

# React JS

React Introduction,  
Components, State, Hooks,  
Use Effects



# React JS

- ☐ React Introduction
- ☐ Installation
- ☐ Component & Props
- ☐ State
- ☐ Hooks
- ☐ UseEffect

# Objektif sesi

- Memahami React JS sebagai pustaka javascript untuk membangun antar muka
- Memahami mengapa React JS dapat membantu kita dalam membangun antar muka di web
- Dapat menggunakan React JS dan JSX
- Memahami konsep dasar React JS
  - Component
  - State
  - Hooks

# React JS

- ☒ React Introduction
- ☐ Installation
- ☐ Component & Props
- ☐ State
- ☐ Hooks
- ☐ UseEffect



# Apa itu ReactJS?

A JavaScript library for  
building user interfaces



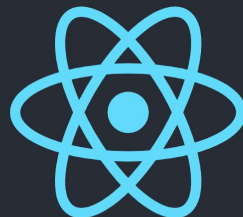
# React

A JavaScript library for building user interfaces

[Get Started](#)[Take the Tutorial >](#)[Follow @reactnative](#)[Star](#)

# React Native

Learn once, write anywhere.

[Get started](#)[Learn the basics >](#)

The background of the slide is a faded, grayscale aerial photograph of a dense urban skyline with numerous skyscrapers and buildings. A teal vertical bar is located on the far left side of the image.

**Apa keuntungannya?**

# Imperative Vs Declarative

## IMPERATIVE

```
// Imperative Programming
let array = [1, 2, 3, 4, 5, 6]
var evenNumbers: [Int] = []
for i in 0..array.count {
  if array[i] % 2 == 0 {
    evenNumbers.append(array[i])
  }
}
```

VS

## DECLARATIVE

```
// Declarative
let evenNumbers2 = array.filter { $0 % 2 == 0 }
```

## Imperative

```
let arr = [1, 2, 3, 4, 5],
    arr2 = [];

for (var i=0; i<arr.length; i++) {
  arr2[i] = arr[i]*2;
}

console.log(arr2);
```

## Declarative

```
let arr = [1, 2, 3, 4, 5];

arr2 = arr.map(function(v, i) {
  return v * 2;
});

console.log(arr2);
```





```
<html>
  <body>
    <div>Hello World</div>
    <script type="module">
      // your JavaScript here
    </script>
  </body>
</html>
```



```
<body>
  <div id="root"></div>
  <script type="module">
    const rootElement = document.getElementById('root')
    const element = document.createElement('div')
    element.textContent = 'Hello World'
    element.className = 'container'
    rootElement.append(element)
  </script>
</body>
```

|



```
<body>
  <div id="root"></div>
  <script src="https://unpkg.com/react@18.1.0/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@18.1.0/umd/react-dom.development.js"></script>
  <script type="module">
    const rootElement = document.getElementById('root')
    const element = React.createElement('div', {
      className: 'container',
      children: 'Hello World',
    })
    ReactDOM.createRoot(rootElement).render(element)
  </script>
</body>|
```



```
function App() {  
  return (  
    <div>  
      <h1>Hello World</h1>  
    </div>  
  );  
}
```


Code diatas ditulis dalam file dengan format javascript, yang berarti kita dapat langsung berinteraksi dengan html langsung di dalam javascript! Thanks to BABEL

# React JS

- ☒ React Introduction
- ☐ Installation
- ☐ Component & Props
- ☐ State
- ☐ Hooks
- ☐ UseEffect



# React JS Installation



```
<script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>  
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" crossorigin></script>
```

Diatas adalah snippet code untuk development,  
Namun kita juga bisa menggunakan production version

# JSX Installation ( Babel )



```
<script src="https://unpkg.com/@babel/standalone@7.12.4/babel.js"></script>
```

# Extensions Tambahan

[Home](#) > [Extensions](#) > React Developer Tools



## React Developer Tools

Add to Chrome

 Featured

★★★★★ 1,416 ⓘ | [Developer Tools](#) | 4,000,000+ users

[Overview](#)

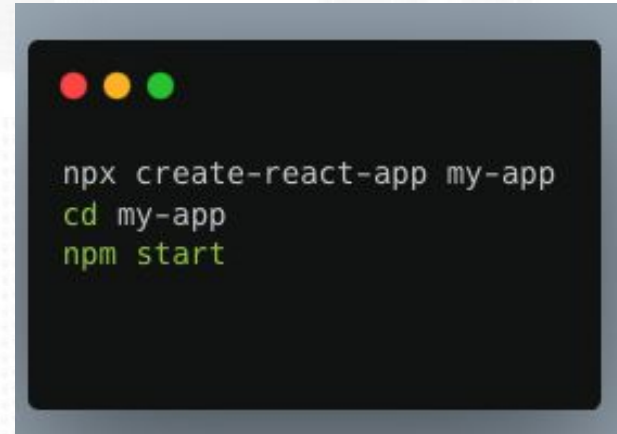
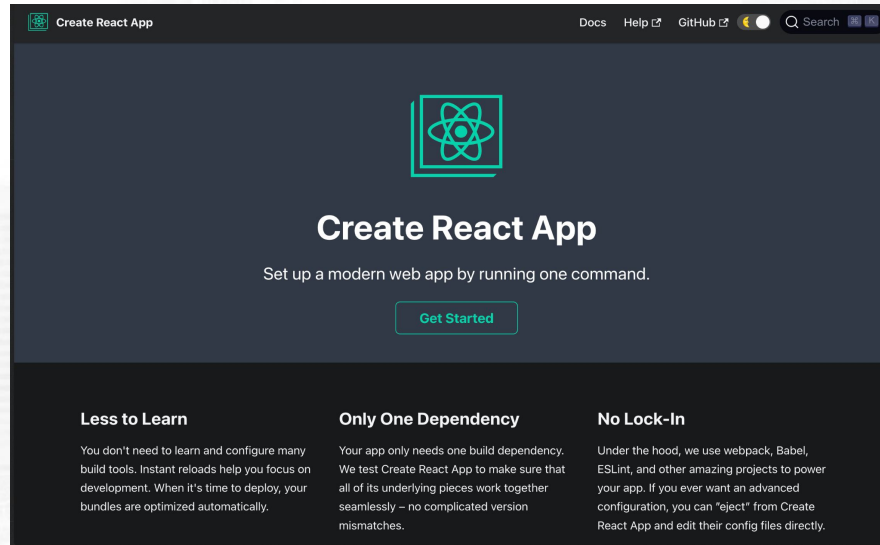
[Privacy practices](#)

[Reviews](#)

[Support](#)

[Related](#)

# A boilerplate / Framework to start with React



Dengan menggunakan ini, secara otomatis kita akan menginstall React ( development / production ) dan Babel. Selain itu sudah ada folder structure sederhana yang bisa kita kembangkan.

Penggunaan CRA ( Create react app ) juga bisa kita custom kalau ingin dengan package bernama CRACO ( Create React App Configuration Override )

# React JS

- ☒ React Introduction
- ☒ Installation
- ☒ Component & Props
- ☐ Hooks
- ☐ State
- ☐ UseEffect



## Component dan Props

Sama seperti javascript pada umumnya, kita pasti ingin menggunakan ulang code yang sudah kita tulis dengan function.

Dalam React ( JSX ) kita juga dapat menggunakan konsep yang sama, disebut dengan **COMPONENT**

## Example



```
<div class="container">
  <div class="message">Hello World</div>
  <div class="message">Goodbye World</div>
</div>
```

Props



```
<Message greeting="Hello" subject="World" />
<Message greeting="Goodbye" subject="World" />
```

```
function message({children}) {
  return <div className="message">{children}</div>
}

const element = (
  <div className="container">
    {message({children: 'Hello World'})}
    {message({children: 'Goodbye World'})}
  </div>
)

ReactDOM.createRoot(document.getElementById('root')).render(element)
```

```
function message({children}) {
  return <div className="message">{children}</div>
}

const element = (
  <div className="container">
    {React.createElement(message, {children: 'Hello World'})}
    {React.createElement(message, {children: 'Goodbye World'})}
  </div>
)

ReactDOM.createRoot(document.getElementById('root')).render(element)
```

```
function Message({children}) {
  return <div className="message">{children}</div>
}

const element = (
  <div className="container">
    <Message>Hello World</Message>
    <Message>Goodbye World</Message>
  </div>
)

ReactDOM.createRoot(document.getElementById('root')).render(element)
```

Apa bedanya? Mana yang benar?

## Props with props validation ( Without typescript )

```
function Message({subject, greeting}) {
  return (
    <div className="message">
      {greeting}, {subject}
    </div>
  )
}

Message.propTypes = {
  subject: PropTypes.string.isRequired,
  greeting: PropTypes.string.isRequired,
}

const element = (
  <div className="container">
    <Message subject="World" greeting="Hello" />
    <Message subject="World" greeting="Goodbye" />
  </div>
)

ReactDOM.createRoot(document.getElementById('root')).render(element)
```

# Hooks

Umumnya, interaksi dengan aplikasi akan membutuhkan sesuatu untuk mempertahankan “kondisi” dari sebuah aplikasi

Dalam react, kita menaruh ini dalam “state”. Cara kita menyimpan state, kita akan menggunakan special function yang biasanya disebut dengan hooks.



Hooks yang disediakan oleh react dan paling sering kita gunakan adalah sebagai berikut :

- useState
- useEffect
- useContext
- useRef
- useReducer

Selain itu kita juga dapat membuat hooks versi kita sendiri. Kata kunci dalam pembuatan hooks adalah diawali oleh kata ``use``

Setiap hooks, memiliki API yang unik satu sama lain. Sehingga penggunaanya pun berbeda-beda.

Beberapa mengembalikan sebuah nilai ( `useRef`, `useContext` )

Beberapa mengembalikan 2 buah value ( `useState` ) dan bisa juga tidak mengembalikan sebuah value ( `useEffect` )

# React JS

- ☒ React Introduction
- ☒ Installation
- ☒ Component & Props
- ☒ Hooks Introduction
- ☐ State
- ☐ UseEffect

# State

```
function Counter() {  
  const [count, setCount] = React.useState(0)  
  const increment = () => setCount(count + 1)  
  return <button onClick={increment}>{count}</button>  
}
```

useState adalah sebuah function yang membutuhkan single argument. Argument akan digunakan sebagai initial state, dalam kasus ini, maka initial nya adalah 0

## Handling State and State with initial value

```
function Greeting({initialName = ''}) {
  const [name, setName] = React.useState(initialName)
  function handleChange(event) {
    setName(event.target.value)
  }
  return (
    <div>
      <form>
        <label htmlFor="name">Name: </label>
        <input value={name} onChange={handleChange} id="name" />
      </form>
      {name ? <strong>Hello {name}</strong> : 'Please type your name'}
    </div>
  )
}

function App() {
  return <Greeting initialName="Kody" />
}
```



# React JS

- ☒ React Introduction
- ☒ Installation
- ☒ Component & Props
- ☒ Hooks Introduction
- ☒ State
- ☐ UseEffect


# UseEffect

UseEffect adalah built-in hooks dari react yang memungkinkan kita untuk menjalankan custom code setelah react melakukan proses render. ( dan re-render ).



```
React.useEffect(() => {  
  // your side-effect code here.  
})
```

Kita juga dapat memberikan argument kedua berupa array, yang disebut dengan “dependency array”



```
React.useEffect(() => {  
  // your side-effect code here.  
}, [  
  // dependency array  
])
```

Dengan ini, maka kita bisa kasih signal ke react bahwa kita ingin jalankan code tersebut apabila ada perubahan refrence pada dependency array



# Thank you