

Laporan Praktikum

MODUL 1

**Instalasi Environment dan
Pengenalan React Tingkat Dasar**



Oleh:

Afifah Fikriyah (1203220051) IS05-02

PROGRAM STUDI S1 SISTEM INFORMASI

FAKULTAS REKAYASA INDUSTRI

UNIVERSITAS TELKOM SURABAYA

2025

Tugas

1. Convert functional component → class component

```
1  class MyButton extends React.Component {
2    handleClick = () => {
3      alert(this.props.message);
4    };
5
6    render() {
7      return (
8        <button onClick={this.
9          handleClick} style={{ margin: "10px" }}>
10          {this.props.children}
11        </button>
12      );
13    }
14  }
15
16  class MyApp extends React.Component {
17    render() {
18      return (
19        <div>
20          <h1>Learn React</h1>
21          <MyButton message="Learn React">
22            React</MyButton>
23          <MyButton message="
24            Learn JavaScript">JavaScript</MyButton>
25        </div>
26      );
27    }
28  }
29
30  const myElement = <MyApp />;
31  const myApp = myElement;
32
33  ReactDOM.createRoot(document.getElementById('
34    root')).render(myApp);
35
```

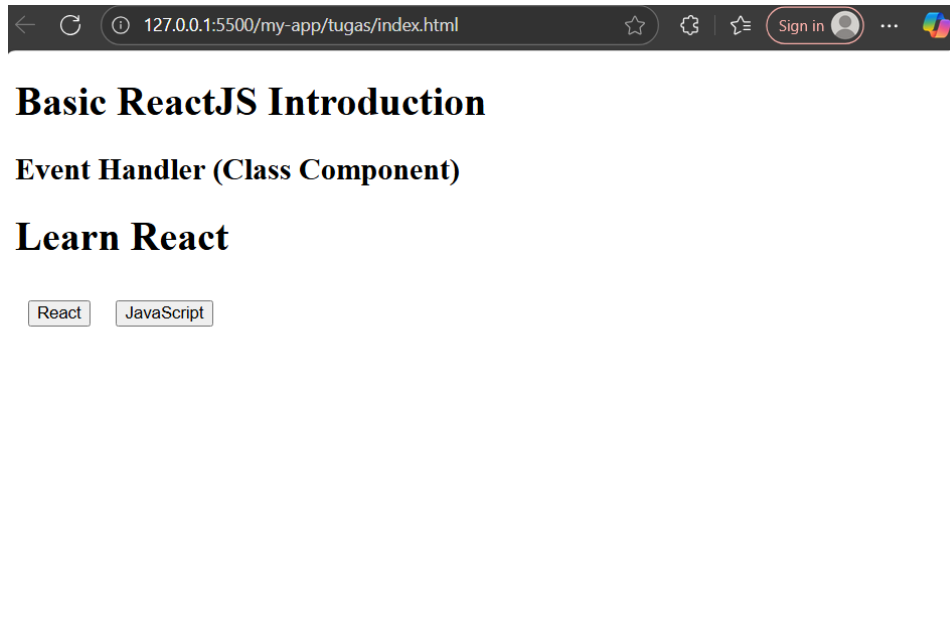
Perbedaan dengan functional component:

Function myButton → class myButton extends React.Component

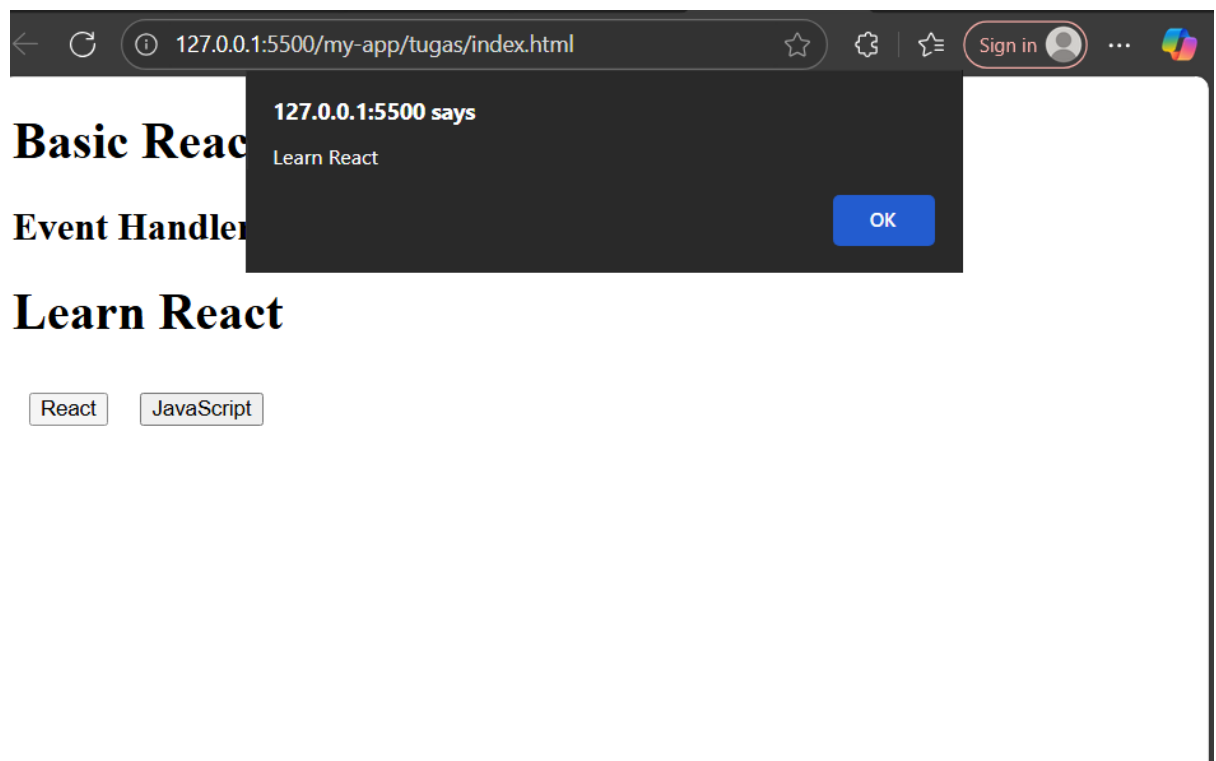
Props → this.props

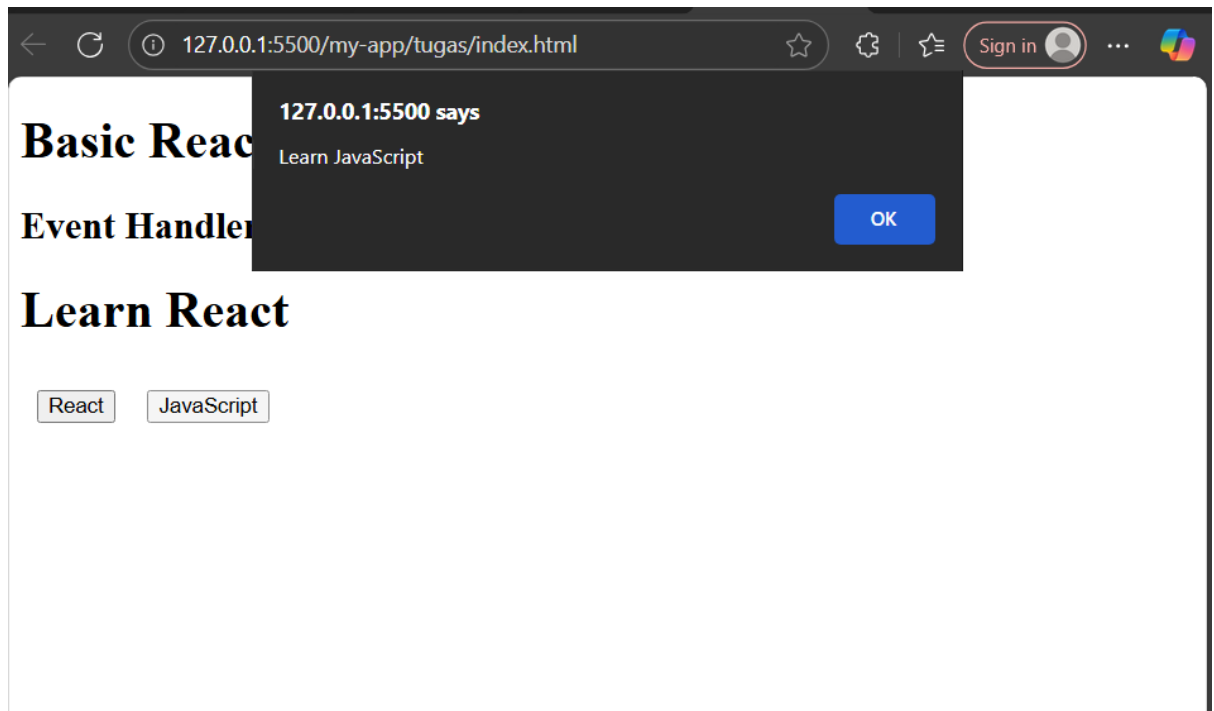
Event handler → menjadi arrow function dengan syntax `handleButtonClick = () => {}`
(tujuannya agar this menjadi instance dari class
Ada render untuk mengembalikan nilai jsx

Output htmlnya



⇒ Yang mana ketika kita tekan tombol react, dia akan mentrigger munculnya alert “learn react” jika tombol react ditekan nah jika javascript yang ditekan maka alert tampil “ learn javascript” yang akan tampil





- ⇒ Event handler sendiri berfungsi untuk menangani suatu event reaksi, contohnya seperti pada tombol, yang memiliki kondisi di setiap aksi buttonnya
- ⇒ Kode ini juga melibatkan props, yaitu proses memanfaatkan (mengirim / menginject/ transfer) data induk kepada data anak

2. Praktik seluruh poin praktikum

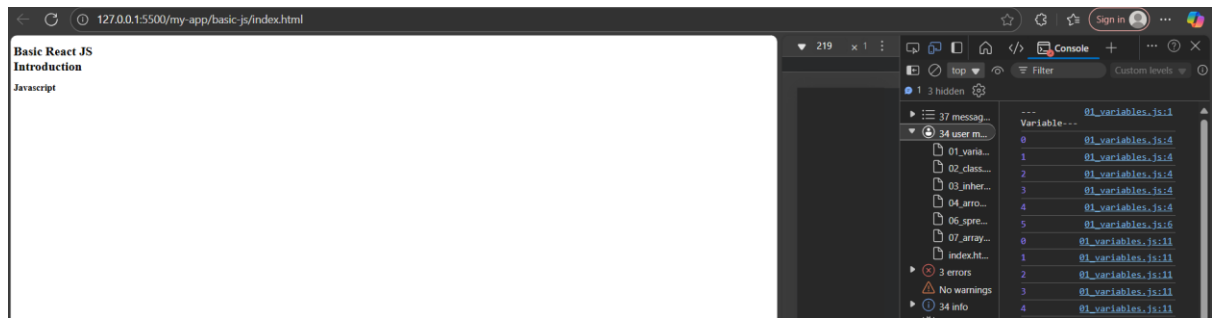
A. Basic Javascript

1. Variabel.js

```
1 console.log ("---Variable---");
2 // var
3 for (var i = 0 ; i<5 ; i++){
4     console.log(i);
5 }
6 console.log(i);
7
8 // var
9 function looping (){
10     for ( var j = 0 ; j < 5 ; j++){
11         console.log(j);
12     }
13 }
14 looping();
15 console.log(j);
16
17 // let
18
19 for (let k = 0 ; k < 5 ; k++){
20     console.log(k);
21 }
22 console.log (k);
23
24 // let
25 for (let l = 0; l <5 ; l++){
26     console.log(l);
27 }
28 if (true){
29     let l = 100;
30     console.log ("l =",l)
31 }
32
33 // const
34 const phi = 3.14;
35 phi = 3.147;
36 console.log(phi);
```

- ⇒ Console merupakan cara untuk menampilkan output dalam console yang didapat dari menu inspect suatu halaman
- ⇒ Yang pertama var, var di sini terdiri dari function scope, yang berarti ketika i berada di luar for perulangan, i masih bisa diakses
- ⇒ Perulangan kode ini mencakup perulangan untuk mencetak angka 0 sampai dengan 5

Output



Perbedaanya terlihat ketika i dicetak di dalam perulangan maupun di luar perulangan , menampilkan 0 sampai 5

Alur iterasi:

i start dari 0, apa benar $i < 5$ jika benar cetak 0, hingga sampai angka ke 5, apakah $i < 5$ jika iya

2. Var dalam fungsi

Memiliki fungsi looping, yang mana var start from 0 , kemudian dengan batas $j < 5$ dan terus melakukan increement , lakukan percetakan dalam console → yang muncul 0 1 2 3 4 dengan syarat panggil dulu fungsi looping, kemudian coba cetak log di luar function, hasilnya

```
0 01_variables.js:11
1 01_variables.js:11
2 01_variables.js:11
3 01_variables.js:11
4 01_variables.js:11
```

Hanya di line 11 yang terbaca (yaitu looping yang dipanggil dalam function) sedangkan yang langsung cetak j dan berada di luar perulangan , tidak mencetak apapun -> caranya harus deklarasikan var j menjadi global variabel atau paling aman, hapus console log yang di luar

```
Uncaught 01_variables.js:17
Reference
Error: j is not defined
at 01_variables.js:17:13
```

```
9 // Var
10 function looping() {
11     for (var j = 0; j < 5; j++) {
12         console.log(j);
13     }
14 }
15 looping();
16 // console.log(j); hapus karena kalau tidak dihapus, kode selanjutnya tidak bisa berjalan
17
```

undefined	01_variables.js:17
0	01_variables.js:21
1	01_variables.js:21
2	01_variables.js:21
3	01_variables.js:21
4	01_variables.js:21

Undefined line 17 dengan kondisi

```

9  // var
10 var j;
11 function looping() {
12     for (var j = 0; j < 5; j++) {
13         console.log(j);
14     }
15 }
16 looping();
17 console.log(j);

```

Karena berada di luar for, sedangkan j tidak diinisialisasi ketika deklarasi, ayo kita coba

```

9  // var
10 var j = 0;
11 function looping() {
12     for (var j = 0; j < 5; j++) {
13         console.log(j);
14     }
15 }
16 looping();
17 console.log(j);

```

0	01_variables.js:17
---	--------------------

Kondisi dihapus

0	01_variables.js:13
1	01_variables.js:13
2	01_variables.js:13
3	01_variables.js:13
4	01_variables.js:13

```

9    // var
10   // var j = 0 ;
11   function looping() {
12       for (var j = 0; j < 5; j++) {
13           console.log(j);
14       }
15   }
16   looping();
17   // console.log(j);
18

```

Hasil perulangan berhasil dicetak

3. let

let dengan variabel k yang batasnya juga sama yaitu kurang dari 5 dengan kondisi increement di setiap perulangannya, lalu mencetak di line 20 untuk hasil k, dan line 22 untuk hasil k di luar perulangan , outputnya sama sama tidak menampilkan apapun, karena let memiliki block scope , oleh karena itu cara memanggilnya dengan menghapus console.log yang berada di luar looping , agar kode selanjutnya bisa terbaca

```

0      01_variables.js:22
1      01_variables.js:22
2      01_variables.js:22
3      01_variables.js:22
4      01_variables.js:22
----- 01_variables.js:25
LET-----
0      01_variables.js:28
1      01_variables.js:28
2      01_variables.js:28
3      01_variables.js:28
4      01_variables.js:28
1 = 100 01_variables.js:32

```

Sebelumnya line 22 sampai 32 tidak menampilkan apapun

pada kode kedua let terletak dalam blok yang berbeda
let l start dari 0 , dengan batas < 5 dan increement pada l, lalu mencetak di dalam
for, buat kondisi jika benar, maka l = 100, lalu lakukan percetakan l = l

```
----- 01_variables.js:25  
LET-----  
0      01_variables.js:28  
1      01_variables.js:28  
2      01_variables.js:28  
3      01_variables.js:28  
4      01_variables.js:28  
l = 100 01_variables.js:32
```

Nah jika dalam kondisi perulangan benar, maka print l = sesuai yang dideklarasikan
yaitu 100

4. const

```
Uncaught 01_variables.js:38  
t  
TypeError: Assignment to  
constant variable.  
    at 01_variables.js:38:5
```

Yang artinya const tidak bisa diubah setelah dideklarasikan, maka ganti kode
menjadi

```
// Const  
const phi = 3.14;  
// phi = 3.147;  
console.log(phi);  
  
3.14 01_variables.js:39
```

Dan terbaca phi awal yang dideklarasikan adalah 3,14

b. Class.js

```
Release Notes: 1.104.2 JS 02_class.js U X
my-app > basic-js > js > JS 02_class.js > ...
1 console.log ("class")
2
3 class Car {
4   constructor(nama){
5     this.name = nama;
6   }
7   present (){
8     return "nama mobilku adalah "+this.name;
9   }
10 }
11 let myCar = new Car ("carr-i");
12 console.log (myCar.present());
```

- ⇒ class. Js merupakan penggunaan class di java script (OOP) , yang mana memiliki class car, lalu method constructor (yang dijalankan setiap pembuatan objek baru), kemudian menyimpan nama (untuk mengisi nama mobil ketika dipanggil)
- ⇒ jadi ketika car dipanggil, ia tidak statis tetapi dinamis (setiap New Car memiliki nama masing masing
- ⇒ lalu cetak di myCar.present → mycar variabel yang menampung nilai new Car

```
class          02_class.js:1
nama mobilku   02_class.js:12
adalah carr-i
```

c. inheritance

```
Release Notes: 1.104.2 JS 03_inheritance.js U X
my-app > basic-js > js > JS 03_inheritance.js > ...
1
2
3 console.log("----- Inheritance -----");
4
5 class Gadget {
6   constructor(name) {
7     this.name = name;
8   }
9   present() {
10    return "I have an " + this.name;
11  }
12 }
13
14 class Handphone extends Gadget {
15   constructor(name, mod) {
16     super(name);
17     this.model = mod;
18   }
19
20   show() {
21     return this.present() + ", it is " + this.model;
22   }
23 }
24
25 let handphone = new Handphone("iphone", "14 Pro Max");
26
27 console.log(handphone.show());
28
```

- ⇒ inheritance merupakan pewarisan, terdiri dari class induk classGadget (yang bisa diwariskan dari this.name dan present fungsi yang menampung nilainya
- ⇒ jika ada let coba = new Gadget ("ipad") maka ketika console.log (coba.present()); hasilnya i have an ipad
- ⇒ kemudian ada class handphone extend gadget, artinya handphone mewarisi gadget, maka handphone memiliki properti dan method yang sama dari gadget, fungsi super untuk memanggil cosntructor dari parent (gadget) , karena gadget butuh parameter name , kita kirim lewat super
- ⇒ tanpa super, javascript akan error karena wajibnya kelas warisan untuk memanggil parent (this)
- ⇒ walaupun ia mewarisi gadget, ia juga bisa menambah properti baru (yaitu mod)
- ⇒ function show untuk mengembalikan method parent present ditambah this.model
- ⇒ maka outputnya i have an iphone , it is 14 pro max
- ⇒ iphone sebagai name handphone, dan 14 promax adalah modelnya

```

----- 03_inheritance.js:3
Inheritance -----
I have 03_inheritance.js:27
an iphone, it is 14 Pro Max

```

d. arrow function

```

my-app > basic-js > js > JS 04_arrow_function.js > ...
1  console.log("----- Arrow Function -----");
2
3  // Normal Function
4  function hello() {
5      return "Hello World!";
6  }
7
8  // Normal function
9  hello = function() {
10     return "Hello World!";
11 }
12
13 console.log(hello());
14
15 // Arrow Function
16 hello = () => {
17     return "Hello World! This is from Arrow Function";
18 }
19
20 // Arrow Function
21 hello = () => "Hello World! This is from Arrow Function";
22
23 console.log(hello());
24
25 // Arrow Function with 1 parameter
26 hello = (myName) => "Hello " + myName + "! This is from Arrow Function";
27 // Arrow Function with 1 parameter
28 hello = myName => `Hello ${myName}! This is from Arrow Function`;
29
30 console.log(hello("Purnama"));
31

```

- ⇒ function hello merupakan fungsi normal yang mengembalikan hello world (basic) dan bisa dipanggil di luar console.log(hello());
 - ⇒ selanjutnya hello = function () {return..} merupakan fungsi ekspresi, yaitu menyimpan fungsi di dalam variabel hello
 - ⇒ arrow function (cara ringkas dengan syntax nama fungsi = () => {return } identik dengan arrow yang artinya tanda panah
 - ⇒ arrow function juga bisa disederhanakan dengan hello = () => tanpa return dan langsung text dalam kondisi return hanya 1
 - ⇒ arrow function 1 param, hello = (myName) => "hello" + myName + "this is arrow function" → boleh disingkat karena param hanya myName , cara manggilnya console.log (hello("purnama"));
- ini outputnya

```

Found)

----- 04_arrow_function.js:1
Arrow Function -----

Hello 04_arrow_function.js:13
World!

Hello 04_arrow_function.js:23
World! This is from Arrow
Function

Hello 04_arrow_function.js:30
Purnama! This is from Arrow
Function

```

Yang pertama fungsi biasa (hello world) yang kedua fungsi hello dalam arrow function, dan arrow function yang memiliki parameter

e. destructuring

```

Release Notes: 1.104.2  JS 03_inheritance.js U  JS 05_destructuring.js U X
my-app > basic-js > js > JS 05_destructuring.js > ...
1 console.log("----- Destructuring -----");
2
3 // Array
4 let items = ["Table", "Handphone", "Computer"];
5
6 // Manual variable declaration
7 let item1 = items[0];
8 let item2 = items[1];
9 let item3 = items[2];
10
11 console.log(item1);
12 console.log(item2);
13 console.log(item3);
14
15 // Variable declaration with destructuring
16 let [item_1, item_2, item_3] = items
17
18 console.log(item_1);
19 console.log(item_2);
20 console.log(item_3);
21
22 // Object
23 let student = {
24   name: "Ahmad",
25   age: 22,
26   department: "Information System"
27 }
28
29 // Manual variable declaration
30 let studentName = student.name;
31 let studentAge = student.age;
32 let studentDepartment = student.department;
33
34 console.log(studentName);
35 console.log(studentAge);
36 console.log(studentDepartment);
37

```

```

39  // With Spread Operator
40  let employee2 = {
41      ...employee,
42      age: 20,
43      city: "Surabaya"
44  };
45  console.log(employee2);
46

```

- ⇒ destructuring array
- ⇒ array awal (table, handphone, computer)
- ⇒ cara manual satu satu pemanggilan let item 1 = items [0] sampe item 3 = items[2]
- ⇒ cara pakai destructuring let [item_1,item_2,item_3] = items;
- ⇒ hasilnya sama persis

```

----- 05_desctructuring.js:1
Destructuring -----
Table 05_desctructuring.js:11
Handp 05_desctructuring.js:12
hone
Compu 05_desctructuring.js:13
ter
Table 05_desctructuring.js:18
Handp 05_desctructuring.js:19
hone
Compu 05_desctructuring.js:20
ter

```

Destructuring proses mengubah struktur array yang awalnya berisi item item , menjadi satu kesatuan wadah

➔ destructuring object , dengan cara object yang atributnya di panggil satu satu semisal ada let student berisi atribut name age dan department, maka ada 3 variabel baru dengan student.atribut pada suatu object tersebut

➔ lalu destructuring object dengan let {atribut1,atribut2,atribut3} = nama object
Maka kita bisa akses seluruh nilai dari object tanpa harus satu satu pemanggilannya

f. spread_operator

```
1 console.log("----- Spread Operator -----");
2
3 // Array
4 let arr1 = [10, 20, 30];
5 let arr2 = [100, 200, 300];
6
7 // Without Spread Operator
8 let arr3 = [arr1, arr2];
9 console.log(arr3);
10
11 // With Spread Operator
12 let arr4 = [...arr1, ...arr2];
13 console.log(arr4);
14
15 // With Spread Operator
16 let arr5 = [...arr1, 40, 50];
17 console.log(arr5);
18
19 // With Spread Operator
20 let arr6 = [30, 40, 50, ...arr1];
21
22 console.log(arr6);
23
24 // Object
25 let employee = {
26     name: "Budi",
27     age: 22,
28     position: "Software Engineer"
29 };
30
31 // Without Spread Operator
32 let employee1 = {
33     employee,
34     age: 25,
35     city: "Jakarta"
36 };
37 console.log(employee1);
38
39 // With Spread Operator
40 let employee2 = {
41     ...employee,
42     age: 20,
43     city: "Surabaya"
44 };
45 console.log(employee2);
46
```

- ⇒ spread operator
- ⇒ jika ada arr 1 dan arr 2 dengan nilai berbeda seperti arr1 dari range 10 – 30, dan arr 2 100 – 300
- ⇒ nah jika tanpa spread langsung saja simpan arr 1 dan arr 2 dalam satu variabel, misal arr3
- ⇒ jika dengan spread operator , tetap dengan menyimpan dalam satu variabel di mana [...arr1,...arr2] titik titik artinya mengakses nilai array sehingga dimasukkan satu persatu dalam variabel tersebut
- ⇒ jika ingin menambah 40, 50 dalam arr 1 misalnya maka let arr5 = [...arr1,40,50]
- ⇒ jika spread di awal, maka [30,40,50,...arr1], spread operator bisa digunakan di awal,tengah, maupun akhir
- ⇒ spread operator pada object satu variabel dibungkus kurung kurawal dengan beberapa atribut, seperti name, age, dan position
- ⇒ jika tanpa spread, maka tulis satu satu kodenya , dengan menyimpan satu variabel penampung, dan di dalam kurung kurawal memanggil object tadi lalu menambahnya dengan atribut lain seperti age, dan city, dalam hal ini age akan ter override menjadi 25
- ⇒ jika dengan spread maka penggunaan titik titik sebelum pemanggilan object

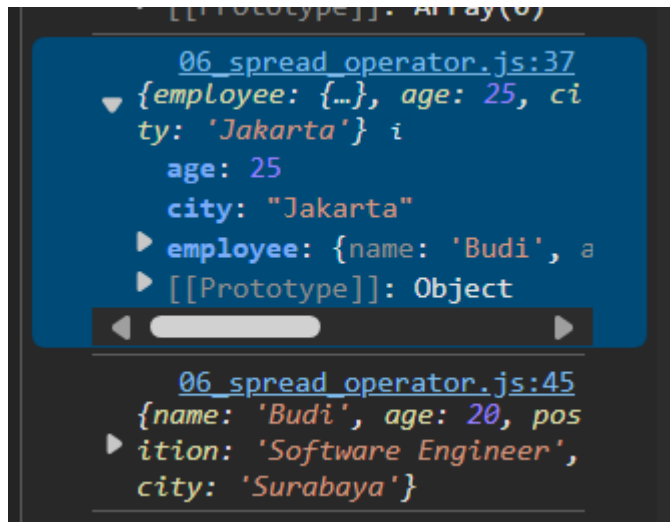
----- 06_spread_operator.js:1
- Spread Operator -----

06_spread_operator.js:9
▶ (2) [Array(3), Array(3)]

06_spread_operator.js:13
▼ (6) [10, 20, 30, 100, 200, 300] ⓘ
0: 10
1: 20
2: 30
3: 100
4: 200
5: 300
length: 6
▶ [[Prototype]]: Array(0)

06_spread_operator.js:17
▼ (5) [10, 20, 30, 40, 50] ⓘ
0: 10
1: 20
2: 30
3: 40
4: 50
length: 5
▶ [[Prototype]]: Array(0)

06_spread_operator.js:22
▼ (6) [30, 40, 50, 10, 20, 30] ⓘ
0: 30
1: 40
2: 50
3: 10
4: 20
5: 30
length: 6
▶ [[Prototype]]: Array(0)



f. array method

```
JS 07_array_method.js U X
my-app > basic-js > js > JS 07_array_method.js > ...
1 console.log("----- Array Method -----");
2
3 // forEach
4 let arr = [10, 20, 30, 40];
5 arr.forEach(function(val, key) {
6   console.log(`Array value in index-${key} is ${val}`);
7 });
8
9 // Map
10 let arrSquare = arr.map((val) => val * val);
11 console.log(arrSquare);
```

- ⇒ metode untuk mengolah array ada 2 : forEach dan Map
- ⇒ forEach merupakan perilaku iterasi pada sebuah array, ia akan mencetak nilai arr dari indeks 0 dan tidak akan menghasilkan array baru
- ⇒ metode map, membuat array baru berdasarkan hasil operasi setiap elemen array lama
- ⇒ val diambil dari arr lalu dikalikan dengan dirinya sendiri, lalu ditampung dalam variabel arrSquare untuk menyimpan nilainya

```
----- 07_array_method.js:1
Array Method -----

Array 07_array_method.js:6
value in index-0 is 10

Array 07_array_method.js:6
value in index-1 is 20

Array 07_array_method.js:6
value in index-2 is 30

Array 07_array_method.js:6
value in index-3 is 40

07_array_method.js:11
▶ (4) [100, 400, 900, 1600]
```

B. Basic React

1. Jsx

```

1
2 // Hello
3 let myElement = "Hello World";
4
5
6 // Basic Rules
7 // myElement = "<h3>Hello React</h3>"; // wrong
8 myElement = <h3>Hello React</h3>; // correct
9
10
11 // Basic Rules
12 // myElement = ; // wrong
13 myElement = ; // correct
14
15
16 // Basic Rules
17 // myElement = (
18 //   <p>Learn Javascript</p>
19 //   <p>Learn React</p>
20 // ); // wrong
21 myElement = (
22   <div>
23     <p>Learn Javascript</p>
24     <p>Learn React</p>
25   </div>
26 ); // correct
27
28
29 // Statement in JSX
30 let myText = "Learn React";
31 myElement = (
32   <div>
33     <h3>{myText}</h3>
34   </div>
35 );
36
37
38 // Array Processing
39 let items = ["Table", "Handphone", "Computer"];
40 myElement = (
41   <ul>
42     {items.map((item, index) => <li>{item}</li>)}
43   </ul>
44 );
45
46 const myApp = myElement;
47
48 ReactDOM.createRoot(document.getElementById('root')).render(myApp);
49
50

```

⇒ Kode di atas merupakan kode dasar JSX (javascript xml) di react, yang menunjukkan dasar penulisan elemen

- ⇒ Let myElement pada deklarasi awal hanya menjadi string biasa, belum merupakan elemen react yang bisa dipanggil
- ⇒ myElement = “<h3> haloooo</h3>” tentu salah karena membungkus elemen h3 dengan tanda kutip, sistem akan mengenali sebagai string
- ⇒ yang benar hilangkan tanda kutip yang membungkus elemen h3
- ⇒ myElement = ; penulisan seperti ini boleh dalam html tapi jsx harus selalu ditutup maka yang benar myElement = ;
- ⇒ root element hanya boleh satu , jsx tidak boleh punya elemen root seperti elemen p yang ada 2 (kode atas)
- ⇒ yang benar bungkus kedua elemen p dengan div sebagai elemen induk
- ⇒ variabel dalam jsx identik dengan kurung kurawal yang bisa berisi variabel, operasi atau bahkan fungsi
- ⇒ mengolah array di jsx dengan mengakses seluruh nilai pada items, dengan items.map untuk looping setiap elemen array yang akan diubah menjadi elem li
- ⇒ render halaman, createRoot membuat akar React di elemen HTML dengan ide root dan render myApp yang membaca seluruh isi jsx keseluruhan

Basic ReactJS Introduction

JSX (Javascript XML)

- Table
- Handphone
- Computer

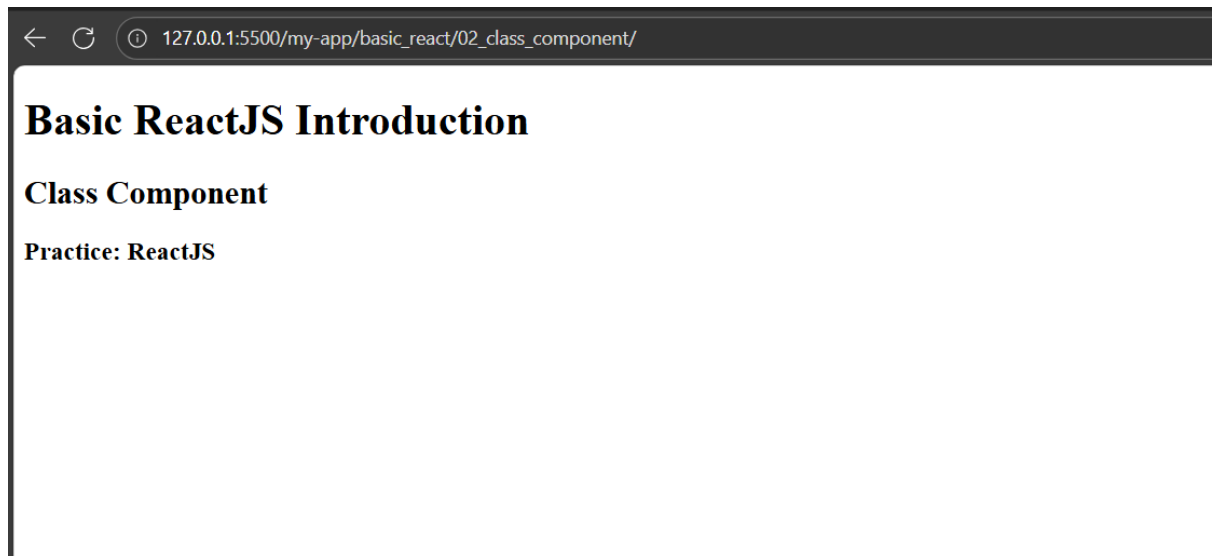
2. class_component

```

1
2 // Class Component
3 class Practice1 extends React.Component {
4   render() {
5     return <h3>Practice: ReactJS</h3>
6   }
7 }
8 let myElement = <Practice1 />;
9
10
11 // Props in Class Component
12 class Practice2 extends React.Component {
13   render() {
14     return <h3>Practice: {this.props.topic}</h3>
15   }
16 }
17 myElement = <Practice2 topic="ReactJS"/>;
18
19
20 // Props Children in Class Component
21 class Practice3 extends React.Component {
22   render() {
23     return <h3 id={this.props.module}>Practice: {this.props.children}</h3>
24   }
25 }
26 myElement = <Practice3 module="01">ReactJS</Practice3>;
27
28 const myApp = myElement;
29
30 ReactDOM.createRoot(document.getElementById('root')).render(myApp);
31
32

```

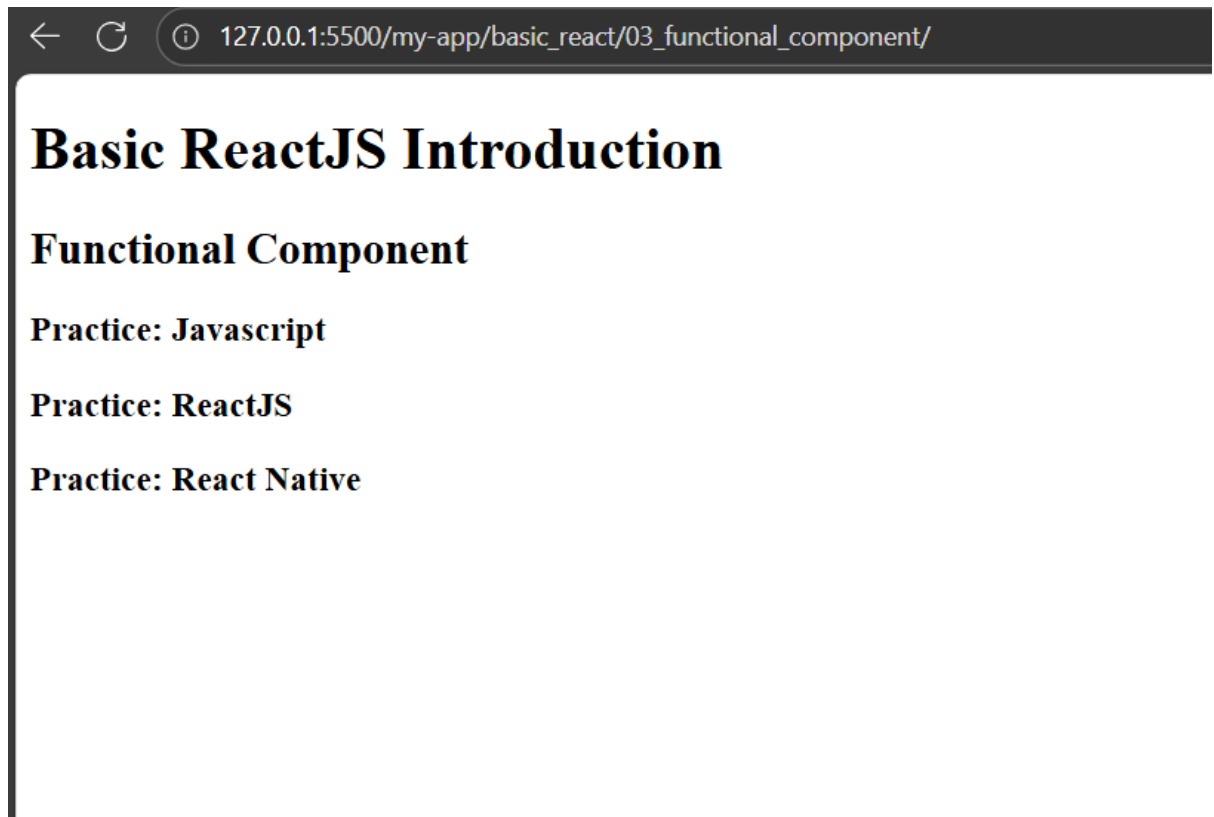
- ⇒ class practice1 membuat komponen react berbasis class
- ⇒ harus extend react.component agar memiliki kemampuan mengenali react
- ⇒ dan wajib memiliki render untuk menampilkan hasil dari elemen jsx yang ditampilkan
- ⇒ dan dipanggil dengan tag jsx <practice />
- ⇒ class practice2 merupakan contoh props, yang mana parameter atau data yang memanfaatkan kelas induk dan kelas anak, di sini topic adalah prop yang dikirim dari practice2 topic
- ⇒ class practice3 juga memanfaatkan props.children yang mengambil isi dalam tag komponen, this props module yaitu mengambil prop module 1, dan this props children mengambil isi yang dibungkus oleh tag pembuka dan penutup
- ⇒ render ke dom, yaitu membuat elemen react dapat diakses, dan menampilkan elemen react di variabel myElement



3. functional_component

```
Code Snapshot JS functional_component.js U X Code Snapshot
my-app > basic_react > 03_functional_component > JS functional_component.js > ...
1 // Functional Component
2 const ReactTitle = () => {
3   return <h3>Currently learning ReactJS</h3>
4 }
5 const JavascriptTitle = () => {
6   return <h3>Currently learning Javascript</h3>
7 }
8 const MyComponent = () => {
9   return (
10     <div>
11       <ReactTitle />
12       <hr />
13       <JavascriptTitle/>
14     </div>
15   );
16 }
17 let myElement = <MyComponent />
18 // Props in Functional Component
19 const Practice = (props) => <h3>Practice: {props.course}</h3>;
20 myElement = (
21   <div>
22     <Practice course="Javascript"/>
23     <Practice course="ReactJS"/>
24     <Practice course="React Native"/>
25   </div>
26 );
27
28 const myApp = myElement;
29
30 ReactDOM.createRoot(document.getElementById('root')).render(myApp);
```

- ⇒ komponen react yang ditulis dalam function, khususnya arrow function
- ⇒ yang namanya fungsi harus mengembalikan nilai jsx
- ⇒ nama fungsi harus huruf besar di awal bukan gaya Camel agar dikenali sebagai komponen
- ⇒ komponen myComponent berisi dua komponen lain (ReactTitle dan Javascript Title) , hari adalah pembatas antar teks, dan myElement berisi element jsx `<MyComponent />` yang siap dirender
- ⇒ props di functional comp, diakses dengan props.namaProp tidak perlu this
- ⇒ props dipanggil dengan mendefinisikan induk sebanyak prop berbeda, karena dengan prop sebuah komponen dapat memiliki sifat yang berbeda beda
- ⇒ jangan lupa untuk render ke halaman dengan createRoot, dan menyimpan myApp ke myElement
- ⇒ Perbandingan dengan class:
 - a. Syntax, syntax class → `class Nama extends React.Component { render(){...} }` dan functional syntax → `const Nama = () => {...`
Functional merupakan arrow function fungsi yang identik dengan tanda panah, dan class diawali dengan syntax class dengan mewarisi React.Component untuk mengenalinya sebagai komponen
 - b. Penggunaan props pada class menggunakan `this.props.namaProp` sedangkan functional hanya `props.namaProp`
 - c. lifeCycle class component pakai `componentDidMount`, dan lain lain sedangkan functional menggunakan hooks seperti `useEffect`, dan lain lain
 - d. penulisan class component lebih panjang, sedangkan functional lebih ringkas dan modern dibandingkan class component

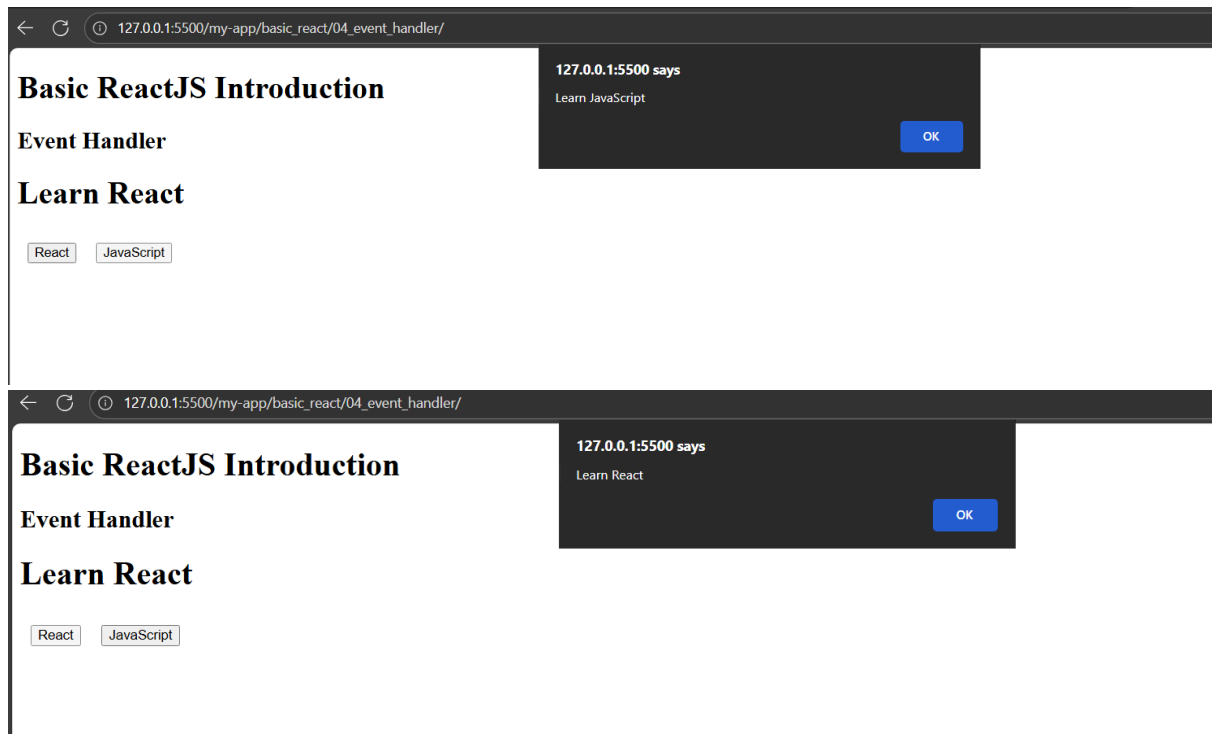


Terlihat practice yang memiliki sifat berbeda

4. event handler

```
Code Snapshot JS event_handler.js U x Code Snapshot
my-app > basic_react > 04_event_handler > JS event_handler.js > ...
1
2 // Button Component
3 const MyButton = (props) => {
4   const handleClick = () => {
5     alert(props.message);
6   }
7
8   return (
9     <button onClick={handleButtonClick} style={{ margin: "10px" }}>
10       {props.children}
11     </button>
12   );
13 }
14
15
16 // App Component
17 const MyApp = () => {
18   return (
19     <div>
20       <h1>Learn React</h1>
21       <MyButton message="Learn React">React</MyButton>
22       <MyButton message="Learn JavaScript">JavaScript</MyButton>
23     </div>
24   );
25 }
26
27 const myElement = <MyApp />;
28 const myApp = myElement;
29
30 ReactDOM.createRoot(document.getElementById('root')).render(myApp);
31
32
```

- ⇒ merupakan contoh functional component yang memanfaatkan props dan event handler
- ⇒ props menampung data yang dikirim dari MyApp
- ⇒ dan handleClick adalah event handler untuk klik tombol
- ⇒ saat tombol diklik, akan mentrigger alert berisi props.message dan props.children menampilkan isi teks di antara tag MyButton
- ⇒ komponen MyApp adalah dua tombol yang menampilkan pesan berbeda (tiap mybutton diklik , message dikirim lewat props
- ⇒ jangan lupa render ke DOM



Terlihat setiap tombol memiliki sifat berbeda yang dikirim props message

5. state

```
Code Snapshot JS state.js U X Code Snapshot
my-app > basic_react > 05_state > JS state.js > Cat > render
1
2 // Cat Component
3 class Cat extends React.Component {
4   // State Declaration
5   state = {
6     isHungry: true
7   };
8
9   render() {
10    return (
11      <div>
12        <p>I am {this.props.name}, and I am {this.state.isHungry ? " hungry" : " full"}
13        <button
14          onClick={() => {
15            this.setState({ isHungry: false });
16          }}
17          disabled={!this.state.isHungry}
18        >
19        {this.state.isHungry ? "Pour me some milk, please!" : "Thank you!"}
20      </button>
21    </div>
22  );
23 }
24 };
25
26
```

- ⇒ state dalam contoh ini berbasis class component
- ⇒ komponen Cat merupakan class component yang mewatisi React.component
- ⇒ state dideklarasikan sebagai objek internal dengan isHungry = true menandakan kucing sedang lapar, dan state bisa diubah dengan this.setState()
- ⇒ render method merupakan wajib method di class component dan mengembalikan JSX dan akan dirender ke DOM
- ⇒ this.props.name mengambil prop yang diberikan dari parent component <Cat name = "TOM" /> dan this.state.isHungry menggunakan ternary operator (if else yang disederhanakan) untuk menampilkan kucing lapar atau tidak
- ⇒ button dengan onclick, dengan kondisi this.setState is hungry false dijalankan, maka setState memperbarui kalau kucing sudah tidak lapar → nilai akan dirender dalam tampilan
- ⇒ dan akan disabled jika isHungry false, (mencegah tombol agar tidak diklik lagi setelah kucing kenyang)
- ⇒ kalau isHungry true maka text menampilkan pourme some milk please dan sebaliknya text thankyou
- ⇒ cafe component merupakan parent component

- ⇒ di sini cafe memiliki dua kucing bernama munkustrap dan spot yang memiliki state sendiri . jadi ketika munkustrap diberi makan dan kenyang tidak pengaruh untuk Spot
- ⇒ jangan lupa render Cafe dan Cat ke dalam DOM

Basic ReactJS Introduction

State

I am Munkustrap, and I am full!

Thank you!

I am Spot, and I am hungry!

Pour me some milk, please!

- C. Counter
 - a. Class component

```

1 class PenghitungClass extends React.Component {
2   constructor(props) {
3     super(props);
4     this.state = { aktif: false, hitung: 0 };
5   }
6
7   handleTambah = () => {
8     this.setState({ aktif: true, hitung: this.state.hitung + 1 });
9   };
10
11   handleKurang = () => {
12     this.setState({ aktif: false, hitung: this.state.hitung - 1 });
13   };
14
15   render() {
16     return (
17       <div className="paper-container">
18         <div className="paper circle">
19           <div className="content">
20             <h2>Ayo Menghitung di Lingkaran Kecil</h2>
21             <h1>{this.state.hitung}</h1>
22             <div>
23               <button onClick={this.handleKurang}>-1</button>
24               <button onClick={this.handleTambah}>+1</button>
25             </div>
26           </div>
27         </div>
28       </div>
29     );
30   }
31 }
32

```

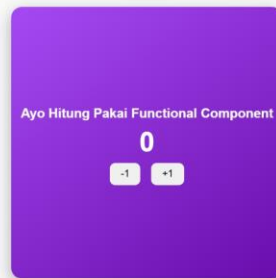
- ⇒ State disimpan this.state (aktif dan hitung), aktif menandakan tombol terakhir yang ditekan ntah true sebagai increment (+1) dan decreement (-1) , hitung adalah nilai counter → state disimpan dalam super props.
- ⇒ Methodnya handleTambah dan handleKurang yaitu memiliki nilai aktif true/false dengan this state hitung decreement/increment
- ⇒ Perubahan state menggunakan this set state yang otomatis re render

b. Functional component

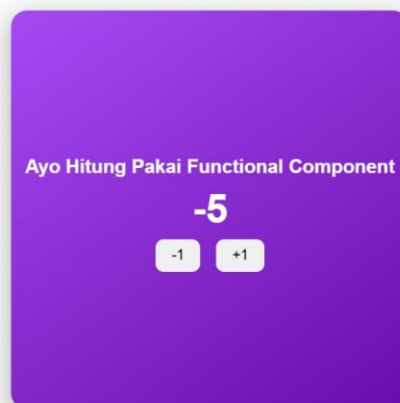
```
praktikum
Code Snapshot JS state.js U JS counter.js U X Code Snapshot
my-app > tugas > JS counter.js > PenghitungClass
32
33 function PenghitungFunctional() {
34   const [hitung, setHitung] = React.useState(0);
35
36   const tambah = () => setHitung(hitung + 1);
37   const kurang = () => setHitung(hitung - 1);
38
39   return (
40     <div className="paper-container">
41       <div className="paper">
42         <div className="content">
43           <h2>Ayo Hitung Pakai Functional Component</h2>
44           <h1>{hitung}</h1>
45           <div>
46             <button onClick={kurang}>-1</button>
47             <button onClick={tambah}>+1</button>
48           </div>
49         </div>
50       </div>
51     </div>
52   );
53 }
54
55 const App = () => (
56   <div>
57     <PenghitungClass />
58     <PenghitungFunctional />
59   </div>
60 );
61
62 const root = ReactDOM.createRoot(document.getElementById("root"));
63 root.render(<App />);
64
```

- ⇒ Statinya menggunakan hook (hitung, setHitung) lalu panggil react.useState start dari 0
- ⇒ Event handler berbentuk komponen tambah dan kurang, tambah increment dan kurang untuk decrement, pemanggilannya tidak perlu this, karena functional. setHitung (hitung-1) untuk decrement dan hitung+1 untuk increment
- ⇒ Parent component app, menampung 2 komponen (class dan komponen) untuk dirender ke dalam DOM

Counter ReactJS - class and func



Counter ReactJS - class and func



D. Tambahan (instalansi + cara buat project expo)


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.194]
(c) Microsoft Corporation. All rights reserved.

D:\SEM 7\PAB\praktikum>code .

D:\SEM 7\PAB\praktikum>node -v
v20.13.1

D:\SEM 7\PAB\praktikum>git --version
git version 2.44.0.windows.1

D:\SEM 7\PAB\praktikum>
```

Environment yang dibutuhkan adalah node dan git, node js sendiri adalah runtime javascript di sisi server yang memungkinkan menjalankan JS di luar browser, node js juga terdiri dari beberapa package seperti react/react-native, dan lain lain selain itu index untuk merender menggunakan transpile js berupa babel.

Sedangkan git merupakan version control yang mencatat setiap perubahan kode, memudahkan kolaborasi tim dan rollback apabila ada kesalahan

```
D:\SEM 7\PAB\praktikum>npx create-expo-app my-app
Creating an Expo project using the default template.

To choose from all available templates (https://github.com/expo/expo/tree/main/templates) pass in the --template arg:
$ npx create-expo-app --template


To choose from all available examples (https://github.com/expo/examples) pass in the --example arg:
$ npx create-expo-app --example
```

Command untuk create project expo , setelah itu muncul qr code

```
C:\Windows\system32\cmd.exe

D:\SEM 7\PAB\praktikum>cd my-app

D:\SEM 7\PAB\praktikum\my-app>npx expo start
Starting project at D:\SEM 7\PAB\praktikum\my-app
React Compiler enabled
Starting Metro Bundler



> Metro waiting on exp://192.168.160.195:8081
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)
> Web is waiting on http://localhost:8081
> Using Expo Go
> Press s | switch to development build
```

Discan dan distart di expo go hp android, kemudian membuat folder

```
basic-js
├── js
│   ├── 01_variables.js
│   ├── 02_class.js
│   ├── 03_inheritance.js
│   ├── 04_arrow_function.js
│   ├── 05_descstructuring.js
│   ├── 06_spread_operator.js
│   └── 07_array_method.js
└── index.html
```

```
basic_react
├── 01_jsx
│   ├── index.html
│   └── jsx.js
├── 02_class_component
│   ├── class_component.js
│   └── index.html
├── 03_functional_component
│   ├── functional_component.js
│   └── index.html
├── 04_event_handler
│   ├── event_handler.js
│   └── index.html
├── 05_state
│   ├── index.html
│   └── state.js
```

▼ tugas	●
JS classcomp_eventhandler.js	U
JS counter.js	U
<> index_count.html	U
<> index.html	U