

React (Weak 2.1)

Amr A Khllaf

March 6, 2025

Contents

| | | |
|----------|--|-----------|
| 1 | JS Revision (Array Destructuring, Ternary Operator, Map, Filter Methods) | 2 |
| 1.1 | index.html file | 2 |
| 1.2 | main.js file | 2 |
| 1.2.1 | Array Destructing | 2 |
| 1.2.2 | Ternary Operator | 3 |
| 1.2.3 | Filter Method (Array Method - ES6) | 5 |
| 1.2.4 | Map Method (Array Method - ES6) | 6 |
| 2 | Binding Concept | 7 |
| 2.1 | Home.jsx file | 7 |
| 2.2 | About.jsx file | 7 |
| 2.3 | index.css file (Main CSS File for all the components) | 8 |
| 3 | Events in JSX and the problem with Functional Component Variables | 9 |
| 3.1 | Gallery.jsx file | 9 |
| 4 | Discussing the Problem again and What is STATE | 11 |
| 5 | What is hooks, Where to use them and differences between state and variables | 12 |
| 5.0.1 | Comparison between State and Variables | 13 |
| 6 | Rendering Component in another Component & PROPS Concept | 13 |
| 6.1 | Why We need to make component in another component? | 15 |
| 6.1.1 | So How to send data from component to another component? | 16 |
| 6.1.2 | Parent.jsx file | 16 |
| 6.1.3 | Child.jsx file without using destructuring | 17 |
| 6.1.4 | Child.jsx file using destructuring (Best Practice) | 17 |
| 7 | Sending Object and Destructuring it to the Child Component | 18 |
| 7.1 | Parent.jsx file | 18 |
| 7.2 | Child.jsx file without using Object destructuring | 19 |
| 7.3 | Child.jsx file using Object destructuring | 19 |
| 8 | What if We Have Many Products, How to list them & What is the difference between map and forEach? | 20 |
| 8.1 | Parent.jsx file | 20 |
| 8.2 | Interview Question: What is the difference between map and forEach? | 21 |

| | | |
|-------|--|----|
| 8.2.1 | Example | 21 |
| 8.2.2 | Note on Parent and Child | 22 |
| 8.3 | Child.jsx file | 22 |
| 8.3.1 | Bootstrap hint for grid system | 23 |

1 JS Revision (Array Destructuring, Ternary Operator, Map, Filter Methods)

1.1 index.html file

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
      ↪ />
    <title>Js Revision</title>
  </head>
  <body>
    <script src="./js/main.js"></script>
  </body>
</html>
```

1.2 main.js file

1.2.1 Array Destructuring

```
"use strict";
# This line is used to enable strict mode in JS

function sayHello(){
  console.log("Hello");
}
// Array can contain any data type
const allAges = [20,"Khllaf" ,sayHello, 30];

let x = allAges[0];
let y = allAges[1];
let z = allAges[2];

# but there is another way which is better and more readable, which is
↪ array destructuring

// Destructing => can be used with arrays and objects (any reference Data
  ↪ type)

// Destructing => "3amlya 3aksya latre2t el Declaration"

// const [x] = allAges; // ==> this like this => let x = allAges[0];
```

```
const [x,y,z] = allAges; // ==> this like this => let x = allAges[0]; let
  ↳ y = allAges[1];
# We Will See this line a lot in react

console.log(x , y); // 20 Khllaf
console.log(z); // [Function: sayHello]
// This is mean the reference of the function sayHello is stored in z

# I Can call the function like this
sayHello(); // Hello

# but i can do this also:

z(); // Hello
// this is mean u have the reference of the function in z, So call it and
  ↳ this is the concept of Destructing
```

1.2.2 Ternary Operator

```
let userAge = 30;

if (userAge > 18) {
  console.log("Hello Adult");
} else {
  console.log("Hello Child");
}

# This line has a little problem which i write the code in multi lines ,
  ↳ but sometimes i will need to write the code in one line
# So i can use the "Ternary Operator"

// Ternary Operator
// Syntax => condition ? true : false

userAge > 18 ? console.log("Hello Adult") : console.log("Hello Child"); //
  ↳ Hello Adult

# What is ternary operator?
// It's a short way to write if else statement
// It's a one line if else statement
// Ternary mean 3 parts
// condition ? true : false

# Some differences between if else and ternary operator

## if else

// 1. if else can be used with multiple conditions
```

```
// 2. i don't have to write the else part in the ternary operator , i can  
→ write only the if part  
// 3. I have scope in the if else statement, so i can write multiple lines  
→ of code  
// 4. I can use else if in the if else statement
```

```
if (userAge > 18) {  
  console.log("Hello Adult");  
} else if (userAge < 18) {  
  console.log("Hello Child");  
} else {  
  console.log("Hello Teenager");  
}
```

Ternary Operator

```
// 1- Ternary Operator can be used with one condition only  
// 2- Ternary Operator must have the else part  
// 3- I don't have scope in the ternary operator, so i can't write  
→ multiple lines of code(only one line)
```

```
## But if i want to write multi line in the ternary operator, i can do this  
→ by using the IIFE (Immediately Invoked Function Expression)
```

```
// example:
```

```
userAge > 18 ? (() => {  
  console.log("Hello Adult");  
  console.log("Hello Adult");  
  console.log("Hello Adult");  
})() : console.log("Hello Child");
```

```
## or i can use a normal function in ternary operator
```

```
userAge > 18 ? function(){  
  console.log("Hello Adult");  
  console.log("Hello Adult");  
  console.log("Hello Adult");  
}() : console.log("Hello Child");
```

```
// 4.If i want to use else if in the ternary operator, i can do this by  
→ using "the nested ternary operator"
```

```
userAge > 18 ? console.log("Hello Adult") : userAge < 18 ?  
→ console.log("Hello Child") : console.log("Hello Teenager");
```

```
// Hello Adult
```

```
# Nested Ternary Operator Woks Like This:

if (userAge > 18) {
  console.log("Hello Adult");
}
else {
  // ==18 or < 18
  if (userAge < 18) {
    console.log("Hello Child");
  }
  else {
    console.log("Hello Teenager");
  }
}
}
```

1.2.3 Filter Method (Array Method - ES6)

- The filter() method creates a new array with all elements that pass the test implemented by the provided function.
- it will loop over the array and check the condition, if the condition is true, it will add the element to the new array, if the condition is false, it will not add the element to the new array and it returns the new array with the condition i want to filter by.
- it takes a callback function as a parameter, and this callback function takes 3 parameters (element, index, array)

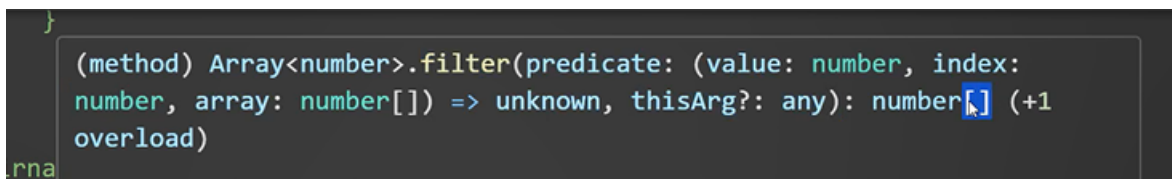


Figure 1: filter Method Return

- if the method take a function as a params , it will be one of two :
 1. if u have already the function implemented, u can pass the function name as a parameter (function reference)

” allAges.filter(sayHello);”

2. if u don't have the function implemented, u can write the function directly as a parameter (anonymous function)

```
let allAges = [20, 8, 10, 23, 30];

allAges.filter(function (age) {
  return age > 18;
}); // [20, 23, 30]
```

```
# or i can use the arrow function

allAges.filter((age) => {
  return age > 18;
}); // [20, 23, 30]

# or i can use the arrow function in one line if the return statement is
↳ the only statement in the function

allAges.filter( age => age > 18); // [20, 23, 30]
```

1.2.4 Map Method (Array Method - ES6)

- The map() method creates a new array populated with the results of calling a provided function on every element in the calling array.
- it will loop over the array and apply the function on each element and return the new array with the new values.
- it takes a callback function as a parameter, and this callback function takes 3 parameters (element, index, array)
- amp change the values of the array and return the new(Same) array with the new values

```
let allAges = [20, 8, 10, 23, 30];

// age parameter is the element in the array
allAges.map(function (age) {
  return age * 2;
}); // [40, 16, 20, 46, 60]

# Trick to understand map method

allAges.map(function (age) {
  return "Test";
}); // ["Test", "Test", "Test", "Test", "Test"]

# "Ka2ny ba2olo kol el elements fe el array bt3dy 3leeh yeb2a 'Test'"

# the Benefit of this , i can make like this :

const res = allAges.map(function(age){ return `<li>${age}</li>`; });

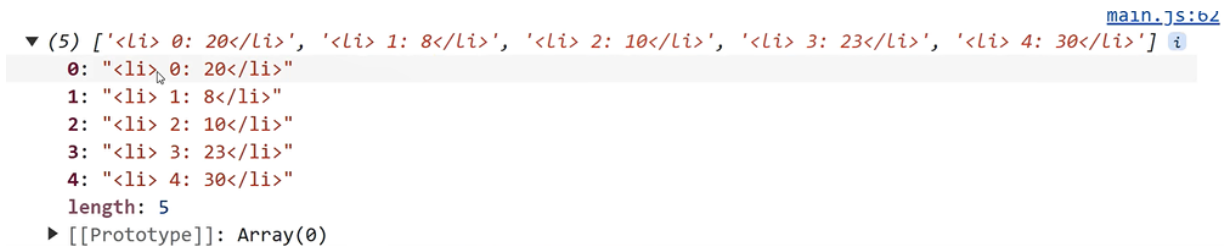
# use arrow function

const res = allAges.map( (age, x) => `<li> ${x}: ${age}</li>`);

console.log(res); // [<li>0: 20</li>", "<li>1: 8</li>", "<li>2: 10</li>",
↳ "<li>3: 23</li>", "<li>4: 30</li>"]

# Diff between map and for loop
```

- i can write the map **in** one line
- i can't write the for loop in one line
- **in** react we will use to write ternary operator and map method a lot



```
main.js:64
▼ (5) ['<li> 0: 20</li>', '<li> 1: 8</li>', '<li> 2: 10</li>', '<li> 3: 23</li>', '<li> 4: 30</li>'] ⓘ
  0: "<li> 0: 20</li>"
  1: "<li> 1: 8</li>"
  2: "<li> 2: 10</li>"
  3: "<li> 3: 23</li>"
  4: "<li> 4: 30</li>"
  length: 5
  __proto__: Array(0)
```

Figure 2: map Method

2 Binding Concept

2.1 Home.jsx file

```
import React from "react";

export default function Home() {
  const userName = "Khllaf";
  // Binding Concept => connect the JS Code with the JSX Code

  // Old Way(3la 2demmo) ti Write a var in a string , i have to use :
  // `Hello ${userName}`

  // Binding syntax: Any JSX Position ==> Add {} (I can add the {} in any
  // position of jsx code) to write JS Code, it's one line only

  return (
    <>
      <h2>Hello ya {userName}</h2>
      <p>
        Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam quia
        quae
        quaerat repellat officiis perspiciatis!
      </p>
    </>
  );
}
```

2.2 About.jsx file

```
import React from "react";

export default function About() {
```

```
// Ex : When the backend Dev give me the data from the backend, i have
↳ to show it in the front end even if it was a test1 or test2 class

const clsName = "test2";
return (
  <React.Fragment>
    // If i want to add more than class .. there is 2 ways to do this :
    ↳ 1.
      className = {"padding " + "margin " + clsName}
      // Note : Don't forget the space after the class name, but the {} is
    ↳ a js code
      in one line 2. Use the Template Literals which is the best way to do
    ↳ this :
      className={`padding margin ${clsName}`}
      <div className={`padding margin ${clsName}`}>
        // I can use the Binding Concept in the className attribute
        <h2>About Age</h2>
      </div>
    </React.Fragment>
  );
}
```

2.3 index.css file (Main CSS File for all the components)

- I will use & write the same css file for all the components, React make this file for us to use it in all the components

```
.test {
  background-color: red;
  color: white;
}

.test2 {
  background-color: blue;
  color: white;
}

.test3 {
  background-color: violet;
  color: white;
}

.padding {
  padding: 10px;
}

.margin {
  margin: 10px;
}
```

- The Div With test class in the About component will take the properties of this css file

3 Events in JSX and the problem with Functional Component Variables

3.1 Gallery.jsx file

```
import React from "react";
import { useState } from "react";
export default function Gallery() {
  // let counter = 100;

  const [counter, setCounter] = useState(100);
  // due to this is a function , I can do "many useStates in the same
  ↪ component"

  // let userName = "";
  const [userName, setUserName] = useState("");
  function incrementCounter() {
    // counter++;
    // Change By Using the function that provided by the useState
    ↪ (setCounter)
    setCounter(counter + 1);
  }

  function showUserName() {
    // userName = "Khllaf";
    setUserName("Khllaf");
    console.log(`User Name: ${userName}`);
  }

  console.log("Increment Counter", counter);
  return (
    <>
      <div className="test3 padding">
        <h2>Gallery Component</h2>
        <h2>Counter: {counter}</h2>
        <h2>User Name: {userName}</h2>

        { /* NOT HTML Code

          Events in JSX and HTML

          1. in HTML events => small letters , JSX events => camelCase

          2. in HTML used to call your function , JSX used to pass
          ↪ reference of your function(Like addEventListener)
```

- if my function take a parameter , i can use the anonymous
→ function to pass the parameter to the function
- like this `onClick={() => incrementCounter(5)}`

```
    */}  
    <button onClick={incrementCounter} className="test padding">  
      Click  
    </button>  
    <button onClick={showUserName} className="test padding margin">  
      Show USer Name  
    </button>  
  </div>  
</>  
);  
}
```

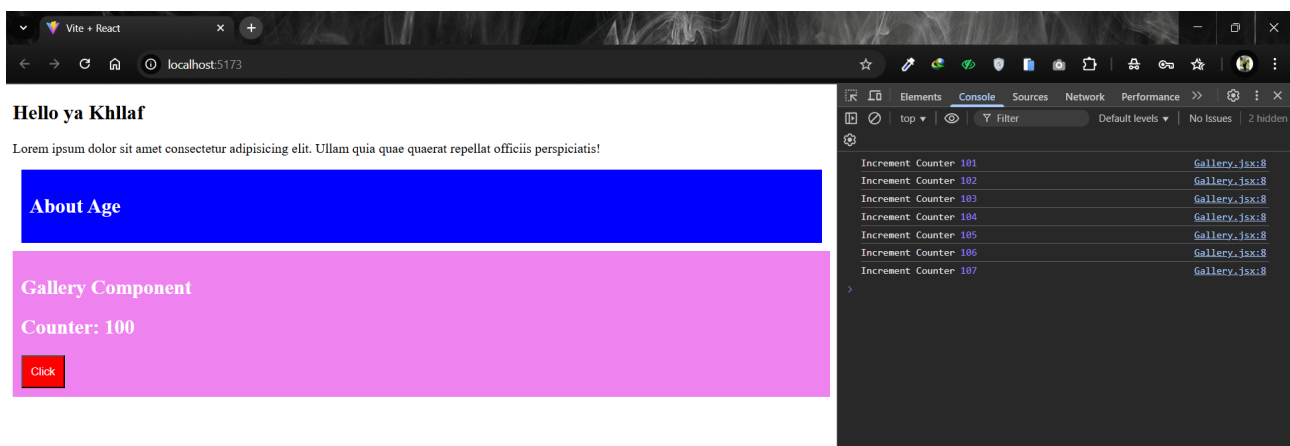


Figure 3: Counter Problem

- Every time u click on the button , the counter will be updated in our console but not in the UI , because u have rendered the component only one time and the counter is a variable in the functional component, so it will not be updated in the UI.
- To solve this problem, we have to use the state in the functional component

4 Discussing the Problem again and What is STATE

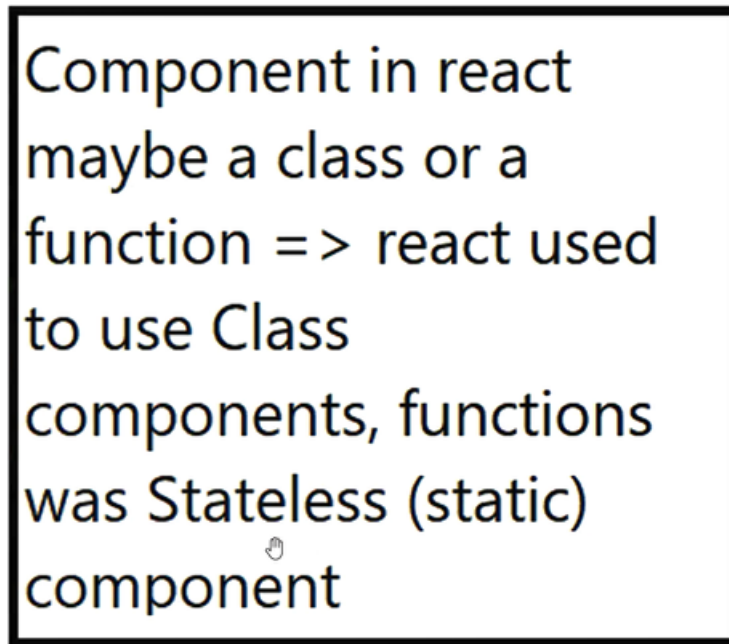


Figure 4: Stateless Function

- Stateless Function(static): The function that doesn't have a state, so in the past we used to use the class component to have a state in our component, and use the function component with static elements only.
- "state?" => is the data that i want to change in my component (container include your data)
 - Data Like:
 - variables
 - var x = 30
 - var x = localStorage
 - var x = APIs
- react use the "state" to "Re-render" your component that has this state(it's like the display function in CRUD Operations)
- So the difference between Class Component and Functional Component is the state , the class component has a state and the functional component doesn't have a state
 - So if you want to Re-render your component, you have to use the "state" in your component
- After React 16.8, we can use the state in the functional component by using the "Hooks" Concept

5 What is hooks, Where to use them and differences between state and variables

- Hooks are functions that let you use state and other React features in functional components
- Hook is a function must start with “use” keyword
 - like state => useState() =>
 - * Note: You can't use any hook “outside a functional component scope”.
- to Use the useState hook, you have to import it from the react library

```
import { useState } from "react";
```

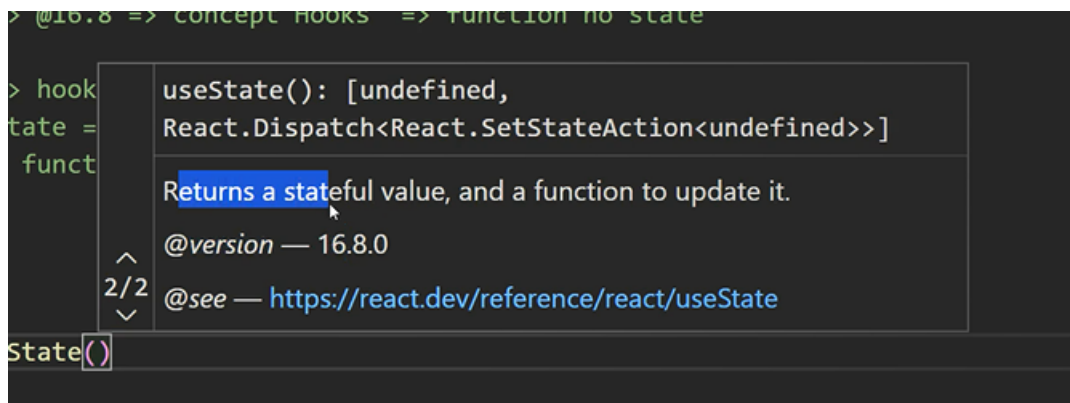


Figure 5: useState

- `useState()` => return an “array with 2 elements”
 - 1. The state variable (stateful value)
 - 2. The function to update the state variable, throw this function you can Re-render & Update your component
 - [state value , function to update this value]
 - So the rule of Destructing is applied here

```
const [counter, setCounter] = useState(100);  
  
function incrementCounter() {  
  setCounter(counter + 1); // Re-render the component  
}  
  
console.log("Increment Counter", counter);  
// The counter will be updated in the UI  
  
return (  
  <>  
    <div className="test3 padding">  
      <h2>Gallery Component</h2>  
      <h2>Counter: {counter}</h2>  
      <button onClick={incrementCounter} className="test padding">  
        Click  
      </button>  
    </div>  
  </>  
)
```

```
</>  
);
```

- There is a snippet for the useState hook in the react library, so u can write “useStateSnippet” and press tab to get the snippet

5.0.1 Comparison between State and Variables

- Interview Question: What is the difference between State and Variables in React?
1. All of them their values can be changed (mutable)
 2. The difference is in the Re-rendering
 - Variables => if u change the value of the variable, it will not be updated in the UI
 - State => if u change the value of the state, it will be updated in the UI
 3. In Re-Render => State still holding it's previous value, but the variable will not hold it's previous value

- EX:

```
let hamada= 30;
```

```
const [counter, setCounter] = useState(100);
```

```
let hamada = 50;
```

```
const [state, setState] = useState(200);
```

- if i go and change the state value to 200, the state will hold the
→ previous value (100) and the new value (200) but the variable will not
→ hold the previous value (30) but it will hold the new value (50)

- For Ex : if u have a button and change the value of the variable , then
→ you re-render the page the value of the variable will be the default
→ value but the state will hold the previous value

6 Rendering Component in another Component & PROPS Concept

- To Start This Concept u need to remember the relation between the parent and child and the child with child called (Siblings)

for Ex:

```
<div>  
  </h2>Hello</h2>  
  </h2>Hello</h2>  
  </h2>Hello</h2>  
</div>
```

- Div is the parent of the h2, and the all th h2 are siblings
- This Concept is exist in HTML Code and also it's exist in JSX Code
- like:

```
export default function App() {  
  return (  
    <>  
      <Home />  
      <About />  
      <Gallery />  
    </>  
  );  
}
```

- Home, About, Gallery are siblings
- What we need to make a relation of parent and child like the siblings between Home, About, Gallery
- We will do 2 new Components Parent and Child
 - This Concept will happen if you render a component in another component
- Child Component

```
import React from "react";  
  
export default function Child() {  
  return (  
    <>  
      <h2>Child</h2>  
    </>  
  );  
}
```

- Parent Component

```
import React from "react";  
import Child from "../Child/Child";  
  
export default function Parent() {  
  return (  
    <>  
      <h2>Parent Component</h2>  
      <Child />  
    </>  
  );  
}
```

- App Component

```
import React from "react";
```

```
import Home from "../Components/Home/Home";
import About from "../Components/About/About";
import Gallery from "../Components/Gallery/Gallery";
import Parent from "../Components/Parent/Parent";
import Child from "../Components/Child/Child";

export default function App() {
  return (
    <>
      { /* <Home />
        <About />

        <Gallery /> */ }

      <Parent />

      { /* <Child /> */ }
    </>
  );
}
```

- So the Parent Component is the parent of the Child Component
- This Is happen from the first time we create the first component in the react app
- The App Component is the parent of the Home, About, Gallery Components

6.1 Why We need to make component in another component?

- Bkol Basata ana delwa2ty 3ayz a3ml navbar fe kol el pages, ma3kol kol shakl 3ndy 3la kol page, so i can make a component for the navbar and render it in all the pages
- bardo mommkn a3ml component gwa el navbar wa 2dr andeh kman f el footer
- el component hwa asln 3bara 3n function w eshan kda a3dr anadeh (render) 200 alf mra fe el code
- I do this because the component may be page or sup-page



Figure 6: Page Sup-Page

6.1.1 So How to send data from component to another component?

- To Send Data from Component to another Component => Data must be sent from parent component to child component not vice versa (React Condition)
- You can send your data using the “PROPS” (Properties) concept
 - “PROPS” is written in the Component Selector (Like attributes in HTML)
 - “PROPS” is done in 2 steps
 1. Send the data from the parent component to Child Component and this is done in Parent Component
 2. Display the data in the Child Component and this is done in Child Component
 - * Function Components receives their PROPs in the first parameter of your function

6.1.2 Parent.jsx file

```
import React from "react";
import Child from "../Child/Child";
import { useState } from "react";

export default function Parent() {
  // const userName = "Khllaf";
  // For Ex : this is the home Page
  const [user, setUser] = useState("Amr");
  return (
    <>
      <div className="test3 margin padding">
        { /* <h2>Hello From {userName}</h2> */ }
        { /* <h2>Hello From {user}</h2> */ }
        { /* */ }
        <h2>Hello in Home Page</h2>
      </div>
    </>
  );
}
```


Why We need to make component in another component?

```
    /* I need to Take the name and give to child and i will will it
    ↪ appear from the child component */

    /* ----- */

    /* Navbar */
    <Child name={user} age={30} salary={3000} />
    /* this mean i have sent (Data) to the child component
    - to make the var in integer type .. you need to use the {} to
    ↪ make it as a JS code .. like salary={30}, age="30" this mean it is a
    ↪ string
    - i can make the Binding by using name of the prop in the child
    ↪ component .. like name={user} (Dynamic Binding)
    */
    </div>
  </>
);
}
```

6.1.3 Child.jsx file without using destructuring

```
import React from "react";

export default function Child(props) {
  // - Function Components receives their PROPs in the "first parameter"
  ↪ of your function like event info i n function in addEventListener
  // - The PROPs is an "object" that contains all the properties that you
  ↪ have sent when you rendered this component
  // - props is the "object" that contains the properties that i have sent
  ↪ from the parent component

  console.log(props); // {name: "Amr Ahmed", age: "30"}
  console.log(props.name); // Amr
  console.log(props.age); // 30
  console.log(props.salary); // 30

  return (
    <>
      <div className="test margin padding">
        <h2>UserName: {props.name}</h2>
        <h2>UserAge: {props.age}</h2>
        <h2>UserSalary: {props.salary}</h2>
      </div>
    </>
  );
}
```

6.1.4 Child.jsx file using destructuring (Best Practice)

```
import React from "react";
```

```
export default function Child({ name, age, salary }) {
  console.log(name); // Amr
  console.log(age); // 30
  console.log(salary); // 3000

  return (
    <>
      <div className="test margin padding">
        <h2>UserName: {name}</h2>
        <h2>UserAge: {age}</h2>
        <h2>UserSalary: {salary}</h2>
      </div>
    </>
  );
}
```

7 Sending Object and Destructuring it to the Child Component

- if the data that sent from backend is "Object"
- How To destruct Object:
 - 1. I can use the Object Destructuring

```
const ProductInfo = {
  name: "Iphone",
  price: 1000,
  category: "Mobile",
};
```

- To Destruct **this** object
- Destructing => "3amlya 3aksya latre2t el Declaration"

```
const { name, price, category } = ProductInfo;
```

7.1 Parent.jsx file

```
import React from "react";
import Child from "../Child/Child";
import { useState } from "react";

export default function Parent() {
  const [Product, setProduct] = useState({
    name: "Iphone",
    price: 1000,
    category: "Mobile",
  });
  return (
```

```
<>
  <div className="test3 margin padding">
    <h2>Hello in Home Page</h2>

    {/* ----- */}
    {/* Card that display the details of the product object */}

    <Child ProductInfo={Product} />
  </div>
</>
);
}
```

7.2 Child.jsx file without using Object destructuring

```
import React from "react";

export default function Child({ ProductInfo }) {

  return (
    <>
      {/* ----- */}
      <h2>Product Name is : {ProductInfo.name}</h2>
      <h2>Product Price is : {ProductInfo.price}</h2>
      <h2>Product Category is : {ProductInfo.category}</h2>
    </div>
  </>
);
}
```

- if i don't want to use the ProductInfo in the child component, i can use the Object Destructuring

7.3 Child.jsx file using Object destructuring

```
import React from "react";

export default function Child({ ProductInfo }) {

  const { name, price, category } = ProductInfo; // Object Destructuring
  ↪ (Best Practice)

  return (
    <>
      {/* ----- */}
      <h2>Product Name is : {name}</h2>
      <h2>Product Price is : {price}</h2>
      <h2>Product Category is : {category}</h2>
    </div>
  )
}
```

```
    </>
  );
}
```

8 What if We Have Many Products, How to list them & What is the difference between map and forEach?

- If i have many products and i want to list them in the child component
- I can use the map method to loop over the array and list the products

8.1 Parent.jsx file

```
import React from "react";
import Child from "../Child/Child";
import { useState } from "react";

export default function Parent() {
  const [allProducts, setAllProducts] = useState(
    [
      {
        name: "Iphone",
        price: 1000,
        category: "Mobile",
      },
      {
        name: "Samsung",
        price: 800,
        category: "Mobile",
      },
      {
        name: "Dell",
        price: 2000,
        category: "Laptop",
      },
      {
        name: "HP",
        price: 1500,
        category: "Laptop",
      },
      {
        name: "Sony",
        price: 500,
        category: "TV",
      },
      {
        name: "LG",
        price: 400,
        category: "TV",
      },
    ]
  );
}
```

Interview Question: What is the difference between map and forEach?

```
    }
  );
  return (
    <>
    <div className="test3 margin padding">
      <h2>Hello in Home Page</h2>
      { /* ----- */ }
      { /* Card that display the details of the product object */ }
      - I can Render the Child Component many times to list the products
    ↪ , but
      the problem is the code will be repeated and it's not dynamic code
    ↪ // <Child
      ProductInfo={allProducts[0]}
    />
    // <Child ProductInfo={allProducts[1]} />
    // So i will use the map method to loop over the array and list the
    ↪ products
      in binding
      {allProducts.map((product, index) => {
        return <Child key={index} ProductInfo={product} />;
        // product is like allProducts[0], allProducts[1],
        ↪ allProducts[2]
      })}
    </div>
  </>
);
}
```

8.2 Interview Question: What is the difference between map and forEach?

- foreach is like the map in Implementation but the map return a new array with the new values and the forEach doesn't return anything
- The map method is used to loop over the array and apply the function on each element and return the new array with the new values
- The forEach method is used to loop over the array and apply the function on each element but it doesn't return anything

8.2.1 Example

```
let allProducts = [
  {
    name: "Iphone",
    price: 1000,
    category: "Mobile",
  },
  {
    name: "Samsung",
    price: 800,
```

```
    category: "Mobile",
  },
  {
    name: "Dell",
    price: 2000,
    category: "Laptop",
  },
  {
    name: "HP",
    price: 1500,
    category: "Laptop",
  },
  {
    name: "Sony",
    price: 500,
    category: "TV",
  },
  {
    name: "LG",
    price: 400,
    category: "TV",
  },
];

// Map Method

let newProducts = allProducts.map((product) => {
  return product.name;
});

console.log(newProducts); // ["Iphone", "Samsung", "Dell", "HP", "Sony",
↪  "LG"]

// forEach Method

let newProducts = allProducts.forEach((product) => {
  return product.name;
});

console.log(newProducts); // undefined
```

8.2.2 Note on Parent and Child

- When you need to use or modify on the back end data » Use the Parent Component and make the child component as a presentational (display) component

8.3 Child.jsx file

```
import React from "react";

export default function Child({ ProductInfo }) {
```

```

const { name, price, category } = ProductInfo; // Object Destructuring
↳ (Best Practice)

return (
  <>
    { /* ----- */ }
    <h2>Product Name is : {name}</h2>
    <h2>Product Price is : {price}</h2>
    <h2>Product Category is : {category}</h2>
  </div>
</>
);
}

```

8.3.1 Bootstrap hint for grid system

- parent jsx file

```

<div className="container">
  <div className="row">
    {allProducts.map((product) => {
      return ( <Child ProductInfo={product} /> );
    })}
  </div>

```

- Child jsx file
 - is the part that represent the product and repeated many times as it was the column

```

<div className="test col-md-4 margin padding">

  <h2>Product Name is : {name}</h2>
  <h2>Product Price is : {price}</h2>
  <h2>Product Category is : {category}</h2>
</div>
</>

```



Figure 7: Grid System using Bootstrap
