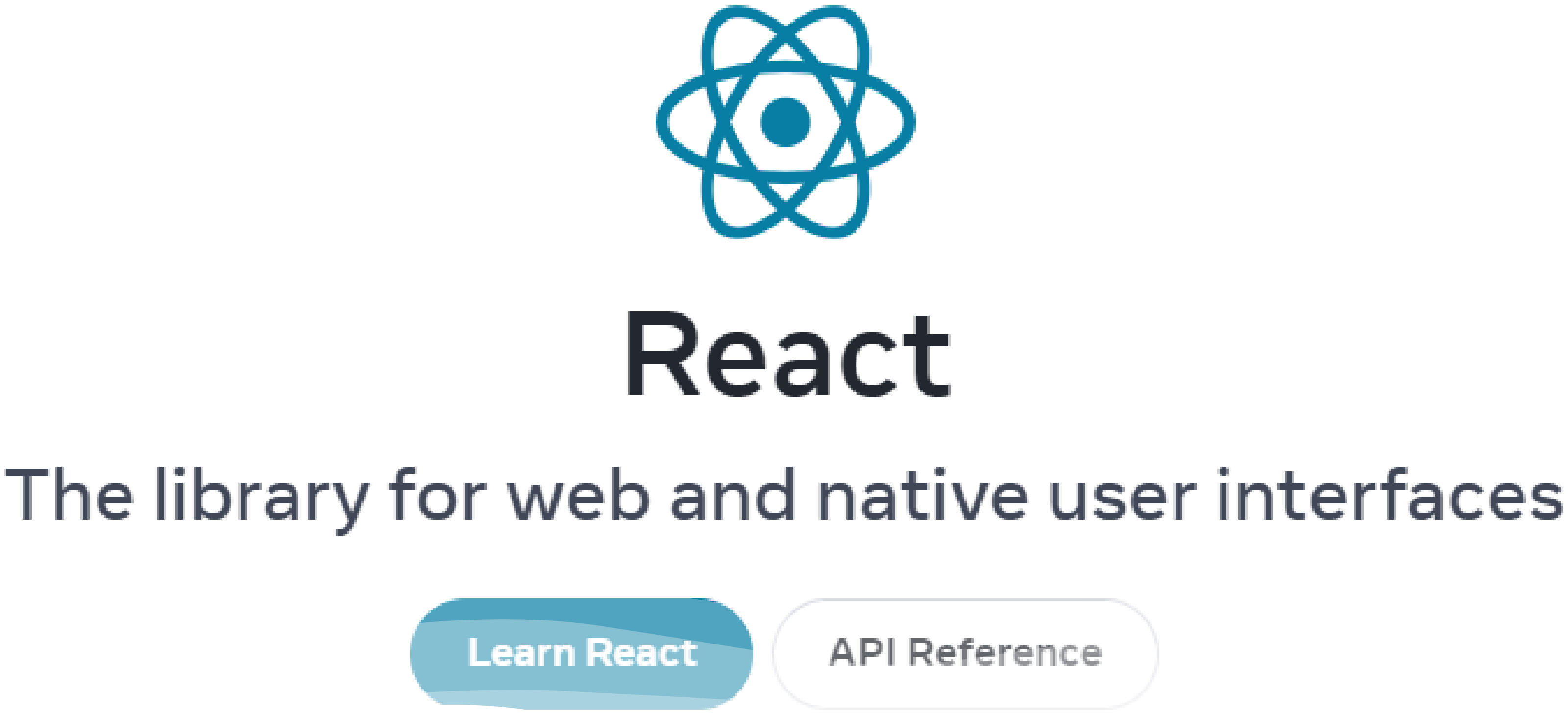


React

The library for web and native user interfaces

Introduction

Agenda

Background.

The V in MVC

TSX (Typescript Extension Syntax).

Developer tools..

React Component basics.

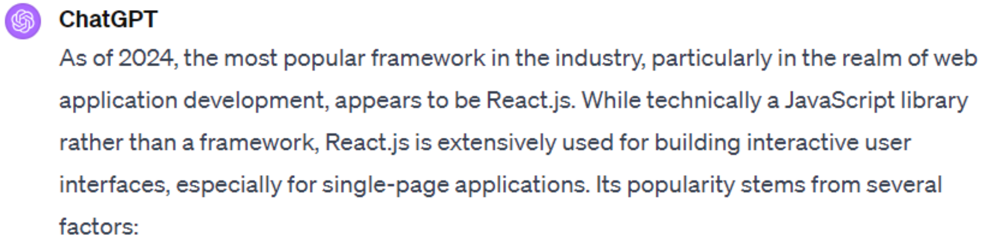
Material Design.

React

- **A Javascript framework for building dynamic Web** User Interfaces**.**
  - **A Single Page Apps technology.**
  - **Open-sourced in 2012.**

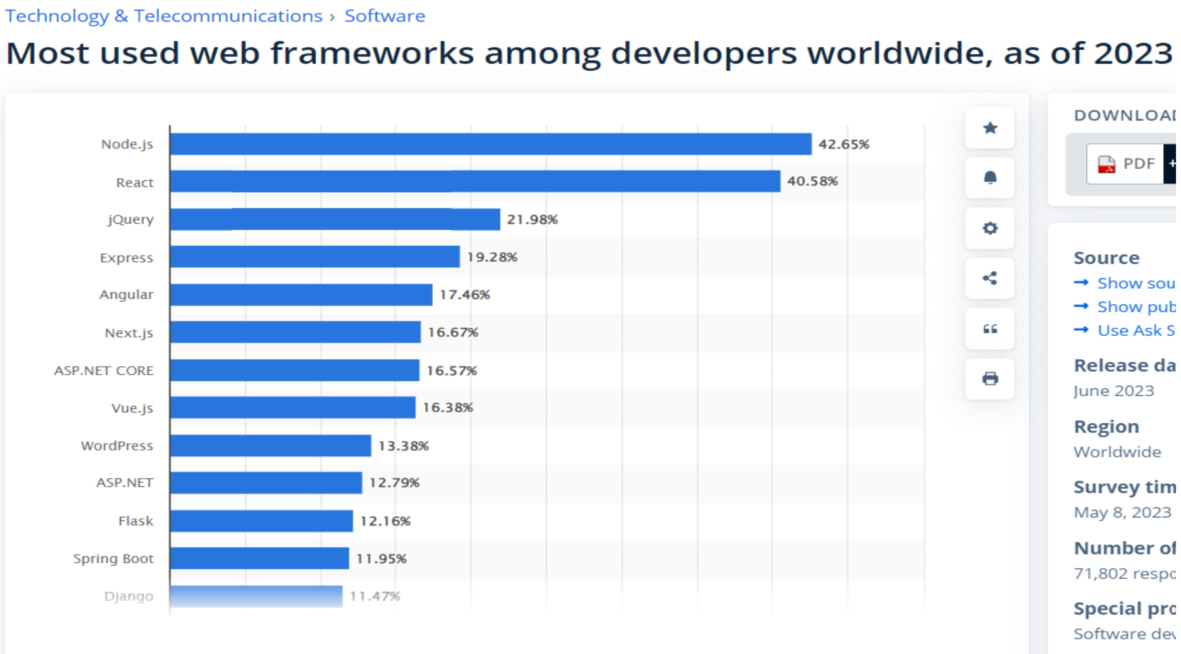
- **Client-side framework.**
  - **More a library than a framework.**

Why React?



ChatGPT

As of 2024, the most popular framework in the industry, particularly in the realm of web application development, appears to be React.js. While technically a JavaScript library rather than a framework, React.js is extensively used for building interactive user interfaces, especially for single-page applications. Its popularity stems from several factors:



Technology & Telecommunications › Software

Most used web frameworks among developers worldwide, as of 2023

Source
→ Show sou
→ Show pub
→ Use Ask S

Release da
June 2023

Region
Worldwide

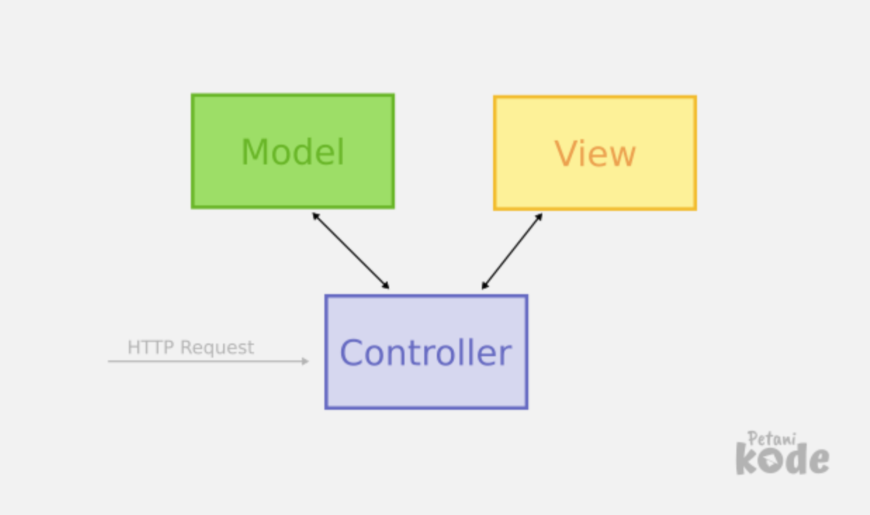
Survey tim
May 8, 2023

Number of
71,802 respo

Special pro
Software dev

Before React

- MVC pattern **– The convention for** app design. **Promoted** by **market leaders**, e.g. **AngularJS (1.x), EmberJS, BackboneJS.**
- **React is not MVC, just V.**
  - **It challenged established best practice (MVC).**
- Templating widespread use in the V layer.
  - **React based on “components”.**

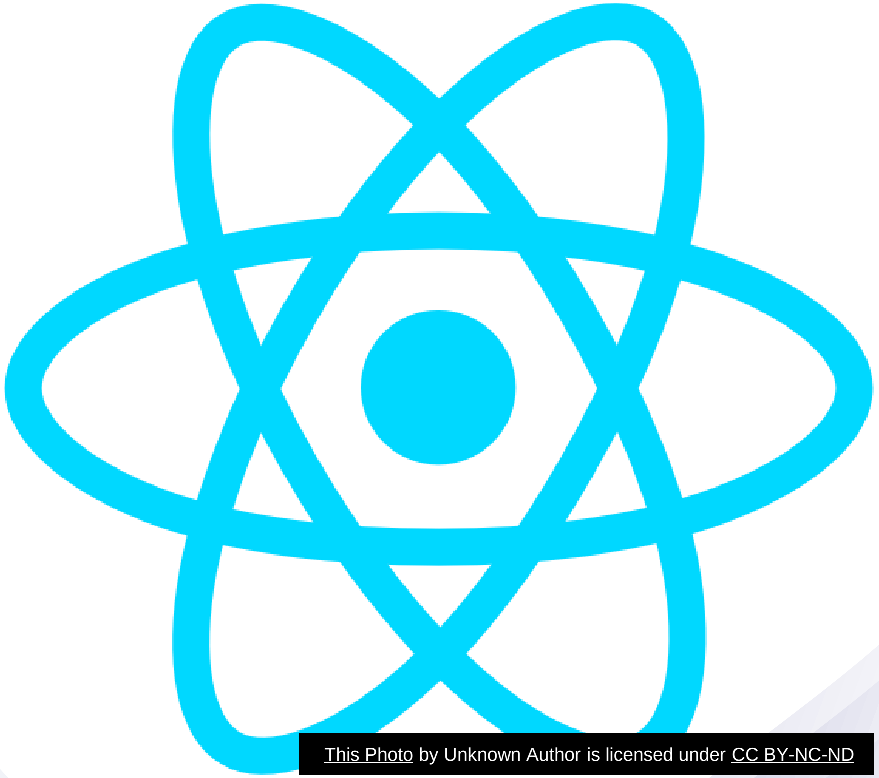


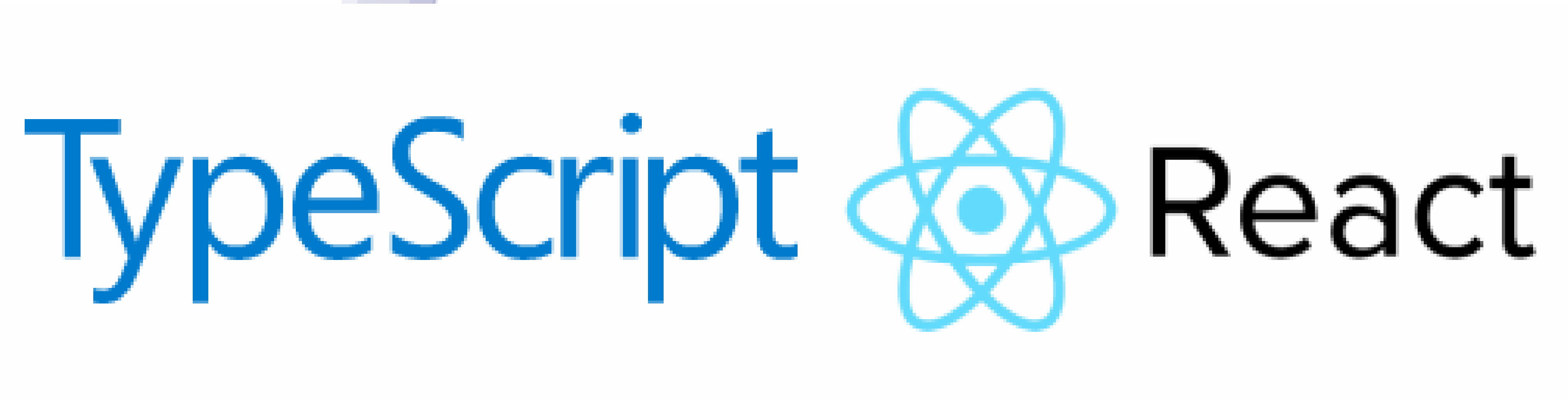
React Components

- **Philosophy:** *Build components, not templates.*
- **All about the User Interface (UI).**
  - Not focused on business logic or the data model (MVC)

- **Component - A unit comprised of:**

  UI description (HTML) + UI behaviour (JS)
  - **Two aspects are tightly coupled and co-located.**
    - Pre-React frameworks decoupled them.
  - **Benefits:**
    1. Improved Composition.
    2. Greater Reusability.

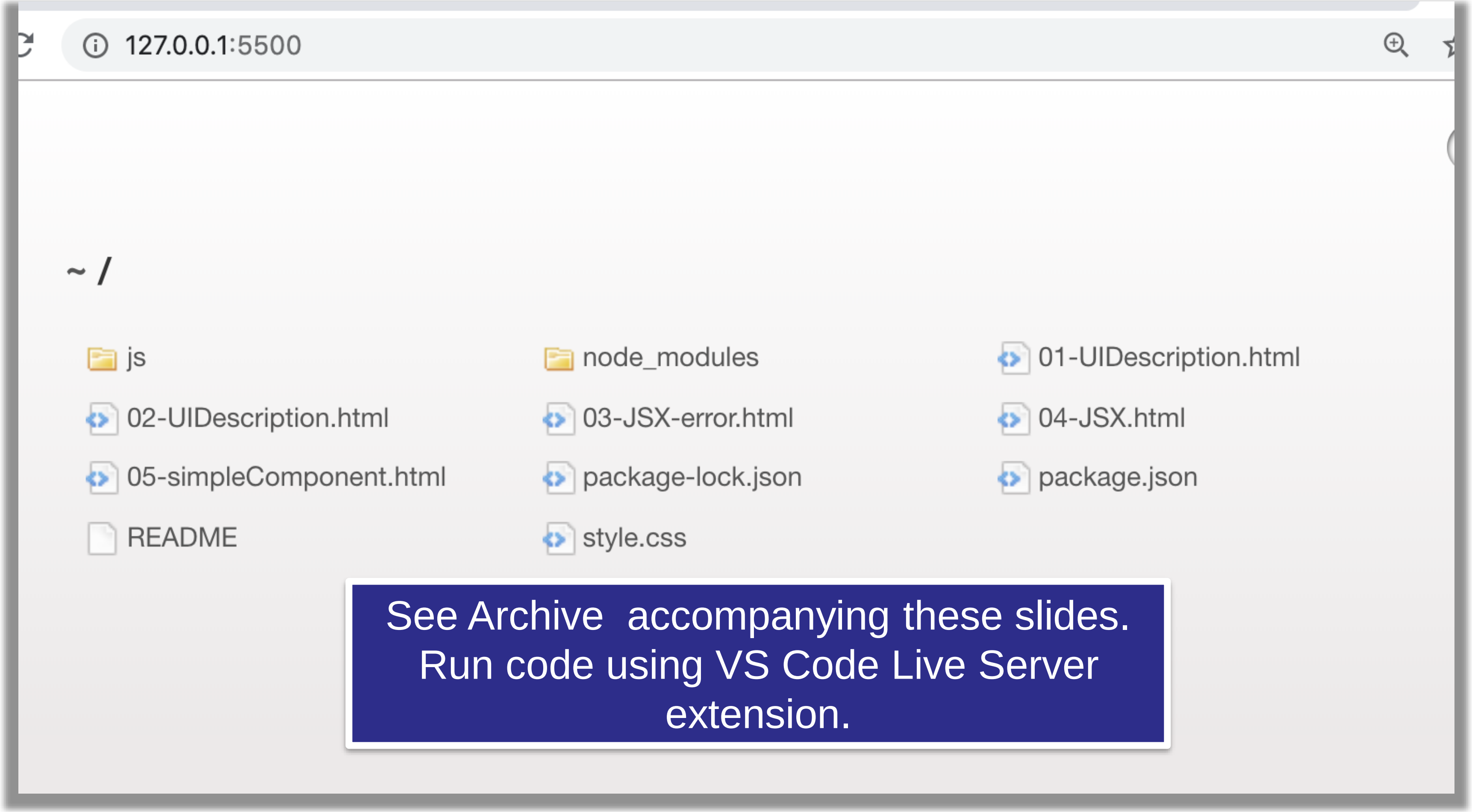




- Used to add type definitions to JavaScript codebases.
- TypeScript supports JSX (=>TSX)
- Include in your React project using **@types/react** and **@types/react-dom**
- Needs to be **Transpiled/Compiled** to Javascript to run in Browser/Client.

In-Class Code Demos.

(See lecture archive)



127.0.0.1:5500

~ /

js
node_modules
01-UIDescription.html
02-UIDescription.html
03-JSX-error.html
04-JSX.html
05-simpleComponent.html
package-lock.json
package.json
README
style.css

See Archive accompanying these slides. Run code using VS Code Live Server extension.

Creating the UI Description.

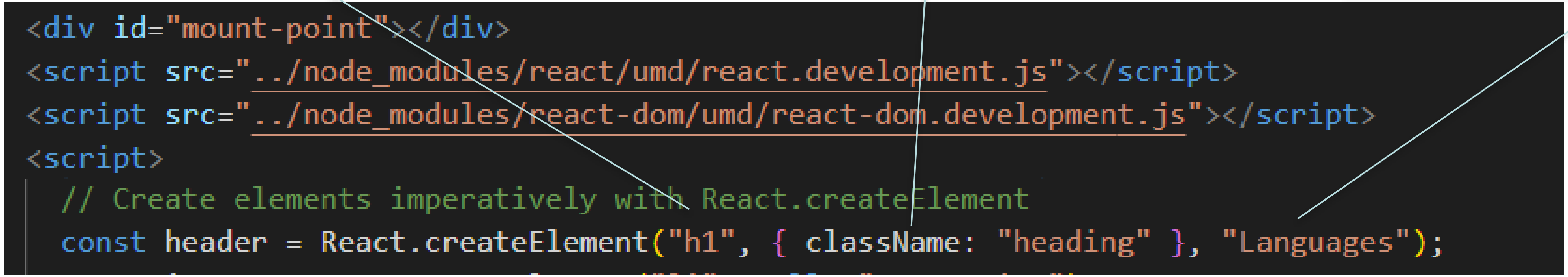
(Vanilla React)

- React.createElement() **– creates a HTML element.**
- ReactDOM.createRoot() – **used to attach the created element to existing DOM node.**
- React.createElement() takes three or more **arguments**:

Type (e.g. h1, div, …)

Properties (style, event handler…)

Children(0 to many child elements)

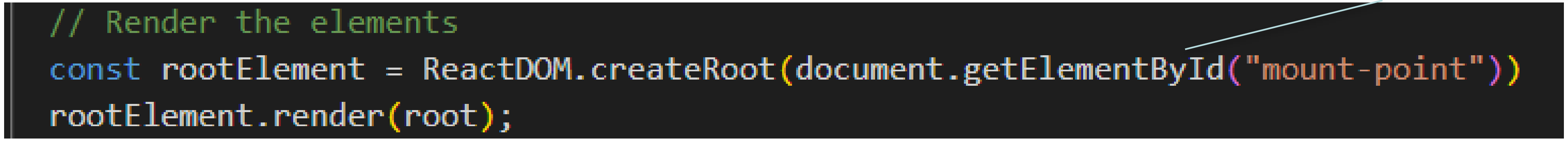


```
<div id="mount-point"></div>
<script src="../node_modules/react/umd/react.development.js"></script>
<script src="../node_modules/react-dom/umd/react-dom.development.js"></script>
<script>
  // Create elements imperatively with React.createElement
  const header = React.createElement("h1", { className: "heading" }, "Languages");
```

  – **We don't use createElement() directly - too cumbersome. More later…**

- ReactDOM.createRoot () has 1 **argument:**

DOM element on which to render your React Root Element



```
// Render the elements
const rootElement = ReactDOM.createRoot(document.getElementById("mount-point"))
rootElement.render(root);
```

UI Description Implementation

(the imperative way)

- **See the demos:**
  - **Ref.** 01-UIDescription.html.
  - **Nesting** createElement**() calls (Ref**. 02-UIDescription.html)

- **Imperative programming** is a programming paradigm that uses statements that change a program's state.

  focuses on describing how a program operates, step by step.

- **Declarative programming** is a programming paradigm … that expresses the logic of a computation without describing its control flow.

  focuses on what the result should be without specifying how it should achieve the results

UI description implementation

(the declarative way)

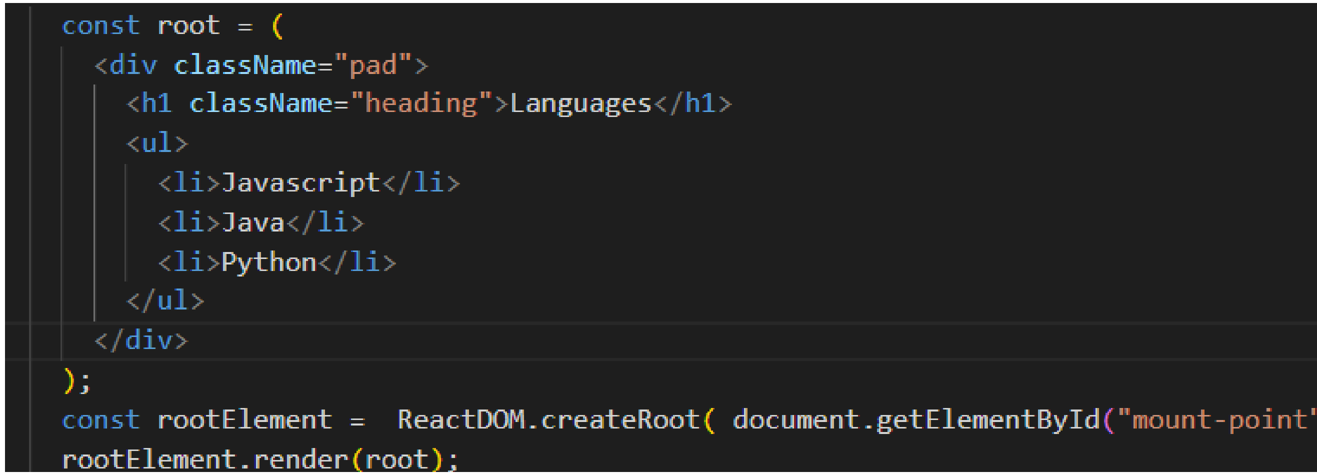
- **TSX – TypeScript XML.**

- Declarative syntax for coding UI descriptions.

- Retains the full power of Typescript.
- Allows tight coupling between UI behavior and UI description.

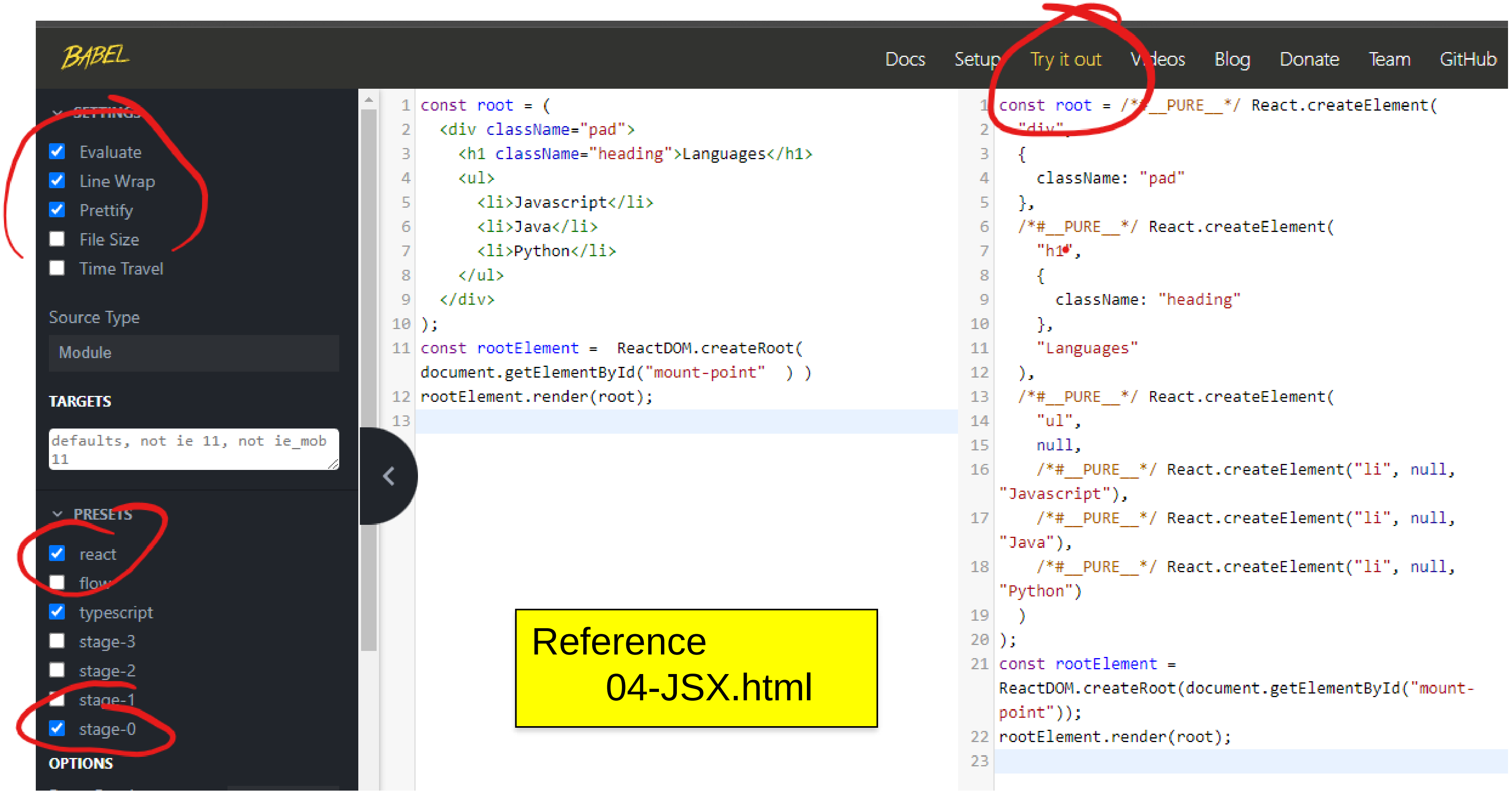
- **However, must be transpiled before being sent to browser.**

- **Reference** 03-JSX-error.html and 04-JSX.html



```
const root = (
  <div className="pad">
    <h1 className="heading">Languages</h1>
    <ul>
      <li>Javascript</li>
      <li>Java</li>
      <li>Python</li>
    </ul>
  </div>
);
const rootElement =  ReactDOM.createRoot( document.getElementById("mount-point"
rootElement.render(root);
```

REPL (Read-Evaluate-Print-Loop) transpiler.



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

```
const root = (
  <div className="pad">
    <h1 className="heading">Languages</h1>
    <ul>
      <li>Javascript</li>
      <li>Java</li>
      <li>Python</li>
    </ul>
  </div>
);
const rootElement =  ReactDOM.createRoot(
document.getElementById("mount-point"  ) )
rootElement.render(root);
```

```
const root = /*#__PURE__*/ React.createElement(
  "div",
  {
    className: "pad"
  },
  /*#__PURE__*/ React.createElement(
    "h1",
    {
      className: "heading"
    },
    "Languages"
  ),
  /*#__PURE__*/ React.createElement(
    "ul",
    null,
    /*#__PURE__*/ React.createElement("li", null,
"Javascript"),
    /*#__PURE__*/ React.createElement("li", null,
"Java"),
    /*#__PURE__*/ React.createElement("li", null,
"Python")
  )
);
const rootElement =
ReactDOM.createRoot(document.getElementById("mount-
point"));
rootElement.render(root);
```

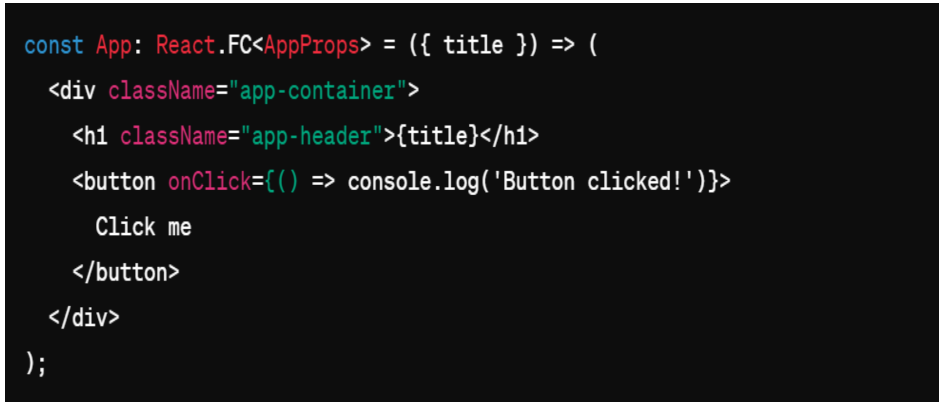
Reference
04-JSX.html

TSX(TypeScript XML)

- **JSX(Javascript XML)** is a file syntax extension used by React. JSX resembles HTML, allowing developers to write UI components with an HTML-like (it is actually XML).
- **TSX is the TypeScript version of JSX,** TypeScript is a superset that adds static typing in JavaScript.
  - Typescript files containing TSX use the .tsx extension.
- Some minor HTML tag attributes differences, e.g. className (class), htmlFor (for).

- Allows UI description to be coded **in a declarative style** and be in-lined in TypeScript.

- **Combines templating with the power of TS.**



```
const App: React.FC<AppProps> = ({ title }) => (
  <div className="app-container">
    <h1 className="app-header">{title}</h1>
    <button onClick={() => console.log('Button clicked!')}>
      Click me
    </button>
  </div>
);
```

TSX Transpiling

- **So browsers don’t do Typescript!**
  - Needs to be **Transpiled to Javascript**
- What can we use to Transpile?
  - The Babel platform.
  - The Vite library.

- How?
  1. Manually, via REPL environment or command line.
     - When experimenting only.
  2. Using an instrumented web server – Vite library instrumentation.
     - Ideal for development.
  3. Using bundler tools as part of the build process – Vite again.
     - Production standard.

React Components.

- **We develop COMPONENTS.**
  - A TS function that returns a UI description, i.e. TSX**.**

- **We reference a component like a HTML tag.**

  e.g.

```
const rootElement =
     ReactDOM.createRoot(document.getElementById("mount-point"));
rootElement.render( <DynamicLanguages/> );
```

- **Reference** 05-simpleComponent.html