**Christabel**

Background: React JS is a JavaScript framework for creating user interfaces. It was created by former Facebook engineer Jordan Walke and first released to the public in 2013. It is maintained by Facebook (Meta) and is released under the MIT license, which means it has few restrictions on its use.

**Christabel**

**Kangwei**

In the last part, we are going to learn about how React is different from other frameworks. We will focus on how React is different from Angular and Vue. As we have learned from the previous slides, React uses JSX and virtual DOM. JSX allows programmers to write HTML inside JavaScript files. Virtual DOM helps to compare the previous state of the component and update the items that were changed in the Real DOM , instead of updating all the components again. And react has one way data binding which keeps everything modular and fast. One-way data binding is unidirectional data flow which means that when a developer designs a React application, he will often nest child components within parent components. This helps developers have a better control of the web application which allows them to know where and when error occurs.

For Angular, it is a full-fledged MVC framework. Angular uses regular DOM. So, instead of updating the only ones that were already changed, it will update the entire tree structure of HTML tags. Angular has two-way data binding. There is model state and UI state. The model state reflects any changes made to the UI elements. And UI state reflects any changes in the model state. This feature helps the framework to connect the DOM to the model data through the controller.

And Vue, it contains a special file type similar to JSX. It also uses Virtual DOM which is similar to React. And it has reactive two-way data binding which means it supports both one-way and two-way data binding. It automatically selects the correct way to update the elements based on the user’s input type.

Next, we will compare these three frameworks in the aspects of state management, routing, and client-server communication. For state management, React contains countless libraries and it has an inbuilt solution called React Context. React Context lets the developer pass state through the component tree without passing properties down manually at entry level. Angular uses its inbuilt solutions to manage the state without using the third-party packages. Angular has two inbuilt solutions called NGXS and NGRX. Vue has its own state management library called Vuex. It helps to store and share data across the application without trading off its performance.

Next, For the routing, React doesn't have an inbuilt solution for routing but it contains third-party packages called React Router and Wouter. For Angular, it has its own inbuilt Router to handle normal routing, nested routing, redirect, parameter passing etc. For Vue, it is similar to Angular which has an inbuilt router.

For the third aspect, the client-server communication, React uses Axios and Fetch to consume and display data from an API. Angular uses the HTTP Client module to let the application execute standard API calls. For Vue, it is similar to React which uses Axios and Fetch.

**Micaela**

JSX Syntax

* JSX is a syntax extension to JavaScript. Used with React to describe what the UI should look like. JSX produces React “elements”
  + React doesn’t require using JSX, but it can be a helpful visual aide
  + Not a string, and not HTML
* ~~According to reactjs.org: “React embraces the fact that rendering logic is inherently coupled with other UI logic: how events are handled, how the state changes over time, and how the data is prepared for displayed. Instead of artificially separating technologies by putting markup and logic in separate files, React separates concerns with loosely coupled units called “components” that contain both”~~
* To use a variable or any valid JavaScript expression inside JSX, you wrap it in curly braces
  + Ex: reference the property of an object (user.name)
  + Ex: embed the result of calling a JavaScript function, formatName(user)
  + Here we have a normal JavaScript function called formatName. Then down below we declare an element using the result of formatName.
* After compilation, JSX expressions become regular JavaScript function calls and evaluate to JavaScript objects
  + So you can do things like assign JSX to variables, return it, and anything else you can do with JavaScript objects
  + In this example, depending on the value of user, different JSX expressions would be evaluated and different objects would be returned
* As said before, JSX gets compiled down to a normal Javascript function call to React.createElement(). These two snippets of code have the same functionality. They both create React elements. React elements can be conceptually thought of as objects taking this form. Props is its own object within the element object and contains the attribute values and whatever children the element has. We’ll talk more about props in a little bit.
  + Use quotes to specify string literals as attributes:

Rendering Elements

* Elements are the smallest building blocks in React apps. This is the example element from before. Recall that DOM = Document Object Model. React elements are cheaper to create than browser elements and the ReactDom takes care of updating the DOM for us.
* The root DOM node is just the node where everything inside will be managed by ReactDOM. You can have multiple root nodes, but usually you just use one root node.
  + To render a React element into the root node, you pass both to the ReactDom.render() function. Here the element named element will be inserted as a child of the root div node.
  + You can see the result at the bottom.
* Immutable
  + React elements are immutable meaning you can’t change its attributes or children. However you can call render again in order to alter the HTML.
  + When you call render again, ReactDOM compares the new element to the previous element and only modifies the part of the code that changed.
  + In this example, Even though we create a new element describing the whole UI tree on every tick, only the text node whose contents have changed gets updated by React DOM.
    - If you inspect the page you would see that only the date in the curly brackets is changing in the HTML code.

Components and Props

* Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.
  + Components are often confused with elements but they are different. Components can be thought of similarly to JavaScript functions. They take inputs called “props” and return React elements.
  + One way to define a component is through function components.
    - Here we have a function component called Welcome. It is called a function component because it accepts a single “props” object argument and returns a React element and is literally a normal JavaScript function as in it is not a class
* Another way to define a component is to use an ES6 class and extend React.Component. Again the resulting component is called Welcome.
* React elements can represent DOM tags (on the left) or user-defined components (on the right).
  + When React sees an element representing a user-defined component, it passes JSX attributes and children to this component as a single object (“props”). The user defined component will then return a React element.
* In this example, we define an element using a user-defined component. We then call render with that element. In the render function, React calls the Welcome component with the props object having a property named “name” with the value “Sara”. The Welcome component then returns the React element corresponding to a header tag containing “Hello, Sara”. ReactDom then updates the DOM to reflect the React element.
  + You can see the result at the bottom.
* Components can refer to other components. This encourages abstraction and separation of functionality so you can reuses certain components for buttons, forms, etc.
  + You could also create a huge component with a bunch of nested elements but this is hard to manage.
  + One rule that React is strict with is that React components must never attempt to change their “props” inputs. This can be avoided by using the concept of state to respond to changes in user actions, network responses, etc.

**Nathan**

State: Classes were introduced to JavaScript in 2015 with the release of ECMAScript 6, and they are used heavily in React. For a component to control its own data, as opposed to being controlled by another member, it has to have state. A stateful component is more reusable and object-oriented than a stateless one. To create a component that has state, you create a class for that component and include its default state in the constructor. The key implementation difference between using state and using props (short for “properties”) is that you specify the component’s state using ‘this.state’. There are a few important things to remember when it comes to state. One is that state is encapsulated, or local, meaning that it is accessible only by its owner. This means that components should not know or care if another component has state or how it may use it. This follows the black box programming paradigm and the principle of design-by-contract, where the only important parts of a system are its inputs and outputs.

Lifecycle: The life cycle of a component is exactly what it sounds like; when a component is no longer needed, it can be destroyed to surrender its resources. This is similar to freeing dynamically allocated resources in non-garbage-collecting languages. React uses a few special methods to specify the life cycle of a component. These methods are called componentDidMount() and componentWillUnmount(), and they render and remove the component, respectively.

Handling Events: Event handling with React works just like event handling with the document-object model (DOM). React uses JSX, an extension to Javascript syntax, which merges markup and logic for a component. JSX essentially converts HTML tags to react components, which lets you better visualize what you’re creating when you’re writing code. JSX is not required to use React effectively, but it is helpful.

Conditional Rendering: Controlling which components render and when using React is extremely simple; you can just use the default Javascript boolean-checking syntax, like if/else, ternary operator, switch statement, etc. Javascript tricks like short circuit evaluation can make code shorter and more readable, but this is ultimately up to the programmer. All you need to do is treat component actions like you would any other statement. When calling a function that renders a component, it’s as simple as returning the version that you would like rendered based on whatever checks are appropriate. Encapsulating versions of a component and then rendering the correct one makes your code more direct and logical than splitting versions into separate functions.