

React



**WA DATA SCIENCE
INNOVATION HUB**
Powering an AI Future

Our Sponsor

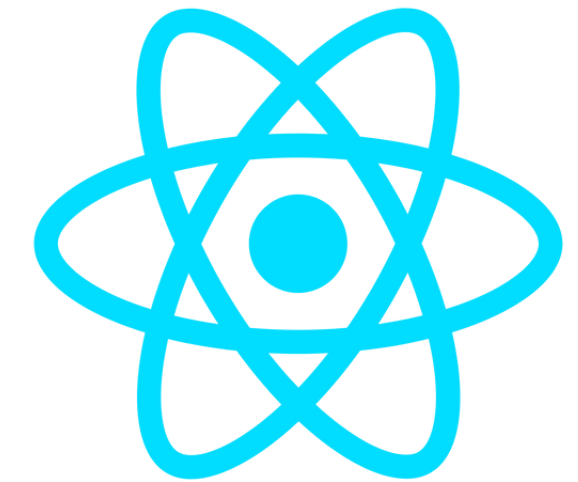


**WA DATA SCIENCE
INNOVATION HUB**

Powering an AI Future

React

- React is a JavaScript library for building UIs
- Used for front end development (can be used with other libraries to render certain environments)
 - React Native for mobile development
- Created and used by [Facebook](#)



React Native



Fundamentals of React



Traditional
approach



React
approach

- JavaScript and HTML in the same file (JSX)
- Components everywhere
- Embrace functional programming

Functions are first class citizens

- Functions can be **saved as variables**
- Functions can be **passed as arguments**
- Functions can be **returned from functions**

```
let add = function() {  
  console.log('Now adding numbers');  
  const five = 3 + 2;  
};
```

```
function performTask(task) {  
  task();  
  console.log('Task performed!');  
}  
  
performTask(add);
```

```
function foo() {  
  return function() {  
    console.log('What gets printed?');  
  };  
}  
  
foo  
foo();  
foo()();
```

Variables are immutable

```
let a = 4;  
  
a = 2; // Mutates `a`
```

```
let b = [1, 2, 3];  
  
b.push(4); // Mutates `b`  
  
let c = [...b, 4]; // Does not mutate `b`
```

- Use **const** instead of **let**
- Be careful when using **objects** and **arrays** as their properties/elements can still be modified even if declared with **const**
- Treat objects as **immutable**. If you want to update an object, **destructure** the object, **update** the values, and put them into a **new** object

Functions have no side effects

- They are called **pure functions**
- A pure function only calculates things based on its input and returns a result **without affecting anything outside of itself**
- Given the **same input**, a pure function always returns the **same result**

```
const b = [];  
  
function hasSideEffects() {  
  b = [0]; // Mutates `b`  
}
```

Anatomy of React Components

The component is just a function

Inputs are passed through a single argument object called "props" and often destructured or unwrapped to get the values inside

The function outputs HTML

```
export default function MyComponent({name}) {  
  return <div>Hello, world! My name is {name}</div>;  
}
```

```
const html = <MyComponent name="aaron" />;
```

The function is **executed** as if it was a HTML tag

Parameters are passed in as HTML attributes

Component Rendering

- When a component function executes, we say it "renders"
- A component re-renders only when **its props change** or when **a setter from the useState** hook is called

What are Hooks?

Hooks:

- **Special functions** that allow developers to “hook” into **state** and **lifecycle** of React components

State:

- **Local data** stored by a component that can change over time

Lifecycle:

- The different stages a component goes through during its existence, from creation (**mounting**) to updates (**re-rendering**) to removal (**unmounting**)

Built-in hooks:

We will cover
these today

{ useState
useEffect

We will not cover
these today

{ useContext
useReducer
useMemo
useRef
useCallback

useState


Purpose:

- Remember values internally when the component re-renders
- Tell React to re-render the component when the value changes


```
const [val, setVal] = useState(100);
```



The current value



A setter function to
change the value



The initial
value to use

useEffect

Purpose:

- Lets developers **synchronise** a component with an external system
- Can be used for performing **side effects** such as updating the screen, starting an animation, changing the data
- Effect will re-trigger if there is a change in one of its dependency values

Libraries

- **Component Libraries**
 - Material UI, React Bootstrap, headless UI
- **State Management and Data Fetching**
 - Redux, Zustand, Jotai
 - TanStack Query, SWR
- **Routing**
 - React Router, TanStack Router
- **Forms**
 - React Hook Form
- **Renderers**
 - React Three Fiber, React Figma
- **Graphics and Animations**
 - Framer Motion



NEXT.js

Dynamic Routes

```
// pages/user/[id].js
import { useRouter } from 'next/router'

export default function Page() {
  const router = useRouter()
  return <p>Hi, {router.query.id}</p>
}
```

url: localhost:3000/user/nicholas

Hi, nicholas

Linking and Navigating

```
import Link from 'next/link'

function Home() {
  return (
    <ul>
      <li>
        {/* /pages/index.tsx */}
        <Link href='/'>Home</Link>
      </li>
      <li>
        {/* /pages/about.tsx */}
        <Link href='/about'>About Us</Link>
      </li>
      <li>
        {/* /pages/user/[name].tsx */}
        <Link href='/user/nicholas'>Profile</Link>
      </li>
    </ul>
  )
}

export default Home
```

- The Next.js router allows you to do client-side route transitions between pages, similar to a single-page application.
- A React component called **Link** is provided to do this client-side route transition.
- **Link** is a drop in replacement for anchor tags, **<a/>**.
- Use **useRouter** for redirects, navigate back in history, and more.

API Routes

```
// pages/api/hello.ts
import type { NextApiResponse, NextApiRequest } from 'next'

type ResponseData = {
  message: string
}

export default function handler(
  req: NextApiRequest,
  res: NextApiResponse<ResponseData>
) {
  if (req.method === 'POST') {
    // Process a POST request
    const { name } = req.body
    res.status(200).json({ message: `Hello ${name}` })
  } else {
    // Handle any other HTTP method
  }
}
```

- NextJS has a node backend!
- Any file inside the folder ***pages/api*** is mapped to ***/api/**** path and will be treated as an API endpoint instead of a page.

Image Optimisation

```
import Image from 'next/image'

export default function Page() {
  return (
    <Image
      src="https://s3.amazonaws.com/my-bucket/profile.png"
      alt="Picture of the author"
      width={500}
      height={500}
    />
  )
}
```

- NextJS Image component extends the HTML element with features for automatic image optimisation
 - Size Optimisation
 - Automatically serve correctly sized images for each device, using modern image formats like WebP and AVIF.
 - Faster Page Load
 - Images are only loaded when they enter the viewport using native browser lazy loading, with optional blur-up placeholders.