

FrontEnd

Class 1

July 14, 2021

What is React?

- <https://reactjs.org/>
- React is a JavaScript library for building user interfaces
- Example `www.netflix.com` in browser
- Highly interactive and no waiting to load, nice and smooth transitions
- Feels like a mobile app even in browser
- Websites traditionally feel like this.
- Lots of http requests/waiting for response. Felt clunky and we felt latency.
- JS allows us to run logic in browser, no need to request new html page.

What is React?

- React is a client-side JS library
- Build modern, reactive user interfaces for the web
- Why do we need react?
- React is the most popular Component based UI library
 - Does not come with certain features like routing. Those are third party packages.
- Alternatives
 - Angular – more built-in features. Component based as well. TypeScript.
 - Vue.js – mixture of Angular and React. Component based with more core features such as routing, but less overload than Angular.

IDE

- VSCode
- Preferences -> Color Theme
- Prettier extension and add under settings -> Formatter
- material icon theme

JavaScript Refresher – let versus const

- var is prior to es6
- let is the new var
- let for something that is variable
- const when u want to assign some value and never change
- jsbin.com

```
const name = 'Chetna';  
console.log(name);  
name = 'Arianna';
```

JavaScript Refresher – Arrow Function

```
function myFunc() {
```

```
...
```

```
}
```

```
const myFunc = () => {
```

```
...
```

```
}
```

JavaScript Refresher – Arrow Function

```
function printMyName(name) {  
  console.log(name);  
}
```

```
printMyName('Chetna');
```

JavaScript Refresher – Arrow Function

```
const printMyName =(name) => {  
  console.log(name);  
}
```

```
printMyName('Chetna');
```


JavaScript Refresher – Arrow Function

```
const printMyName = name => {  
  console.log(name);  
}  
printMyName('Chetna');
```

JavaScript Refresher – Arrow Function

```
const multiply = (number) => {  
  return number * 2;  
}
```

```
const multiply = (number) => number * 2;
```

```
console.log(multiply(5));
```

JavaScript Refresher – export and import

Review files app.js, person.js and utility.js

Default export – can pick name in import

Named export – have to use name provided in export for import or

Can use alias import {smt as Smth} from './utility'

JavaScript Refresher – Classes

Blueprints for JavaScript objects

Classes support inheritance

Person1.js

Classes are one way to create components.

JavaScript Refresher – Classes, Properties and Methods.

Modern syntax

- Properties are variables attached to classes/objects.
- Methods are like functions attached to classes/objects.

- ES7 syntax

Property = 'value';

- ES6 syntax

```
constructor() {  
  this.property = 'value';  
}
```

JavaScript Refresher – Classes, Properties and Methods.

- ES7 syntax

```
myMethod = () => { ... }
```

- ES6 syntax

```
myMethod() {  
  ...  
}
```

person2.js

JavaScript Refresher – Spread and Rest Operator.

...

Spread – Used to split up array elements or object properties

```
const newArray = [...oldArray, 1, 2]
```

```
const newObject = {...oldObject, newProp: 5}
```

Rest – Used to merge a list of function arguments into an array

```
function sortArgs(...args) {  
    return args.sort()  
}
```

spread.js

JavaScript Refresher – Destructuring.

Extract array elements or object properties and store them in variables

Array

```
[a,b] = ['Hello', 'Chetna']
```

```
console.log(a) //Hello
```

```
console.log(b) //Chetna
```

Object

```
{name} = {name:'Chetna', age: 28}
```

```
console.log(name) //Chetna
```

```
console.log(age) //undefined
```

destruct.js

JavaScript Refresher – Reference and Primitive Types.

reference.js

JavaScript Refresher – Array Functions.

array.js

JavaScript Refresher – Resources.

- Read more about let : <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let>
- Read more about const : <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const>
- Read more: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow functions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions)
- Arrays: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global Objects/Array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array)

React Basics

HTML, CSS and JS can be used to build user interfaces

Using React makes building complex, interactive and reactive user interfaces simpler

Don't have to focus on nitty gritty details and that is the advantage of using a library.

Components – Reusable building blocks in your ui. Html/CSS/JS.

Why Components? Reusability/ DRY/ Separation of Concerns/
Syntax and JSX

React Basics - Components

Declarative Approach – means we won't tell React that a certain html elements be created and inserted in a specific place.

Instead, we define the desired target state(s) and let react figure out the actual JS DOM instructions.

Creating a React app - <https://github.com/facebook/create-react-app>

Node.js allows u to run JS code outside of browser. Since React code runs in browser, we don't need node directly, but we need it to execute command for create-react-app. www.nodejs.org

npx is an npm package runner.

npx create-react-app my-app

React Basics – project files

src folder has index.js

Some transformations occur while running npm start

Some not regular JS syntax

```
import './index.css';
```

Simply tells react to include the css

```
ReactDOM.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>,  
  document.getElementById('root')  
);
```

React Basics – package.json file

react and react-dom are dependencies that form react library.

render method takes two arguments.

Public folder has the html file that is used by spa.

It has the div with id of root.

Import App from './App' u can omit .js

App is a component. That is rendered inside the div.

It has a function that is exported.

React Basics – JSX

JavaScript XML

Html code in JS

Transformed code in bundle.js etc is react packaged code.”

JSX code (easier to write) is transformed to code working in browser.

React Basics – How React works

Our own custom HTML instead of working with document and modifying it.
Imperative way (cumbersome) is

```
Const para = document.getElement('p');  
Para.textContent = 'text'  
Document.getElementById('root').append(para);
```

Versus declarative way (end state). React generates the instructions.

```
<div className="App">  
<h2>Let's get started</h2>  
<p>This is also visible</p>  
</div>
```

React Basics – Creating a custom component.

Build a component tree, meaning how things are organized

Component consists of a function returning some html.

Export the function as export default functionName

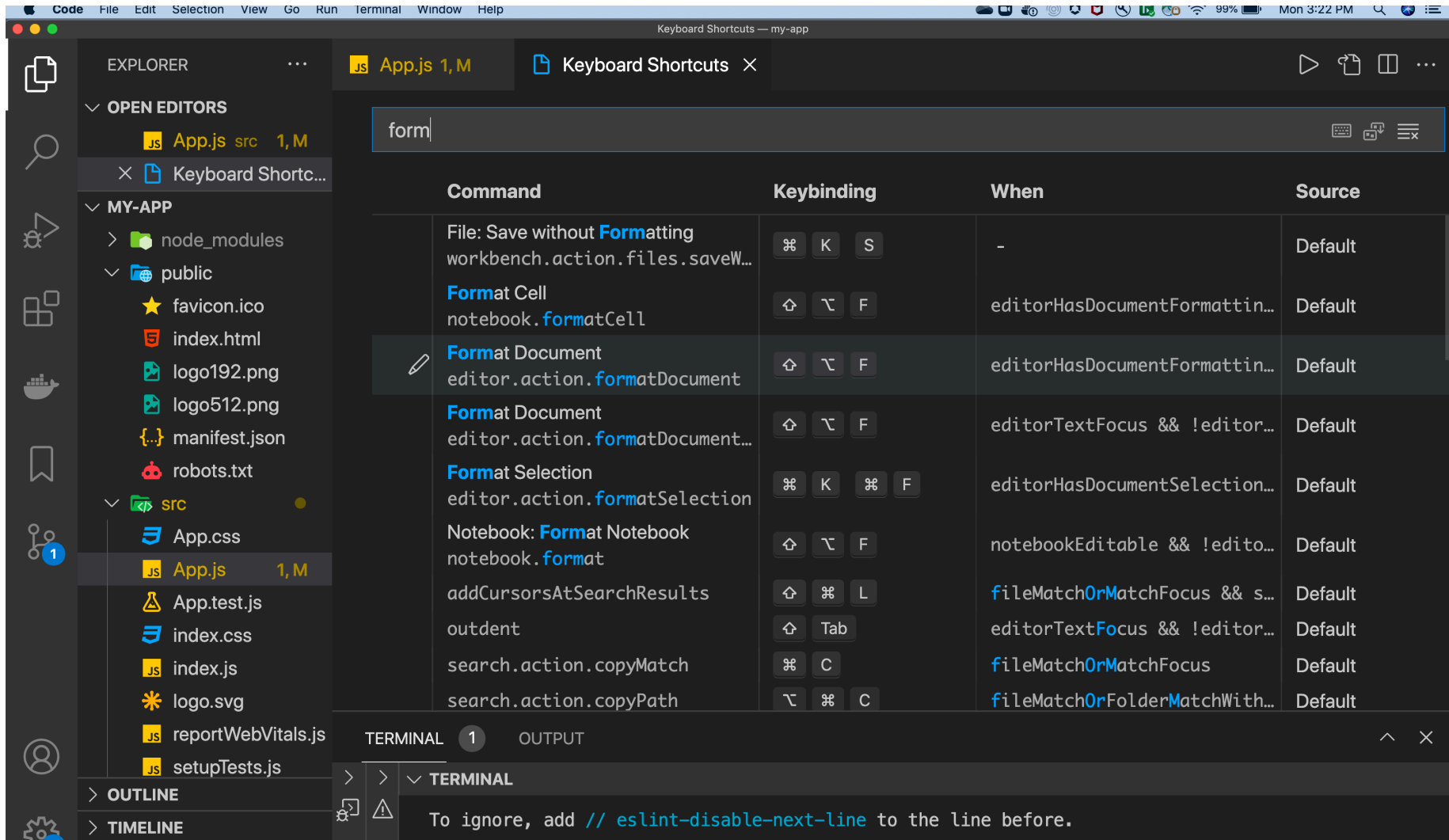
Import the custom component in the place where it needs to be inserted, like App.js

`<ExpenseItem></ExpenseItem>` Should start with uppercase. And only one root element. Can wrap into div tag. Wrapping in brackets also improves readability.

Also import css and define className instead of class (class is reserved name in JS). Not regular html but special JSX syntax.

Single `{}` for dynamic data.

React Basics – Creating a custom component.



React Basics – props.

Parameters for Components. Similar to parameters for a function.

Passing data.

Object parameter that holds properties (props).

Key value pairs.

For example: `props.title`

Components reusable and configurable

React Intro

<https://github.com/kentcdodds/beginners-guide-to-react>

Branch : egghead

open file start and copy and execute command in terminal.

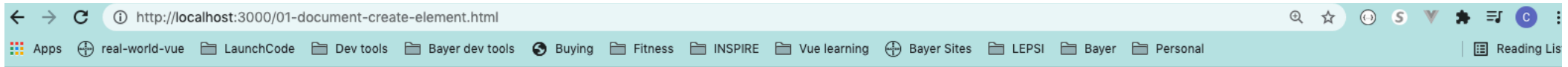
open in browser <http://localhost:3000/>

Browser synch tool.

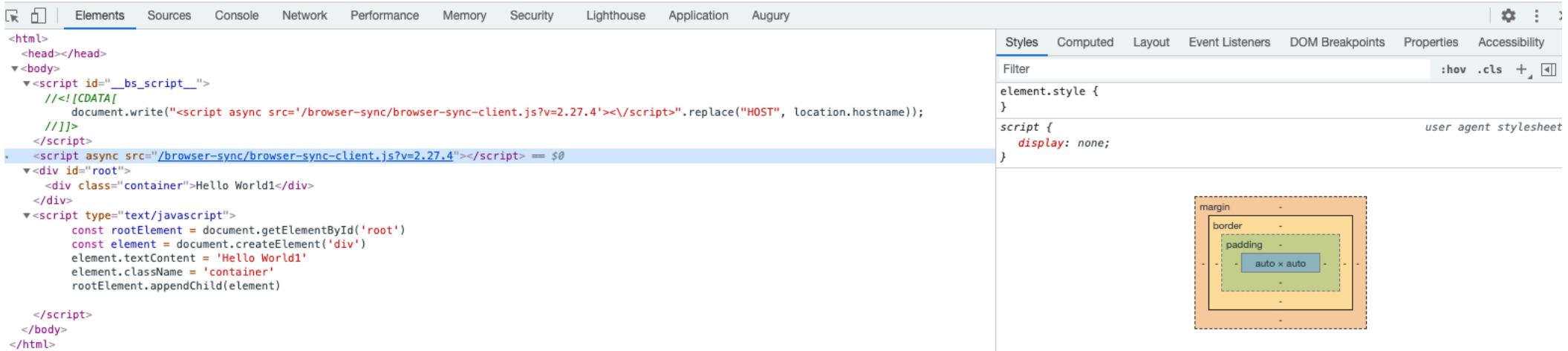
Open 01-document-create-element.html and make a change and see the change in browser!

Also have developer tools open.

React Intro



Hello World1



Create a User Interface with JS and DOM

Review file called [01-document-create-element.html](#)

Create a User Interface with React's createElement API

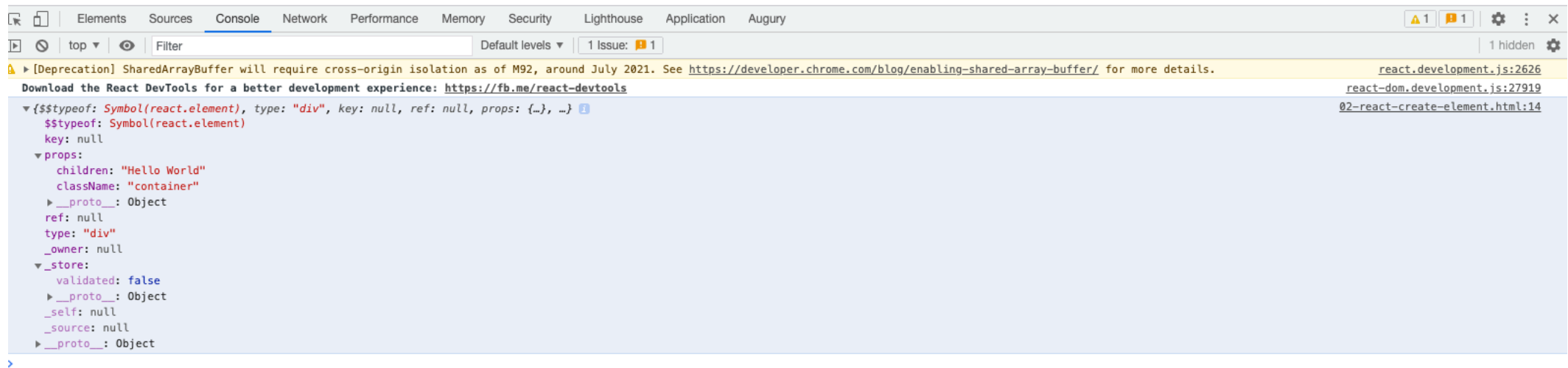
Review file called [02-react-create-element.html](#)

We can use unpkg to download minified versions of react and react dom.

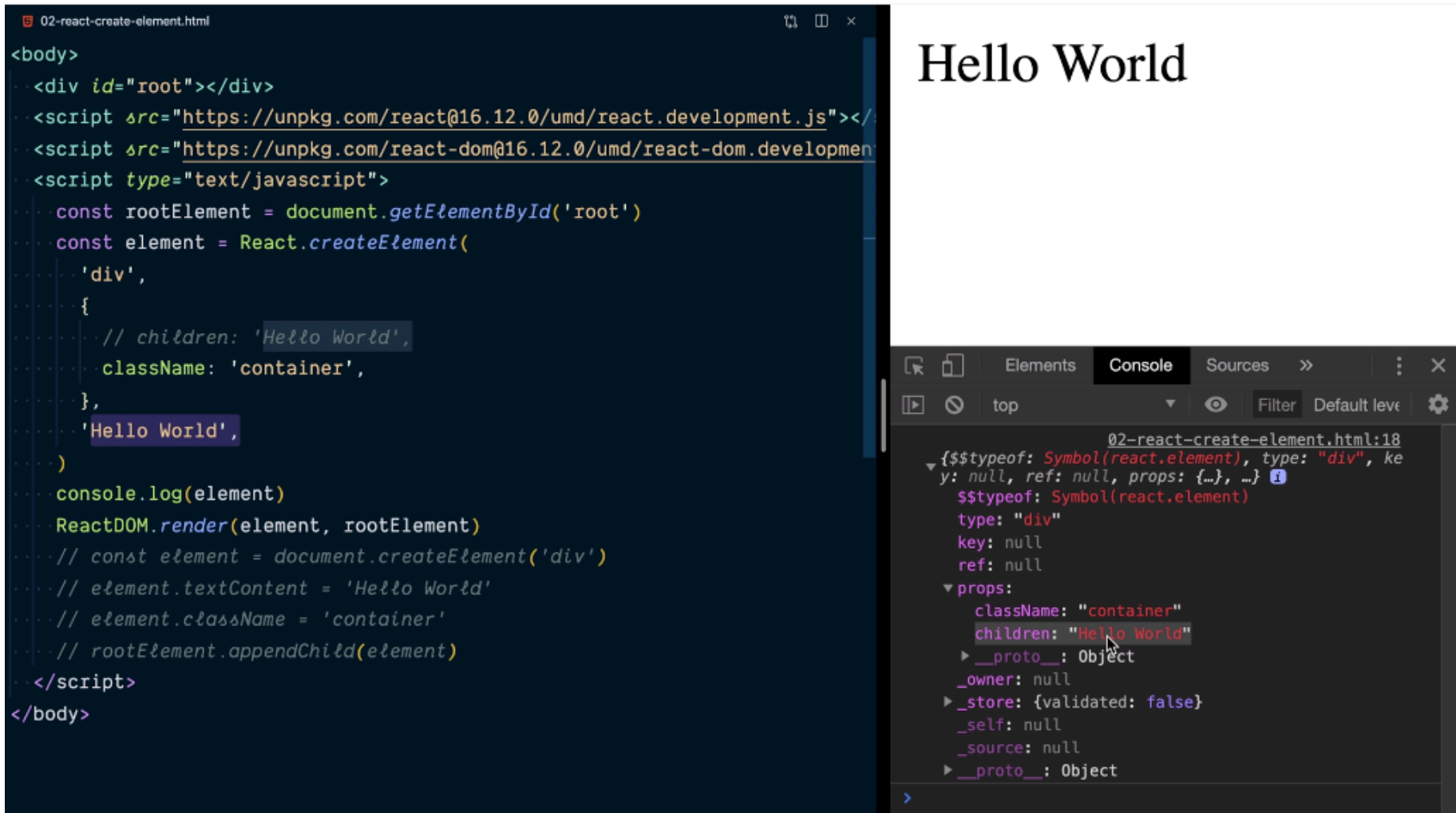
Inspect.

Create a User Interface with React's createElement API

Hello World



Create a User Interface with React's createElement API



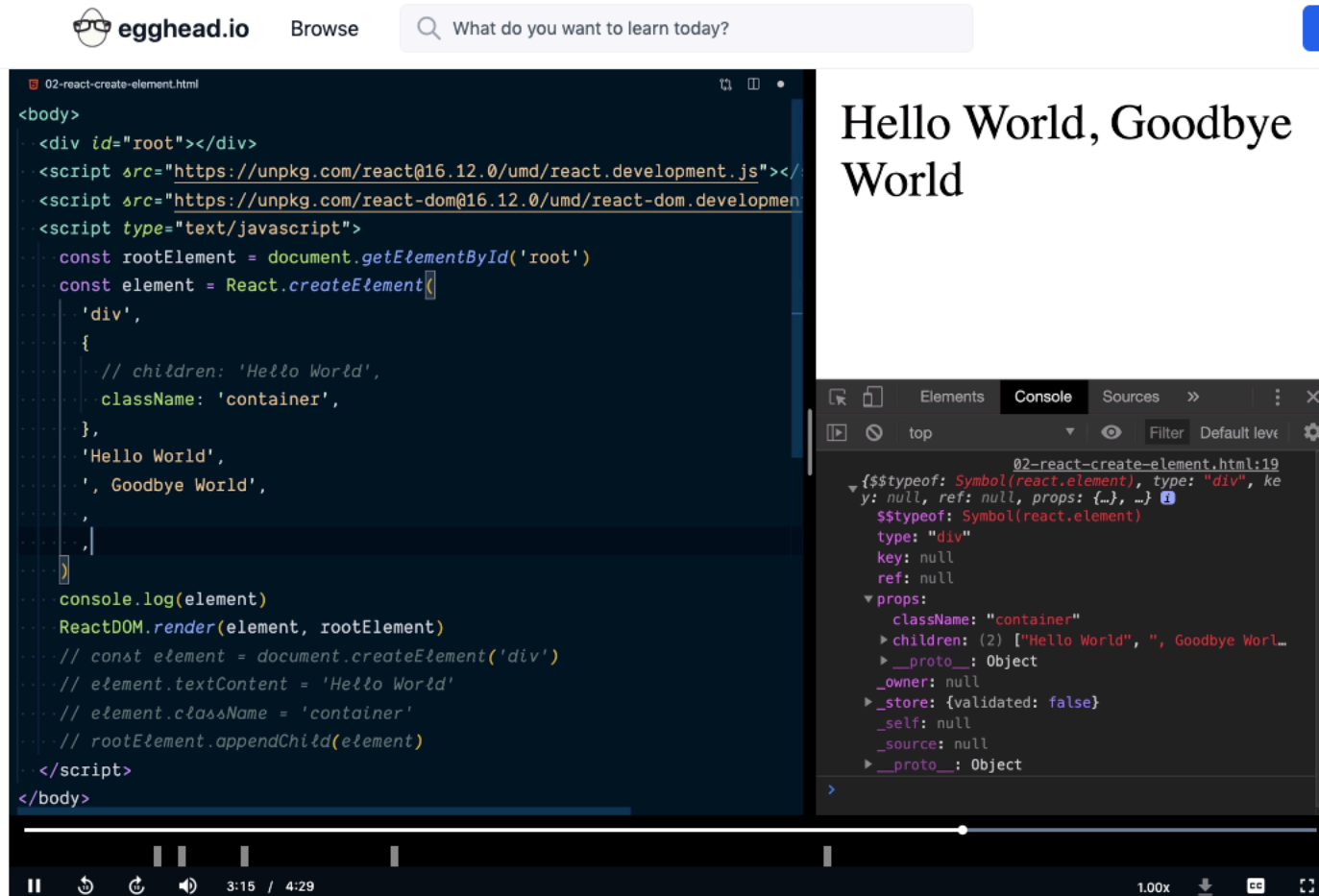
The image shows a web browser window with the title "02-react-create-element.html". The browser displays the text "Hello World" in a large, black, serif font. Below the browser window, the source code of the HTML file is visible. The code uses the React.createElement API to create a div element with the text "Hello World" and the class "container". The code is as follows:

```
<body>
  <div id="root"></div>
  <script src="https://unpkg.com/react@16.12.0/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.12.0/umd/react-dom.development.js"></script>
  <script type="text/javascript">
    const rootElement = document.getElementById('root')
    const element = React.createElement(
      'div',
      {
        // children: 'Hello World',
        className: 'container',
      },
      'Hello World',
    )
    console.log(element)
    ReactDOM.render(element, rootElement)
    // const element = document.createElement('div')
    // element.textContent = 'Hello World'
    // element.className = 'container'
    // rootElement.appendChild(element)
  </script>
</body>
```

The browser's developer tools are open, showing the Console tab. The console displays the React element object created by the code, which is a `div` element with the class `container` and the text `Hello World` as its child. The object structure is as follows:

```
{
  type: "div",
  key: null,
  ref: null,
  props: {
    className: "container",
    children: "Hello World",
  },
  __proto__: Object
}
```

Create a User Interface with React's createElement API



The screenshot shows a video player interface. At the top, there's a header for 'egghead.io' with a search bar containing 'What do you want to learn today?'. The video content is split into three main areas:

- Code Editor (Left):** Displays the following code:

```
<body>
  <div id="root"></div>
  <script src="https://unpkg.com/react@16.12.0/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.12.0/umd/react-dom.development.js"></script>
  <script type="text/javascript">
    const rootElement = document.getElementById('root')
    const element = React.createElement(
      'div',
      {
        // children: 'Hello World',
        className: 'container',
      },
      'Hello World',
      ', Goodbye World',
    )
    console.log(element)
    ReactDOM.render(element, rootElement)
    // const element = document.createElement('div')
    // element.textContent = 'Hello World'
    // element.className = 'container'
    // rootElement.appendChild(element)
  </script>
</body>
```
- Browser Window (Right):** Shows the rendered output: 'Hello World, Goodbye World'.
- Chrome DevTools Console (Bottom Right):** Shows the rendered element structure:

```
{
  type: "div",
  props: {
    className: "container",
    children: ["Hello World", ", Goodbye World"],
  },
}
```

The video player controls at the bottom show a progress bar at 3:15 / 4:29 and a 1.00x playback speed.

Create a User Interface with React's createElement API

```
02-react-create-element.html
<body>
  <div id="root"></div>
  <script src="https://unpkg.com/react@16.12.0/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.12.0/umd/react-dom.development.js"></script>
  <script type="text/javascript">
    const rootElement = document.getElementById('root')
    const element = React.createElement('div', {
      children: React.createElement('span', null, 'Hello', ' World'),
      className: 'container',
    })
    console.log(element)
    ReactDOM.render(element, rootElement)
    // const element = document.createElement('div')
    // element.textContent = 'Hello World'
    // element.className = 'container'
    // rootElement.appendChild(element)
  </script>
</body>
```

Hello World

```
Elements Console Sources >>
top
02-react-create-element.html:14
> |
```

Create a User Interface with React's JSX syntax

Review file called 03-jsx.html

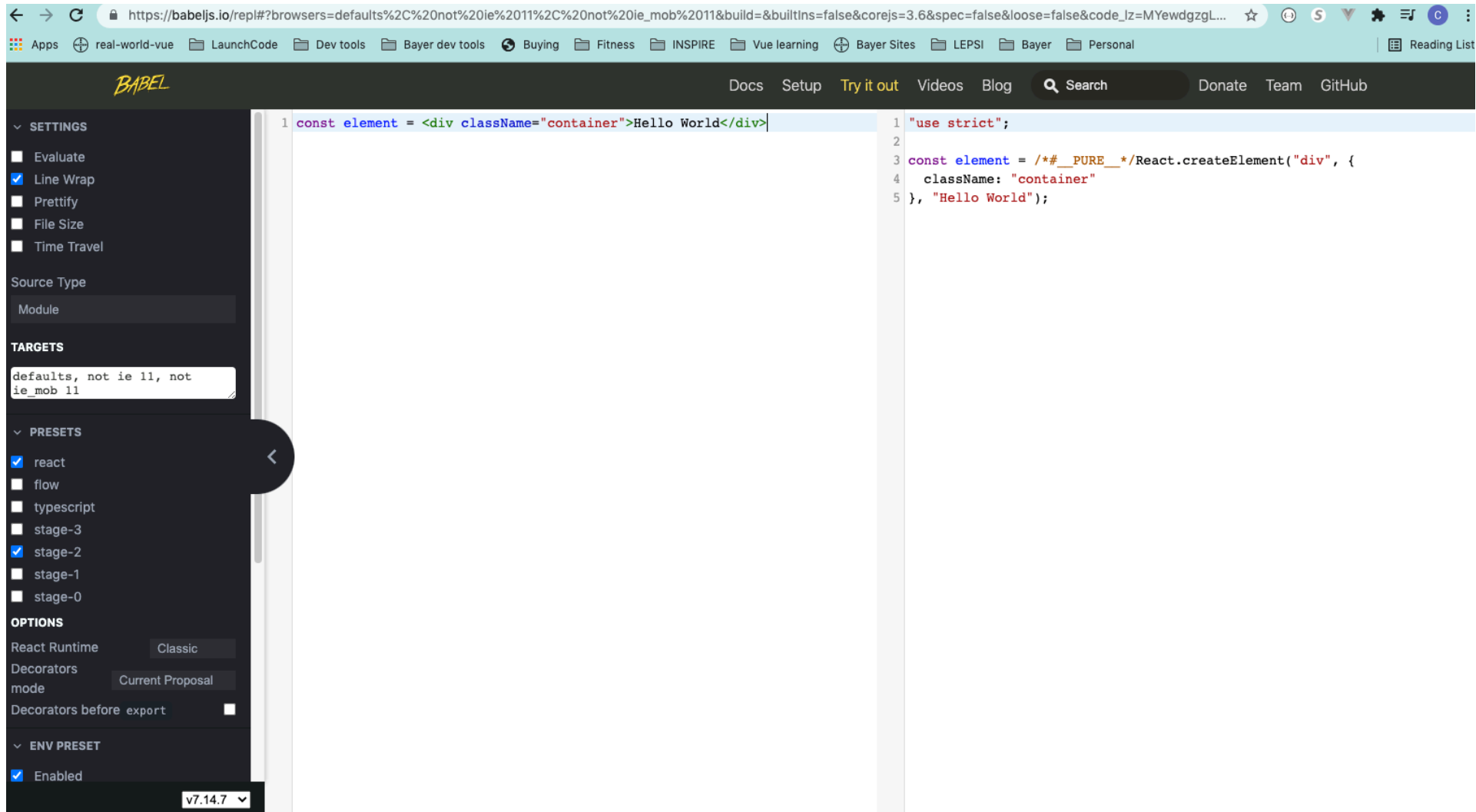
Html like syntax in JS

Not JS code and browser doesn't understand.

Babel compiler that supports next generation of JS and JSX.

<https://babeljs.io/repl>

Create a User Interface with React's JSX syntax



The screenshot displays the Babel REPL (https://babeljs.io/repl) interface. The left sidebar contains settings and presets. The main editor shows the transformation of JSX syntax into standard JavaScript code.

SETTINGS

- ☐ Evaluate
- ☒ Line Wrap
- ☐ Prettify
- ☐ File Size
- ☐ Time Travel

Source Type: Module

TARGETS

defaults, not ie 11, not ie_mob 11

PRESETS

- ☒ react
- ☐ flow
- ☐ typescript
- ☐ stage-3
- ☒ stage-2
- ☐ stage-1
- ☐ stage-0

OPTIONS

React Runtime: Classic

Decorators mode: Current Proposal

Decorators before export: ☐

ENV PRESET

- ☒ Enabled

v7.14.7

Code Editor:

Input (Left):

```
1 const element = <div className="container">Hello World</div>
```

Output (Right):

```
1 "use strict";
2
3 const element = /*#__PURE__*/React.createElement("div", {
4   className: "container"
5 }, "Hello World");
```

Using JSX effectively with React

Review file `04-jsx-tricks.html`

Variable value is interpolated, anything between `{}` is a JS expression.

Project

<https://learn.launchcode.org/courses/306/pages/project-rubric-all-gas-included>