Contents

[I. INTRODUCTION 2](#_Toc123509117)

[II. ES6 3](#_Toc123509118)

[1. Arrow Functions 3](#_Toc123509119)

[2. Enhanced Object Literals 3](#_Toc123509120)

[3. Destructuring 3](#_Toc123509121)

[4. Spread 5](#_Toc123509122)

[5. JS Modules 5](#_Toc123509123)

[III. REACT, REACT DOM 6](#_Toc123509124)

[1. document.createElement() 6](#_Toc123509125)

[2. Add React to your project 6](#_Toc123509126)

[3. React.createElement() 7](#_Toc123509127)

[4. React DOM 8](#_Toc123509128)

[IV. JSX, COMPONENTS AND PROPS 9](#_Toc123509129)

[1. JSX 9](#_Toc123509130)

[2. React Element Types 10](#_Toc123509131)

# INTRODUCTION

SPA: Single Page Application

MPA: Multi-Page Application

CSR: Client-Side Rendering

SSR: Server-Side Rendering

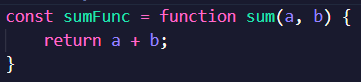
|  |  |
| --- | --- |
| **SPA** | **MPA** |
| * Modern * Trending * Page reloading is not required * Most necessary resources is loaded on the first run * Only need to load new resources when needed * Back-end and front-end independence * Not SEO friendly as MPA * Better user experience on mobile devices * JS dependency * Client-side rendering -> Reduce load for server | * Traditional * Need to reload page * Need to reload the entire page when accessing or routing * More SEO friendly than SPA |

# ES6

## Arrow Functions

-Reduce code length:

Normal:



Arrow:



-To return an object with arrow functions, we must put the object inside **()**:



-If there is **only 1 argument**, we can ignore **()**:



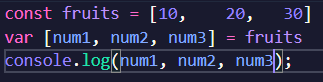
-Arrow function **does not have context** => **Cannot use *this***

## Enhanced Object Literals

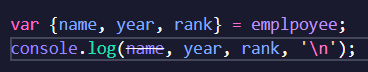
-In **reduce()** function, we must remember to return **accumulator**

## Destructuring

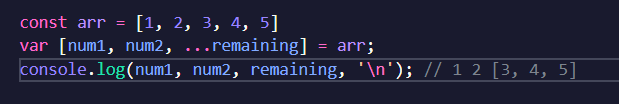
-Destructure an array:

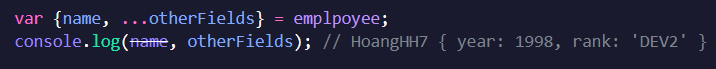


-Destructure an object:

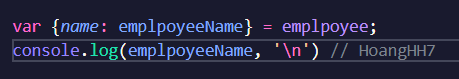


-Rest operator **…var**



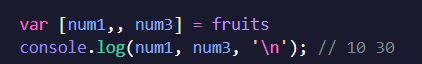


-Change variable’s name when working with objects:



-Omit some elements in the aray.

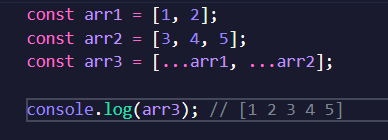
**Ex:** [10, 20, 30]



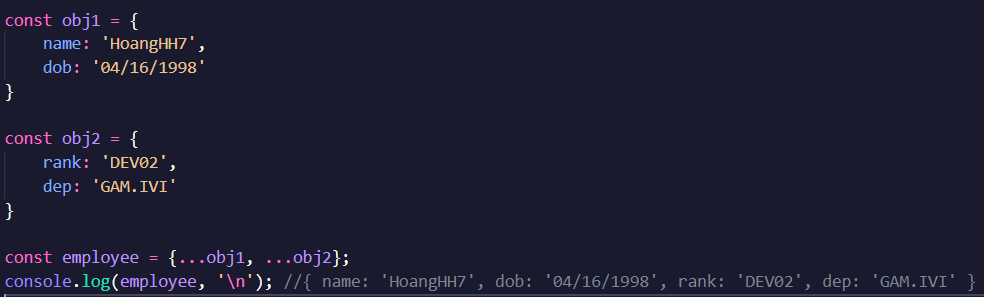
## Spread

-Use the spread operator to “remove” **[]** or **{}**

-Concatenating arrays:



-Concatenating objects:



## JS Modules

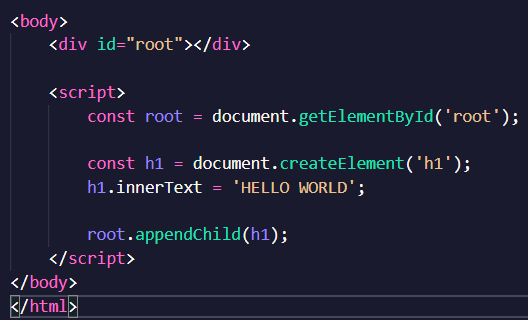
**export default** <function>

**import** <function> **from** <file>

# REACT, REACT DOM

## document.createElement()

-How JS can render HTML elements ?



## Add React to your project

-**NPM** = Node Package Manager

-**UNPKG**

+ Acquire NPM packages as CDN

+ Syntax:

**unpkg.com/:package@:version/:file**

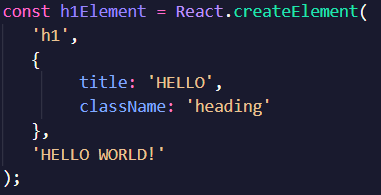
-To add React with UNPKG, use following script:

<script crossorigin src="https://unpkg.com/react@17.0.0/umd/react.production.min.js"></script>

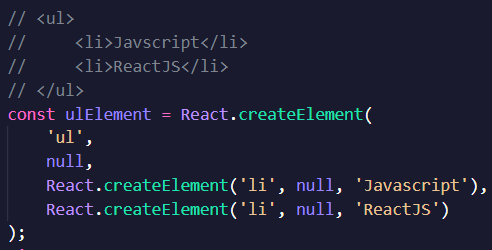
## React.createElement()

-React element is the smallest component of the React library.

-**React.createElement(**<type>, <props>, <childrens…>**)**. Children is also considered as **props**.



-Create element with multiple levels:



## React DOM

-Bridge between React and DOM

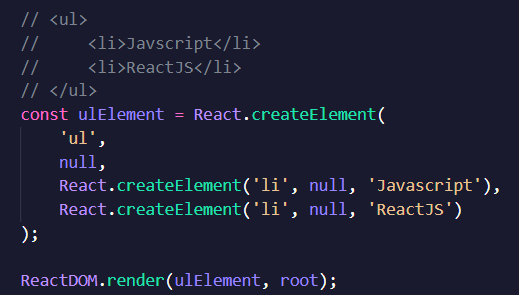
-ReactDOM supports rendering React elements.

-Add ReactDOM:

<script crossorigin src="https://unpkg.com/react-dom@17.0.0/umd/react-dom.production.min.js"></script>

-Syntax:

**ReactDOM.render(**<element>, <container>**)**

****

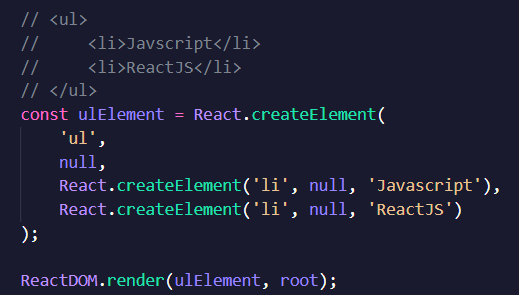
# JSX, COMPONENTS AND PROPS

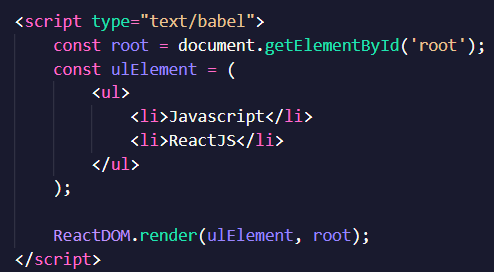
## JSX

-**JSX** = Javascript XML. **JSX** is **NOT HTML**.

-**Babel** translates XML to React code.

-Put JS code inside **{}** of **JSX**.

****  
  
  
**=>**

****

**-**Dynamically creates React elements from an array:



**-Array.map()**: Create a new array with modification to elements. **Map()** is often used for creating React elements from an array.

## React Element Types

-**React.createElement()** support types: function, class and string.

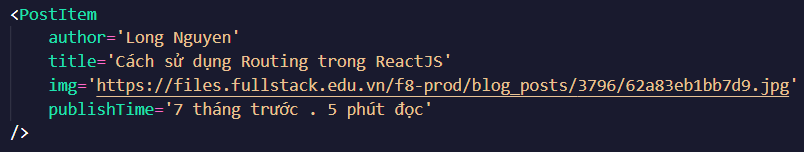
-Function component:



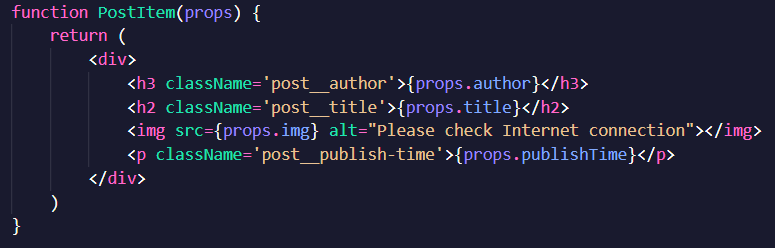
## Props

-Is used to pass data from parent components to child components. It is similar to passing arguments to functions.

-Use props like attributes of a HTML tag:



-To access an attribute, we can use: **props.<***attribute\_name***>**

****

## DOM Events

-DOM events list: [HTML DOM Event Object (w3schools.com)](https://www.w3schools.com/jsref/dom_obj_event.asp)

-HTML DOM event: click => onclick

-React DOM event: **Camel case** (onClick)

**Note**: dblclick => onDoubleClick

## Naming Convention

-Name of a component **must start with a UPPERCASE**. Ex: App, Component, …

-We can use a variable to set the name of a component:

**Ex:**

let Component = ‘button’;

return <Component>CLICK ME!</Component>

* Rendered HTML:

<button>CLICK ME!</button>

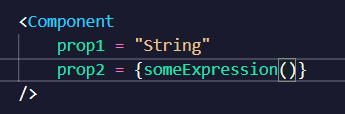
## Dynamic Component Type

-Reference: [JSX In Depth – React (reactjs.org)](https://reactjs.org/docs/jsx-in-depth.html#choosing-the-type-at-runtime)

## Children Props – Render Props

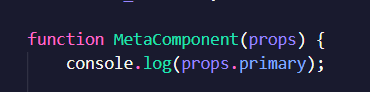
-Ways to pass props to a component: + String literals

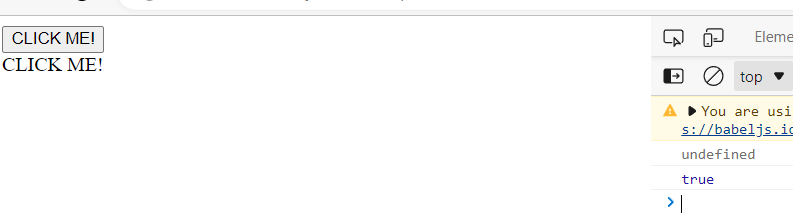
+ JS expressions



-Props **with no assignment** will be automatically considered as **true**







-**Spread props**: