**IHUOMA FAVOUR AGBARU**

**THEORY HOMEWORK**

1. What is React? (E.g. Consider: what is it? What is the benefit of using it? What is its virtual DOM? Why would someone choose it over the standard HTML / CSS stack?) (15 marks)

**Answer:** React is defined as a descriptive, effective, and adaptable JavaScript library used to create user interfaces.

**Benefits of using React**

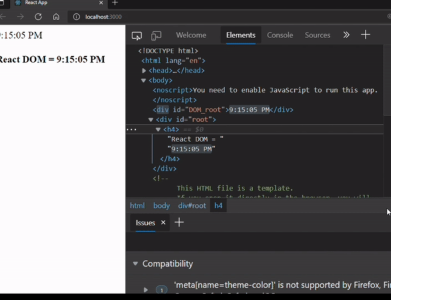
* **Reusable components**: The fundamental unit of any React application is the *Components.* The development time for the application is significantly shortened because these components, which each have their own logic and controls, may be utilized repeatedly throughout it.
* **Easy to understand:** React is simple to understand because it mostly mixes fundamental HTML, CSS and JavaScript concepts with a other(few) useful features.
* It can be used for the development of web applications and mobile applications. React is for we applications but there is another framework derived from react called react native that is used for mobile applications, which means React is used for both web and mobile application developments.
* React has dedicated tools for ease of debugging.
* React makes it easier to create dynamic web applications and offers better/more functionality.

**What is virtual DOM?**

**Answer:** A virtual document object model is a representation of the document object model but it’s lightweight copy.

**Why would someone choose it over the standard HTML / CSS stack?**

* A complete update is made to the virtual DOM.
* A comparison is made between the virtual DOM and its previous state. That which has changed is determined by React.
* On the actual DOM, just the altered objects are updated.
* The screen is changed when the real DOM is modified.



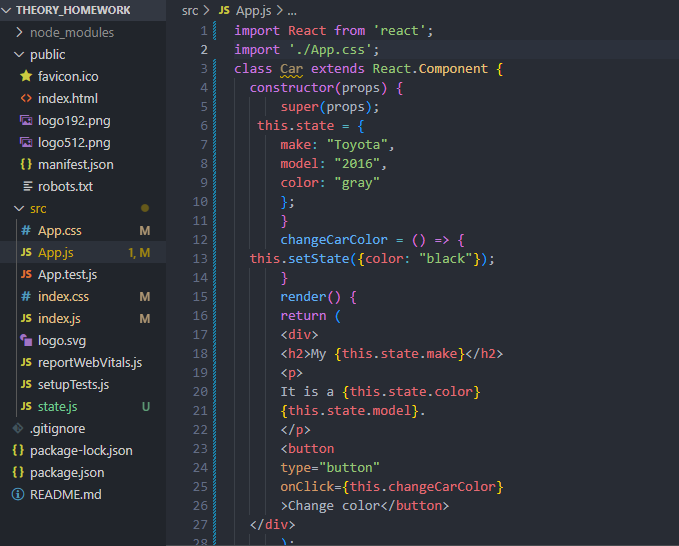
1. What are Props? What is State? What is the difference between them? (10 marks)

***What are Props?***

**ANSWER:** Props is short for properties in React, it allows users to pass arguments or data to components. Props in a component are read only and doesn’t allow changes.

***What is State?***

A state is defined as an object that stores property value for those attributed to a component that could change over a period of time.



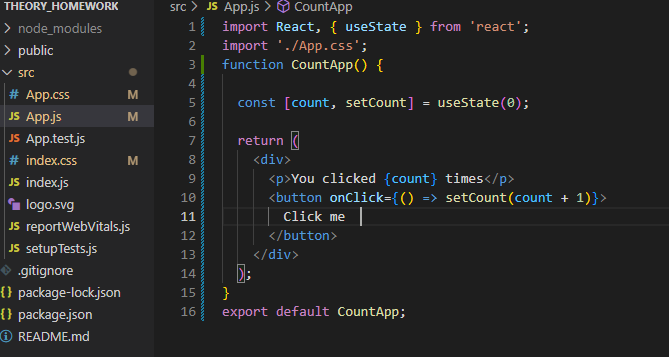
***What is the difference between them?***

|  |  |  |
| --- | --- | --- |
| **S/N** | **PROPS** | **STATE** |
| ***1*** | They cannot be modified after they have been sent. | The data is stored in the state and can be changed over time. |
| ***2*** | Functional and class components can benefit from the use of props. | State can only be used by class components. |
| ***3*** | The parent component sets props for the children components. | Event handlers are responsible for updating the state. |
| ***4*** | They are used to send data and event handlers to a component's children. | The data of the components that must be presented to it store the view in the state. |

1. What are React Hooks? How do they differ from existing lifecycle methods? (10 marks)

***What are React Hooks?***

**ANSWER:** React Hooks are defined as functions which hooks into React state and lifecycle features from functions. Hooks allow you to use State and other React features without writing a class.





***How do they differ from existing lifecycle methods?***

|  |  |  |
| --- | --- | --- |
| ***S/N*** | ***LIFECYCLE METHODS*** | ***LIFECYCLE HOOKS*** |
| ***1*** | Lifecycle methods was introduced with initial versions. | Lifecycle hooks was introduced in 2018. |
| ***2*** | Does not require constructor to initialise. | Requires constructor to initialise. |
| ***3*** | Works with class based components. | Works with function based components. |

1. Design the perfect door - what should it look like, what are the components for it? What design heuristics should it follow, and how does your design match? What made you choose this design? (20 marks).
   1. Consider in particular (likely need to do independent learning): who are your stakeholders? What is their personas? What is the doors requirements and how will your stakeholders benefit from your solution?

**ANSWER:** A door is a moving mechanism which is more like a barrier made of wood, stone, metal, glass, paper, leaves, hides, or a combination of materials that allows the passage of humans, goods and animals.

**Components of doors/design:**

The components for my design are as follows:

* Clarity: Understanding how I wanted my design to be was the first component I put into consideration. How should my door look and for who should it be designed for.
* Time: Timely delivery of how my door should look like was put into consideration.
* Color: I wanted a minimalist door and a glass door.
* Balance: The rectangles and lines used in the design should be balance and not contrast each other.
* Unity: The manner in which different forms in a design piece relate to each other is important.
* Space: The space between the the shapes, Depth, perspective, overlapping, illusion.

**Design Heuristics**

Design heuristics is a usability inspection method for computer software that helps to identify usability problems in the user interface design.

**My design/door followed the following design heuristics pattern:**

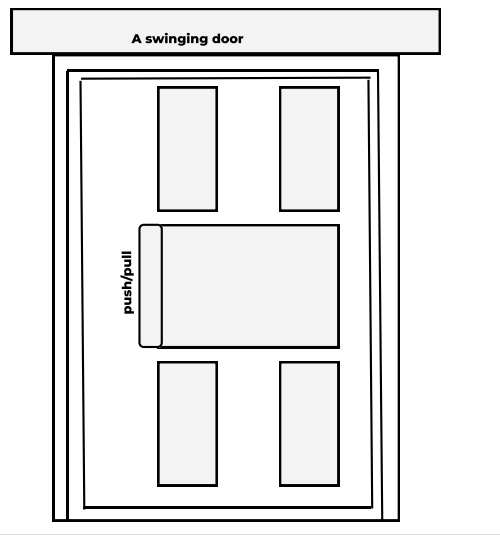
* **Flexible and efficient to use**: A disabled/first time user, should not have issues using the door. It has the option for push for the disabled and someone without issues can also choose to either use the push or pull option.
* **Aesthetics**: my design/door is aesthetically pleasing to behold as it’s a simple minimalist glass door.
* **Consistency and standards**: The user does not have to wonder if the push/pull sign means different things as it solves the need of the user.

**Why did I choose this design?**

I considered an adult human who might be too tired to pull a door after a stressful day and have no energy left in them than to get to their destination or home and I used the pull/ push swinging door as a solution to their needs.

My stakeholders are adult human of both genders. Their personas are as follows;

* Ease of getting things done.
* Minimalist lovers.
* Straight to the point solution.



<https://www.figma.com/file/9G7I8267NDjQtOsQFLgUBW/Untitled?node-id=1%3A2>

1. What is Angular, and how does it differ from React? You may need to conduct independent research and learning for this (10 marks)

**ANSWER:** Angular is a development platform and a component based framework for building scalable web applications built on typescript. In Angular, everything happens under the same place. It offers an ecosystem that allows you to create your application effortlessly. Features like templating two-way binding, RESTFUL API modularization, Ajax handling, dependency injection and many more features that makes your application development accessible and compact. The primary advantage of using Angular is Google. Google has a long- term support for Angular.

**How is Angular different from React**

|  |  |  |
| --- | --- | --- |
| **S/N** | **ANGULAR** | **REACT** |
| **1** | Angular supports both one way and two ways data binding. | React supports one way data binding. |
| **2** | Typescript is statically typed language and is a superset of Javascript. | Typescript can write Javascript but it isn’t included by default. |
| **3** | Incremental Dom: this is when a new DOM is created, it compares it to the previous one and applies the differences to the actual DOM. | Anytime the DOM changes, a new virtual DOM is created, compared to the previous one, and only the differences are modified in the “real” DOM. |
| **4** | Angular is a full- fledged framework | React is a Javascript library. |
| **5** | NGRX is a state management library, which allows the use of reactive state management. NGRX stores all the state in a single tree, allowing you to access all the forms from anywhere in an application. | Every individual react components, have a state. The states are managed separately. REDUX works as a state management library for React. |

1. Please describe Redux in as much detail - especially consider: why would someone use it? What is it? What's the benefit of using it? Are there any potential drawbacks to using it? How can it be added to a project? What is dispatch, provider, actions, etc? (15 marks)

**ANSWER:** Redux is just a store where you may store the values of the variables in your applications state. In order to prevent components from updating or reading the store at random, Redux develops a process and procedures to interface with it.

**Principles of REDUX**

The three most crucial principles are listed below and are what make Redux predictable.

* **Single Source of Truth**

An object tree within a single store houses the state of your entire application. Debugging is simple and development moves more quickly because the entire application state is maintained in a single tree.

* **Read-only state**

Emitting an action, or an object explaining what occurred, is the sole way to change the state. This implies that no one can actually alter the status of your program.

* **Pure functions are used to effect changes.**

You create pure reducers to describe how actions change the state tree. The main location for state alteration is a reducer. Reducer is a function that returns after accepting state and action as arguments.

## **How Redux works**

The way Redux works is simple. There is a central store that holds the entire state of the application. Each component can access the stored state without having to send down props from one component to another.

There are three building parts: actions, store, and reducers.

## What is an action?

An action is a plain object that describes the intention to cause change with a type property. Actions are sent using the ***store.dispatch()*** method.

**What is a reducer?**

Reducer is a function that connects actions and states. A decision is made with the goal to bring about change. The reducer is responsible for this modification. Redux is more dependable, centralized, and debuggable because reducer is the sole way to alter states. Redux follows the unidirectional data flow. By enforcing the restriction on how and when state updates can occur, Redux decreases the complexity of the code. This makes it simple to manage updated states.

### **Store in Redux**

The store holds the application state. It is highly recommended to keep only one store in any Redux application. You can access the state stored, update the state, and register or unregister listeners via helper methods.

**What is dispatch?**

Dispatch is a function of the Redux store. It dispatches an action. This is the only way to trigger a state change.

**What is a provider?**

The react-redux node package has provided us with a component to use called Provider. In order to pass the store as an attribute, we use the Provider object. Instead of storing the store as props, we may avoid doing so by passing the store as an attribute in the Provider component.

## **Why use Redux?**

When using Redux with React, states will no longer need to be lifted up. This makes it easier for you to trace which action causes any change.

### **Redux is maintainable**

Redux is strict about how code should be organized, which makes it easier for someone with knowledge of Redux to understand the structure of any Redux application. This generally makes it easier to maintain.

### **Debugging is easy in Redux**

Redux makes it easy to debug an application. By logging actions and state, it is easy to understand coding errors, network errors, and other forms of bugs that might come up during production.

**Performance Benefits**

React Redux implements many performance optimizations internally so that your own connected component only re-renders when it actually needs to.

### **Ease of testing**

It is easy to test Redux apps since functions used to change the state of pure functions.

### **Server-side rendering**

Redux can also be used for server-side rendering. With it, you can handle the initial render of the app by sending the state of an app to the server along with its response to the server request. The required components are then rendered in HTML and sent to the clients.

**DRAWBACKS OF REDUX**

**1. Lack of Encapsulation**

There is no possibility of encapsulating any data in the Redux library. So, the chances of security issues grow as an application becomes heavier with new components. Since there is no data encapsulation, any component can access it.

**2. Restricted Design**

Most developers have complained about the rigidity of Redux. It is very restricted with minimal alternatives. So, you can use it for a trivial application or anything of that nature as state management will become more complex than it is without Redux.

**3. Excessive Memory Use**

In Redux, every state is immutable. In order to update a state, a reducer has to return a new state every time. This process creates compilation in the long run leading to excessive memory usage.

**4. Increased Complexity**

Even though the benefits are plenty, Redux has an additional complexity in layering when it comes to using manipulation logic actions or reducers.

**5. Time-Consuming**

For small or medium-level local state management, Redux can be highly time-consuming as it requires more inputs of the boilerplate code to begin writing.

**HOW CAN REDUX BE ADDED IN PROJECTS**

* **Install Redux by using the command** *npm install redux react-redux.*
* **Edit the index.js file in the react app by writing** *import{createStore} from ‘react-redux’ and also import {provider} from ‘Redux’.*
* Create the rootreducer file.
* Create the reducers.
* Create the actions.
* Connect the components to the Redux store.

1. Please describe Linux in as much detail as possible (feel free to use notes made during lessons, or draw from the lesson directly!). Especially consider: what is its history? Why would someone use it over other existing operating systems? How does Windows and Mac OSX differ to Linux? How does Linux function, what are some unique features to it? How can it be installed today? (10 marks)

**ANSWER: History of Linux**

Linux was created in 1991 by Linus Torvalds, a then-student at the University of Helsinki. Torvalds built Linux as a free and open source alternative to Minix, another Unix clone that was predominantly used in academic settings. He originally intended to name it “Freax,” but the administrator of the server Torvalds used to distribute the original code named his directory “Linux” after a combination of Torvalds’ first name and the word Unix, and the name stuck. The term “Linux” technically refers to just the Linux kernel. Most people refer to the entire operating system as "Linux" because to most users an OS includes a bundle of programs, tools, and services (like a desktop, clock, an application menu, and so on). Some people, particularly members of the [Free Software Foundation](https://fsf.org/" \t "https://opensource.com/resources/_blank), refer to this collection as GNU/Linux, because many vital tools included are GNU components. However, not all Linux installations use GNU components as a part of the operating system.

A Linux system is divided into three main parts namely:

**Hardware:** This includes all the hardware that your system runs on as well as memory, Central processing Unit(CPU), disks, etc.

**Linux Kernel:** The kernel is the core of the operating system. It manages the hardware and tells it how to interact with the system.

**User Space:** This is where users will be directly interacting with the system.

**How is Linux different from other operating systems.**

* Linux is open-source software. The code used to create Linux is free and available to the public to view, edit, and for users with the appropriate skills to contribute to.
* Linux is incredibly customizable, because not just applications, such as word processors and web browsers, can be swapped out. Linux users also can choose core components, such as which system displays graphics, and other user-interface components.

## **Who uses Linux?**

Companies and individuals choose Linux for their servers because it's secure, flexible, and you can receive excellent support from a large community of users, in addition to companies like Canonical, SUSE, and Red Hat, each of which offer commercial support. Devices such as Android phones and tablets and Chromebooks, digital storage devices, personal video recorders, cameras, wearables, and more, also run Linux.

**Features of Linux?**

* Linux is a highly secure system.
* Linux freeware is the OS of choice for Server environments due to its stability and reliability. A Linux based server could run non-stop for years without reboot.
* Being open-source, anyone with programming knowledge can modify it.
* Easy to learn.
* Linux is a multi-programming system means multiple applications can run at same time.
* Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs.
* Linux is a multi-user system means multiple users can access system resources like memory/ ram/ application programs at same time.

**How can Linux be installed?**

* Download required files into a USB stick or CD/RWCD.
* Download Universal USB Installer (this could be used in place of a bootable USB stick).
* Select Distribution (Ubuntu, Gentoo, Fedora, Debian, etc.)
* Install the chosen distribution.
* Check to see your window if it has the congratulatory message specifying the distribution is successfully installed.

1. What are they, and which is better between Class components and Functional components? Provide a discussion. Consider: Go deep - how does each one work? What is the unique properties or behaviours to each one? Why would someone use one over the other? What are the advantages and disadvantages of each one? Who benefits from these advantages and disadvantages, who is it suitable for? (10 marks).

**ANSWER:** A React functional component is a simple JavaScript function that accepts [props](https://www.educative.io/edpresso/what-are-props-in-react) and returns a React element.

A class component is a JavaScript class that extends React.Component which has a render method.

Functional components lack a significant amount of features as compared to class-based components. The gap is made up with the help of a special ReactJS concept called “hooks”. Hooks are special functions that allow ReactJS features to be used in functional components.

Functional components do not have access to dedicated state variables like [class-based components](https://www.geeksforgeeks.org/reactjs-class-based-components/). The only “state” that a functional component effectively has access to are the props passed to it from its parent component. ReactJS has access to a special hook called [useState()](https://www.geeksforgeeks.org/reactjs-usestate-hook/) that can be used for giving the illusion of working with the state in functional components.

For a functional component, the component is a function that itself returns what you want to render. It takes the place of the render method in the class component. But, traditionally, functions can‘t maintain state or provide additional functions to hook into for lifecycle methods.

In Version16.8, React released hooks which provided functions that can be called to create and maintain state, create side effects, and more.

**Differences between the functional components and Class components.**

|  |  |
| --- | --- |
| **Functional Components** | **Class Components** |
| A functional component is just a plain JavaScript pure function that accepts props as an argument and returns a React element(JSX). | A class component requires you to extend from React. Component and create a render function which returns a React element. |
| There is no render method used in functional components. | It must have the render() method returning JSX (which is syntactically similar to HTML) |
| Functional component run from top to bottom and once the function is returned it cant be kept alive. | Class component is instantiated and different life cycle method is kept alive and being run and invoked depending on phase of class component. |
| They are known as Stateless components as they simply accept data and display them in some form, that they are mainly responsible for rendering UI. | They are also known as Stateful components because they implement logic and state. |
| React lifecycle methods (for example, componentDidMount) cannot be used in functional components. | React lifecycle methods can be used inside class components (for example, componentDidMount). |
| Hooks can be easily used in functional components to make them Stateful.  example: const [name,SetName]= React.useState(‘ ‘) | It requires different syntax inside a class component to implement hooks.  example: constructor(props) {     super(props);     this.state = {name: ‘ ‘}  } |
| Constructors are not used. | Constructor are used as it needs to store state. |

**Benefits of using Functional Components over class components.**

* Functional components are more concise, leading to **cleaner, less complex code**. They do not include lifecycle methods or inherited members that may or may not be required for code functionality.
* Anything that can be done using class components can also be done using functional components. (The only exception is that React supports a special class component called Error Boundaries that cannot be duplicated as a function component.)
* Class components have state and lifecycle methods associated with them, so you must understand when lifecycle events will occur and how to respond to them to manage state.
* Classes also require additional setup for the component or to make API calls for data, primarily implemented via the constructor.

**When to use them?**

If your code is short and easier to read functional components should be used. If your code is long and has more lines than the normal, class components should be used.