React notes

React → A javascript library for building user interfaces

HTML

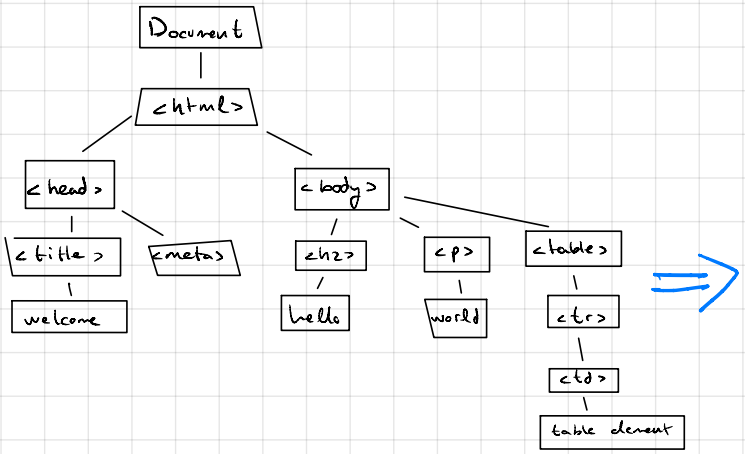
- Declarative for static content
- When element change whole HTML DOM is rendered. Update whole tree

VS

React

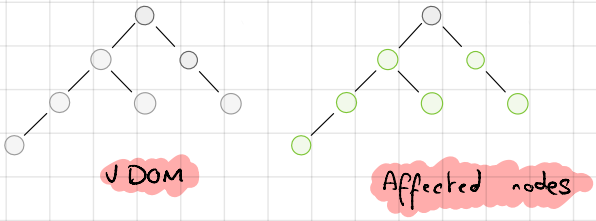
- Declarative for dynamic data
- When some element change virtual DOM render HTML tree from JS (in memory)

RDOM



VDOM

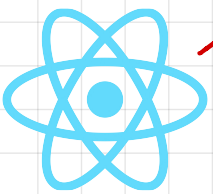
VDOM is a programing concept where VDOM is kept in memory synced with RDOM by ReactDOM library. This process is called reconciliation.



- DOM manipulation is very expensive
- There is too much memory wastage
- it updates slow
- it can directly update HTML
- Creates a new DOM if the element updates
- it allows us to directly target any specific node (HTML element)
- it represents the ui of your app

≠

- DOM manipulation is very easy
- No memory wastage
- it updates fast
- it can't directly update HTML
- Update the JSX if the element update
- it can produce ~ 200k VDOM nodes/sec
- it's a virtual representation



Components
- Live functions
- Input: props, state ⟩ Output: UI
- Reusable and composable
- <Component />
- Can manage a private state

Reactive updates
- React will react
- Take updates to the browser

Virtual views in memory
- Generate HTML using JS
- No HTML template lang
- Tree reconciliation

React components

Uppercase

```
function Hello () {
  1 // return <div> Hello </div>
  2 return React.createElement (
        'div',
        null,
        'Hello'
    );
ReactDOM.render (
  1 // <Hello />
  2 document.getElementById ('element')
);
```

React hooks

Hooks are functions that let you "hook into" React state and lifecycle features from function components. They let you use React without classes

- Hooks
  - useState
    - State variable
    - Function that modify
  - useEffect
    - Function
      - body: Use an effect that made changes
      - return function: Use effect after component unmount
  - React API Hooks (View official documentation)
  - Own + hook

custom state variable

function that change state

```
function logRandom () {
    console.log (Math.random());
}
function Button () {
    const [counter, setCounter] = useState (0);
    return <button onClick = {logRandom}> counter </button>
}
ReactDOM.render (
    <Button />
    document.getElementById ('element'),
);
```

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class

Can declare multiple states

Equivalent Class example

```
class Example extend React.Component {

    constructor (props) {
        super(props);
        this.state = {
            count : 0
        };
    }

    render () {

        return (
            <button onClick={() => this.setstate ({ count : this.state.count +1 })}>
                Click me
            </button>

        );
    }
};
```

Reading State

```
<p> You clicked { this.state.count} times </p>

<p> You clicked {count} times </p>
```

Class type (1)

Function type (2)

Updating State

```
<button onClick={() => this.setState ({ count: this.state.count +1 })}>
    Click me
</button>

<button onClick={() => setCount (count + 1)}>
    Click me
</button>
```

1

2

Custom Hooks

```
import { useState, useEffect } from 'React';

function useFriendStatus (friendId) {
    const [isOnline, setIsOnline] = useState (null);
    useEffect ( () => {
        function handleStatusChange (status) {
            setIsOnline (status.isOnline);
        }
        ChatAPI.subscribeToFriendStatus ( friendId, handleStatusChange);
        return () => {
            ChatAPI. unsubscribeToFriendStatus ( friendId, handleStatusChange);
        };
    });
    return isOnline;
}
```

- Use set prefix convention
- Do 2 components using the same hooks share state? No
- How does a custom hook get isolated state? Each call to a hook gets isolated state. Because we call 'useFriendStatus' directly.

```
function FriendListItem (props) {
    const isOnline = useFriendStatus (props.friend.id);
    return (
        <li style={{ color: isOnline ? 'green' : 'black' }}>
            {props.friend.name}
        </li>
    );
}
```

```
function FriendStatus (props) {
    const isOnline = useFriendStatus (props.friend.id);
    if (isOnline === null) {
        return 'Loading...';
    }
    return isOnline ? 'Online' : 'Offline';
}
```