goto

goto impact foundation

Generasi Gigih

Full Stack Engineer

Module 4 Session 1: Introduction to ReactJS

Our Agenda

Pretest (LMS)

Introduction

Core Material Lecture

Individual/ group/pair practice & debriefing:

QnA & Homework

Wrap Up

Let’s Discuss

Generasi Gigih

1. **Intro**
   a. what is React
   b. who uses React
   c. why React
2. **Quick start**
3. **Virtual DOM**
   a. What is DOM
   b. DOM vs Virtual DOM
   c. Interaction
4. **JSX**
   a. What is JSX
   b. SX vs JS
   c. JSX vs HTML
   d. displaying data in JSX
5. **Component**
   a. Intro
   b. display data on component
   c. styling and data on component
   d. Import and exporting components
   e. Recap

Generasi Gigih

What, Who, and Why React?

What is React JS?

- Open-source JavaScript **library** that is used for **building user interfaces** in a declarative and efficient way
- React is a **view layer library**, **not a framework** like Backbone, Angular etc → you can't use only React to build a fully-functional web app
- Developed by Facebook and run by community

Who uses React?

Top 30 companies using React JS, still so many others...

Who uses React?

Our decision to adopt React was influenced by a number of factors, most notably:

1) startup speed,
2) runtime performance, and
3) modularity.

UI engineers at Netflix

Why React JS?

- **Fast**: React uses Virtual DOM, creating faster web applications (more on this later)
- **Modular**: reusable components and unidirectional data flow
  - reduces the application's development time
  - easier to read and debug
  - less code
- **Scalable**: Large programs with a lot of fast updating data or real time data is where React performs best
- **Popular**:
  - Good documentation and small learning curve
  - Community support

Generasi Gigih

Quick Start

Steps

- Install node js https://nodejs.org/en
- Install editor (VS Code) https://code.visualstudio.com/ (WebStorm, Sublime Text etc can also be used)
- npx create-react-app my-app
- cd my-app
- npm start
- open http://localhost:3000/ to see your app

Generasi Gigih

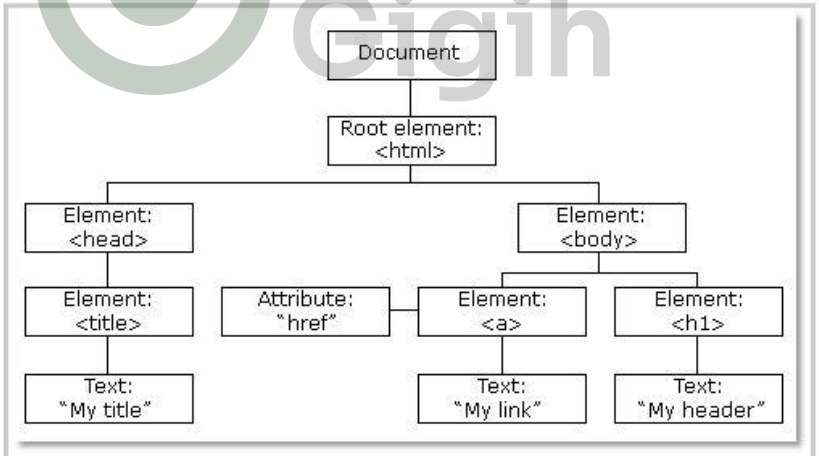
Virtual DOM, JSX, and Components

Fundamental terms in React

Virtual DOM

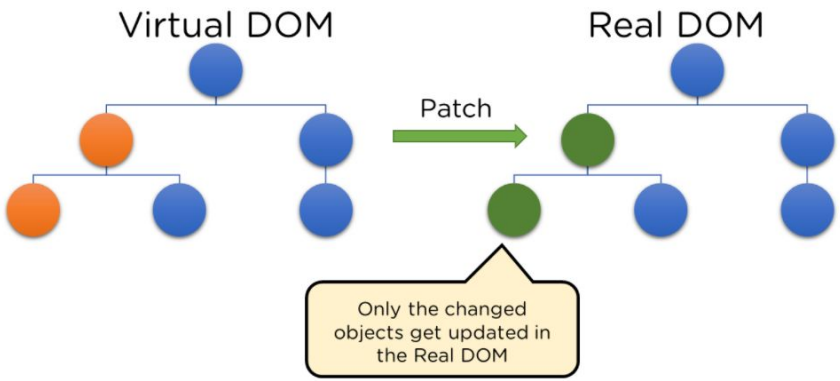
Part 1 : What is DOM

- DOM: Document Object Model
- DOM treats an XML or HTML document as **a tree structure** in which each node is an object representing a part of the document



Part 2 : DOM vs Virtual DOM

- Virtual DOM is React's **lightweight version of the Real DOM**
- Real DOM manipulation is **substantially slower** than virtual DOM manipulation.
- When an object's state changes, Virtual DOM **updates only that object in the real DOM rather than all of them**.



Part 3 : Interaction

- The virtual DOM is used for efficient re-rendering of the DOM
- When the state of an object changes in a React application, **VDOM gets updated**
- It then **compares its previous state** and then updates only those objects in the real DOM instead of updating all of the objects.

- This makes things move fast, especially when compared to other front-end technologies that have to update each object even if only a single object changes in the web application.

JSX

Part 1 : JSX

- Syntax extension to Javascript: a term used in React to describe how the user interface should seem
- It **embeds HTML into JavaScript code**
- Example: header is a JSX

```
const username = 'Generasi Gigih';

const header = <h1>Hello, {username}</h1>;
```

Part 2: JSX vs JS

- Web browser cannot read JSX directly (not a valid Javascript)
- If your file contains JSX, it needs to be compiled to regular Javascript for the browser to read

```
const user = 'Generasi Gigih'
# this is JS, only a string like above
const jsExample = '<div> a JS example </div>'
# this is a JSX that contains HTML
const jsxExample = <div> a JSX example </div>
```

Part 3: JSX vs HTML

- JSX is stricter than HTML
- You have to close tags like **
**
- Your component also **can't return multiple JSX tags**.
  - You have to wrap them into a shared parent,
  - like a ... or
  - an empty <>...</> wrapper:

```
function AboutPage() {
  // <> is an empty wrapper
  return (
    <>
      <h1>About</h1>
      <p>Hello there.<br />How are you doing?</p>
    </>
  );
}
```

Part 4: Displaying data in JSX

- Use curly braces to escape back to Javascript in JSX (already seen in previous examples)

```
const user = {
  name: 'Generasi Gigih',
  imageUrl: 'https://i.imgur.com/abcd.jpg',
  imageSize: 90,
};

const jsxExample = <h1>Hello, {user.name}! </h1>
```

Component

Part 1: Intro to Component

- React apps are made out of components.
- A component is **a piece of the UI (user interface) that has its own logic and appearance**. A component can be as small as a button, or as large as an entire page.
- React components are **JavaScript functions that return markup (JSX)**
- A component is a JS function/class that accepts inputs and returns an output of React element → same input produces same output

Generasi Gigih

React components are JavaScript functions that return markup (JSX)

```
function MyButton() {
  return (
    # this is JSX
    <button>I'm a button</button>
  );
}
```

We can nest a component to another component.

MyApp – parent component
MyButton – child component of MyApp

```
// we can use component MyButton and nest it to another component

export default function MyApp() {
  return (
    <div>
      <h1>Welcome to my app</h1>
      <MyButton />  // previously made component
    </div>
  );
}
```

Notice that <MyButton /> starts with a capital letter. **React component names must always start with a capital letter**, while HTML tags must be lowercase.

```
// we can use component MyButton and nest it to another component

export default function MyApp() {
  return (
    <div>
      <h1>Welcome to my app</h1>
      <MyButton /> // previously made component
    </div>
  );
}
```

```
function MyButton() {
  return (
    <button>
      I'm a button
    </button>
  );
}

export default function MyApp() {
  return (
    <div>
      <h1>Welcome to my app</h1>
      <MyButton />
    </div>
  );
}
```

The **export default** keywords specify the main component in the file

Not familiar? Read more

- https://developer.mozilla.org/en-US/docs/web/javascript/reference/statements/export
-
- https://javascript.info/import-export

Part 2: Root component

- In create react app, your app lives in src/App.js
- src/App.js is the **root component file** (but you can setup to another file)

App.js

```
function Profile() {
  return (
    <img
      src="https://i.imgur.com/MK3eW3As.jpg"
      alt="Katherine Johnson"
    />
  );
}

export default function Gallery() {
  return (
    <section>
      <h1>Amazing scientists</h1>
      <Profile />
      <Profile />
      <Profile />
    </section>
  );
}
```

Part 3: Exporting and Importing Components

- You don't have to put all of your components in the App.js file
- You can follow this steps:
  - **Make a new JS file** to put the components in.
  - **Export your function component** from that file (using either default or named exports) – will explain this later.
  - **Import it** in the file where you'll use the component (using the corresponding technique for importing default or named exports).

Part 4: Example (1)

Gallery.js

```
function Profile() {
  return (
    <img
      src="https://i.imgur.com/QIrZWGIs.jpg"
      alt="Alan L. Hart"
    />
  );
}

export default function Gallery() {
  return (
    <section>
      <h1>Amazing scientists</h1>
      <Profile />
      <Profile />
      <Profile />
    </section>
  );
}
```

Step 1: Gallery.js

Make a new JS file to put the components in. Notice that: there's profile component that is only used in the same file and **not exported.**

Step 2: Gallery.js

Export your function component. Notice that there is Gallery component as **default export.**

Part 4: Example (2)

App.js

```
import Gallery from './Gallery.js';

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

Step 3: App.js

In the root component file (App.js):

- Imports Gallery as a **default import** from Gallery.js.
- Exports the root App component as a **default export**.

Hands on

Hands on 1: Display a data on a component

Remember how to display data / variable in JSX?

How to use const user instead of hard code in below code?

```
const user = 'Generasi Gigih'

const jsxExample = <div> Hello, Generasi Gigih </div>
```

This is an example on how it looks like in a component.

To display data in a component, you just need to display data on its JSX by escaping back to JS

```
const user = {
  name: 'Generasi Gigih',
  imageUrl: 'https://i.imgur.com/abcd.jpg',
  imageSize: 90,
};

export default function Profile() {
  return (
    <>
      <h1>Hello, {user.name}</h1>
      <img
        src={user.imageUrl}
      />
    </>
  );
}
```

Hands on 2: Styling a component

Remember CSS?

Create a class named avatar and style it with border radius!

```
/* In your CSS */
.avatar {
  border-radius: 50%;
}
```

Calling the class from CSS to your component? Just use className after importing your css files

```
import './style.css'
```

```
<img className="avatar" />
```

Inline style in React? Just use style and double curly braces

```
<h1 style={{color:"lightcoral"}} >{name}</h1>;
```

Hands on 3: Combining style and data (exercise)

1. Use your local react app, create new file called Avatar.js
2. Create an object that contains information about user
3. Create a component that contains header and image
4. Use the user name on header tag
5. Use the user picture on image tag
6. Style the avatar and header text
7. Import the Avatar on App.js

Part 5: Component Recap

1. React lets you create components, reusable UI elements for your app.
2. In a React app, every piece of UI is a component.
3. React components are regular JavaScript functions except:
4. Their names always begin with **a capital letter**.
5. They return **JSX markup**.

Generasi Gigih

What we learned

1. What, Why, and Who uses ReactJS
2. JSX
3. Component
4. Import and Export component

Generasi Gigih

Q&A!

Generasi Gigih

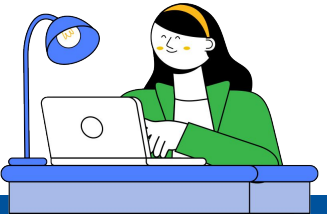
Exercise!

Escaping JSX and CSS

Exercise

1. Create a new react app and try to run in on your computer
2. Try to convert Module 1 exercise (**simplified** Spotify with HTML and CSS) to React App using JSX (UI only, no functionality)
3. If cannot be finished in class please continue at your own pace outside class (**homework**)
4. On the next session, some random participants will have to showcase

Notes: no need to be as complex as spotify, just do a simple one



Generasi Gigih

Showcase Time!

Generasi Gigih

Homework

Convert Spotify UI to React (as simple as possible)

goto

goto impact foundation

Generasi Gigih

See you in the next session!