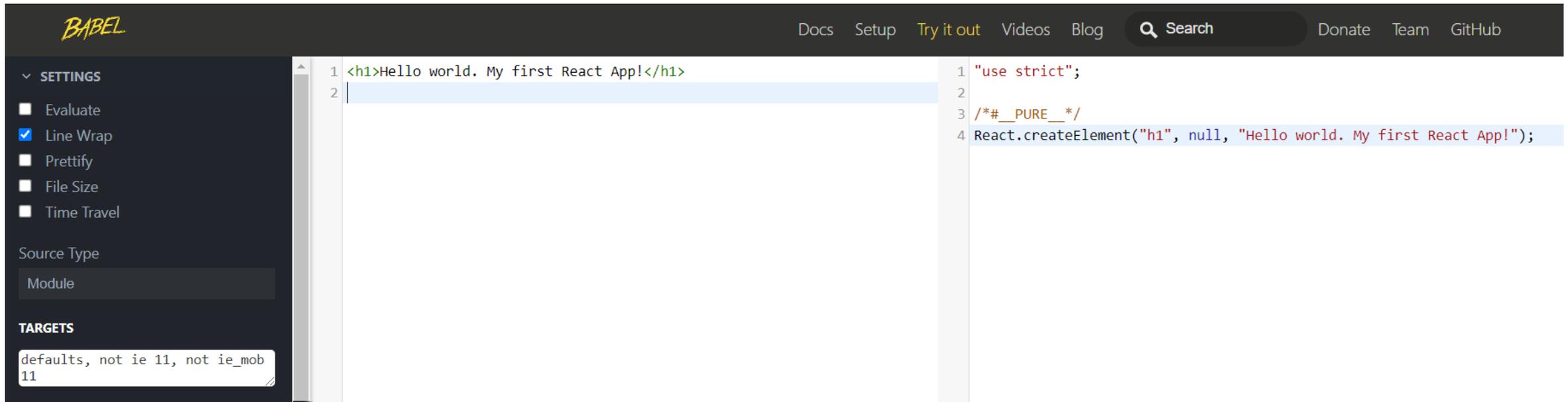
To get a general overview of what React is, we can write React code directly in HTML BUT to use React in production, we need npm which is included with Node.js.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Hello World</title>
    <script src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <!-- Don't use this in production: -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>
    <div id="root"></div>
    <script type="text/babel">

      ReactDOM.render(
        <h1>Hello, world!</h1>,
        document.getElementById('root')
      );
    </script>
  <!--
  Note: this page is a great way to try React but it's not suitable for production.
  It slowly compiles JSX with Babel in the browser and uses a large development build of React.
  Read this section for a production-ready setup with JSX:
  https://reactjs.org/docs/add-react-to-a-website.html#add-jsx-to-a-project
```

How HTML is created?

- Visit <https://babeljs.io/>



The screenshot shows the Babel REPL interface. On the left, there is a sidebar with settings: 'Evaluate' (unchecked), 'Line Wrap' (checked), 'Prettify' (unchecked), 'File Size' (unchecked), and 'Time Travel' (unchecked). Below the settings is a 'Source Type' dropdown menu set to 'Module'. Under 'TARGETS', there is a text input field containing 'defaults, not ie 11, not ie_mob 11'. The main editor area is split into two panes. The left pane shows the input code:

```
1 <h1>Hello world. My first React App!</h1>
2
```

 The right pane shows the output code:

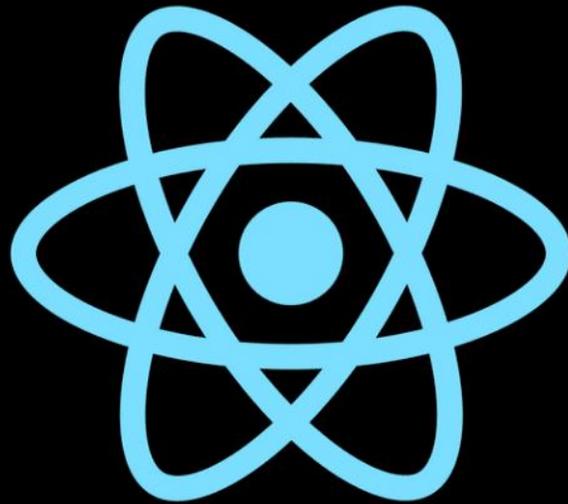
```
1 "use strict";
2
3 /*#__PURE__*/
4 React.createElement("h1", null, "Hello world. My first React App!");
```

- We use HTML but Babel translates it into React element

What to do
next after I
understand
the general
overview?

- Using JSX in scripts with type= " text/babel" annotation is not recommended. It slows down the page and is not advisable for larger projects
- It is not the easiest library, but there is a tool that allows us to create a new react application fast and smoothly: create-react-app (Node.js is required)

Prepare our first simple React component by changing the starting template 😊



React JS

EXAMPLE 1 – class component

- Let's create a component that places application header on the page:

```
<div id="hello"> </div>

<script type="text/babel">
  class Greeting extends React.Component {
    render(){
      return <h1>Hello world. This is my first React App!</h1>
      // return React.createElement('h1',null,'Hello world. This is m
y first React App!') - Babel
    }
  }
  ReactDOM.render(<Greeting />,document.getElementById('hello'))
</script>
```

Let's remember that it is still JS

We can change the code to the code below:

```
render(){  
  var headline = 'This is my first React App!'  
  return <h1>Hello world. {headline}</h1>  
}
```

Remember that return always returns only one thing -> return multiple items with one div

```
render(){
  var headline = 'This is my first React App!'
  return (<div>
    <h1>Hello world.</h1>
    <h2>{headline}</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut rutrum erat risus, in semper lorem interdum a. In eget pretium urna, in commodo orci. </p>
  </div>)
}
```

EXAMPLE 2 – functional component

- Let's create a component that displays a list of three main languages everyone should know to write the source code in React

```
const Languages = function () {  
  const arr_languages = ['HTML', 'JavaScript', 'CSS']  
  return (  
    <div>  
      <ul>  
        <li>{arr_languages[0]}</li>  
        <li>{arr_languages[1]}</li>  
        <li>{arr_languages[2]}</li>  
      </ul>  
    </div>  
  )  
}
```

Placing component in a different component

```
class Greeting extends React.Component {
  render(){
    var headline = 'This is my first React App!'
    return (<div>
      <h1>Hello world.</h1>
      <h2>{headline}</h2>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut
t rutrum erat risus, in semper lorem interdum a. In eget pretium urna,
in commodo orci. </p>
      <Languages />
    </div>)
  }
}
```

Summary

```
<div id="hello"></div>
<script type="text/babel">

  const Languages = function () {
    const arr_languages = ['HTML', 'JavaScript', 'CSS']
    return (
      <div>
        <h2>Everyone should know:</h2>
        <ul>
          <li>{arr_languages[0]}</li>
          <li>{arr_languages[1]}</li>
          <li>{arr_languages[2]}</li>
        </ul>
      </div>
    )
  }

  class Greeting extends React.Component {
    render() {
      var headline = 'This is my first React App!'
      return (<div>
        <h1>Hello world </h1>
        <h2> {headline} </h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed dignissim dictum porttitor.
        <Languages/>
      </div>)
    }
  }

  ReactDOM.render(<Greeting/>, document.getElementById('hello'))
</script>
```

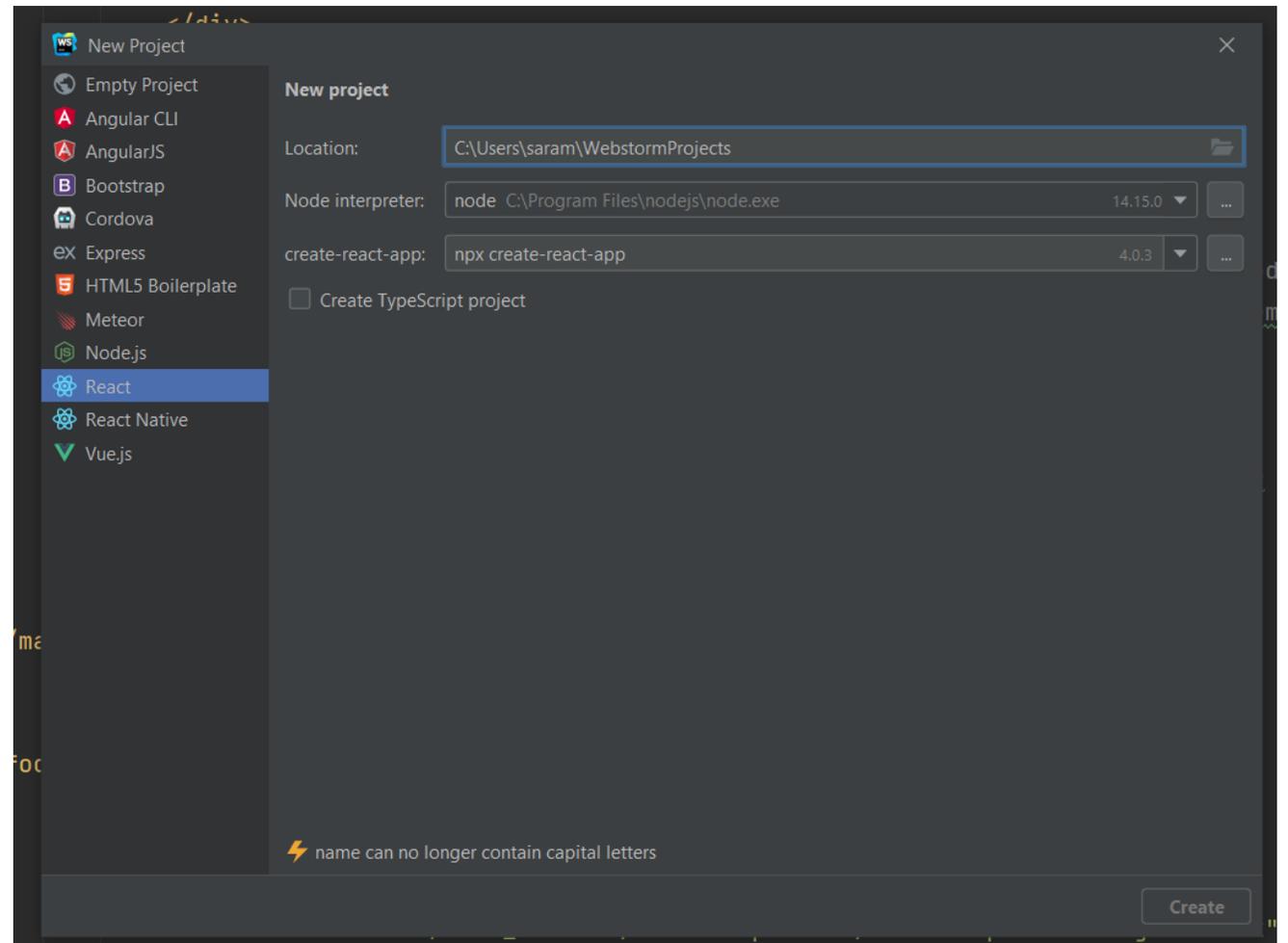
Exercise

- Create a component that displays your basic personal details

It works,
BUT!

- We shouldn't write components like this. This way is good only to get the idea of component.
- Applications should be written using create-react-app solution.

Create-react-app
on WebStorm –
there is a build-in
solution



Create-react-app on different environment, e.g. Visual Studio Code

```
npm install -g create-react-app  
(type it in terminal to use react and create-react-app)
```

```
npx create-react-app projectName  
(type in terminal to create a new react project)
```

```
npm run start (opens a project on  
http://localhost:3000/ )
```

-
- +
 - Starting from the next laboratory, we use create-react-app solution 😊