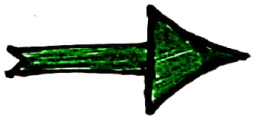
Handwritten Notes 

"Everything in React is a Component"

REACT COMPONENTS

 Save for Later

Ashraya K K
@ashrayaa.

COMPONENT

'Everything is a component in React'.

React Components

→ 2 types :-

a) Functional component - NEW WAY of writing code.

b) Class Based component - OLD WAY

→ Functional component

- is nothing but a javascript function.
- is a normal JS fn which returns some piece of react elements (here, JSX).

Eg :-

```
const HeaderComponent = () => {
    return <h1> HelloWorld </h1>;
};
```

- For any component,

Name starts with capital letter.

(It is not mandatory, but it's a convention)

to render functional component, write :-

<Header Component />

React element

```
const heading = (
  <h1 id="title"
      key="h1">
    Hello World
  </h1>
);
```

converting it into arrow fn will make functional component.

Functional Component

```
const heading = () => {
  return (
    <h1 id="title"
        key="h1">
      Hello World
    </h1> );
};
```

React element is finally an object

Functional component is finally a function.

The Next Amazing thing

(1)

*
```
const Title = () => {
  <h1>Hello World</h1>
}
```

} is a functional component

* const HeaderComponent = () => {

```
return (
    <div>
        <Title />
        <h2> Nomaste React </h2>
        <h2> Hai all </h2>
    </div>
);
};
```

instead of this you can write { Title() }

This is 'component composition'

This is a normal javascript function !!

---

(2) * const title = (

```
<h1> Helloworld </h1>
);
```

} is a normal variable

* const HeaderComponent = () => {

```
return (
    <div>
        {title}
        <h2> Nomaste React </h2>
        <h2> Hai all </h2>
    </div>
);
};
```

NB:

* Whenever you write JSX, you can write any piece of javascript code between paranthesis {}. It will work.

* JSX is very secure.
  JSX makes sure your app is safe.
  It does sanitization.

```
const data = api.getData();

const HeaderComponent = () => {
  return (
    <div>
      {data}
      <h2> Hello World </h2>
    </div>
  );
};
```

Write anything inside {}, JSX will sanitize the code.

Component Composition

If I have to use a component inside a component. Then, it is called component composition / Composing components.

CLASS BASED COMPONENTS

- It was less maintainable, have more code and a little messy.
- They are no longer used.
- Developers can do almost everything using functional components now.
- Functional component, at the eod is just normal js functions. Similary ↓
- Class based components are just normal js class.

CLASS BASED COMPONENT	FUNCTIONAL COMPONENT

`import React from "react";`
`class Profile extends React.component{`
`render(){`
`return <h1>Profile</h1>`
`}}`
`export default Profile;`
	`const Profile =() => {`
	`return (`
	`<h1>Profile</h1>`
	`);`
	`}`
	`export default Profile;`

ProfileClass.js	ProfileFunc.js.

React.component come from here.

You've to tell react that this is a class component not normal javascript class

name of the component

keyword class

```
import React from "react";

class Profile extends React.component {

  render(){ => return some JSX

    return <h1> Profile </h1>;
  }}

export default Profile;
```

Whatever you return from this will be injected to the DOM.

You can't create a class component without render method

`render(){}` is the only mandatory method for class based components.

PROPS IN CLASS COMPONENTS

In About.js, Suppose We called class component 'ProfileClass.js' as :- and if we pass props inside it :-

```
<ProfileClass name={"Ashraya"} />
```

So, in class component, we have 'this' keyword.

So, in 'ProfileClass.js'

```
<h2> Name : {this.props.name} </h2>
```

<ProfileClass name={"Ashraya"} xyz={"abc"}/>

Even if there are many props, react will collect all these props and attach with keyword 'this'. And I can use the props like :-

<h2> Name : {this.props.name} </h2>
<h2> xyz : {this.props.xyz} </h2>

STATE IN CLASS COMPONENT

```
class Profile extends React.component {
    constructor(props); {
        super(props);
        this.state = {
            count : 0,
            count2 : 0,
        };
    }
```

creates & initialize the component's state.

This is necessary because it passes the received props to the constructor of the base component class, allowing the component access it's properties.

Constructor function is called only once, when the component is first created & initialised, making it an ideal place to set the initial state of the component using this.state = {..}.