npx is an npm package runner which install when we install node.

2 ways to incorporate React into your project:
1). npx create-react-app project_name
    npx package runner

2). npm install create-react-app -g
    create-react-app project_name

we don't want to install globally and don't want to update it again and again so will prefer npx commands to use.

Component Types:
Stateless Functional Component
        ->are JS Functions
                eg: function Welcome(props){
                        return <h1>Hello, {props.name}</h1>;
                    }
Stateful Class Component
        ->class extending Component class
        they must consider Render method returning HTML
                eg: class Welcome extends React.component{
                        render(){
                          return <h1>Hello, {this.props.name}</h1>;
                        }
                    }

_____
Components naming we'll use PascalCase convention.

functional components are just JavaScript Functions, they can receive optionally an object of properties (props) and return HTML(JSX).

while working with function component need to import:
import React from 'react'

component folder in src folder.
export these components and import in app.js (as it is the root file where the program starts).

so for the component to be exported we can export like:
export default ComponentName;

and import in app.js like;
import ComponentName from "./components/ComponentName"

although writing normal JS functions
function Greet(){
        return <h1>Hello Nandini</h1>
}
is ok,

but suggested to use ES6 errow functions Syntax;
const Greet = () => <h1>Hello Nandini</h1>
(syntax of Arrow function is more concise and we'll check the more advantages later)

-->> Exporting and Importing of Components
There are two types of export basically:
default and named export.

so default export means
while importing the component in app.js file you can give it any name like:
import MyComponent from "./components/Greet"
and return the same in div tag from render method of class component in app.js file
as:

```
class App extends Component{
        render() {
          return {
                <div className = "App">
                <MyComponent />
                </div>
          }
        }
}
```

Named Export:
eg; export const Greet = () => <h1>Hello Nandini</h1>
but importing as:
import {Greet} from "./components/Greet";

Class Components
        are basically ES6 classes

Similar to functional component a class component also can optionally receive properties (props) as input and return HTML(JSX).
Apart from props, a class component can also maintain private internal state and in simpler words it can maintain some information which is private to that component and use that information to describe the user interface.

for class component:
import React, {Component} from 'react'

(download some ES6 snippets to use imr, imrc short cuts to import react and these things to our components)

Comparing Functional and Class Components

Functional components are simple function receiving props and returning a declaration.
1st Advantage of using functional component is Absence of "this" keyword (this

keyword is difficult for beginners -- so functional component didn't use it)
2nd Advantage is you will be forced to think of a solution withour having to use state (if you have a number of components and each with their own private state, maintenance and debugging your application is kind of difficult, Functional component tend to be without any complicated logic and are mainly responsible for the user interface that's why functional components also known as *stateless components*, *dumb components* or *presentational components*)

Class component (CC) are More feature rich
CC can maintain their own private data (as state)
CC can contain Complex UI logic and most importantly they provide lifecycle hooks.
CC also known as *stateful components*, *smart components* or *container components*

New Feature with new React 16.7.0-alpha

This new feature which kind of contradicts what we've learnt about functional versus state components.

so from this version onwards (2018 React Conference) without writing a class you can use state and Hooks are a new feature proposal that let you use state and other React features without writing a class.

(so as we discussed previously functional components are stateless, we'll take that statement back)

so state and hooks which were exclusive to class components can now be used in Functional Components as well.

Hooks

No breaking changes.
Completely opt-in & 100% backward-compatible.
so now Component Types- Functional Components and Class Components

So we'ld understand, Using state, lifecycle methods and 'this' binding.

JSX

JavaScript XML (JSX) - Extension to the JavaScript language syntax.
Write XML-like code for elements and components.
JSX tags have a tag name, attributes, and children.
JSX is not a necessity to write React applications.
JSX makes your react code simpler and elgant.
JSX ultimately transpiles to pure JavaScript which is understood by the browsers.

---

example:
JSX version of Hello Component:
import React from 'react';

```
const Hello = () => {
    return (
        <div>
            <h1>Hello Nandini!</h1>
        </div>
    )
}

export default Hello
```

_____

JS version of Hello Component:
React provides a method name as createElement (receives 3 parameters, first parameter is the tag, second is optional parameters and third is string.
example:

```
import React from 'react';

const Hello = () => {
    return React.createElement("div", null, React.createElement("h1", null, 'Hello Nandini!!'))
}

export default Hello;
```

Second parameter of createElement method is basically an object of key-value pair that will be applied to the element.

```
  return React.createElement(
        'div',
    {id: 'helo', className: 'dummyClass'},  //as in JS class is a reserved keyword so we use ClassName instead of class
    React.createElement('h1', null, 'Hello Nandini!!!')
    )
```

Basically each JSX element is just syntactic trigger for calling React.createElement and that is why importing React library is mandatory.
JSX translates into React.createElement which in turn uses the React library.

JSX differences
Class replaced by className
for replaced by htmlFor
camelCase property naming convention
        onclick -> onClick
        tabindex -> tabIndex

Follow for New updates:
(https://github.com/facebook/react/issues/13525)

_____

Props

____

Components are reusable
example: reusing Greet Component to greet multi people

intention is to pass name from app component to Greet component and render that name in the browser.
to specify props for a component we specify them as attributes.

1st thing is pass props to component
2nd thing is use 'this' parameter in the function body.

eg:

```
export const Greet = (props) => {
console.log(props);
return <h1>Hello {props.name}</h1>  //curly braces (feature of JSX)to evaulate JSX expressions.
}
```

eg:
app.js file:

```
function App() {
  return (
    <div className="App">
      <Greet name="Bruce Lee" profession="Martial Artist and Actor"/>
      <Greet name="Taylor Swift" profession="American Singer and writer"/>
      <Greet name="Ariana Grande" profession="Singer"/>
      {/* <MyGreetComponent /> */}
      <Welcome/>
      <Hello/>
    </div>
  );
}
```

Greet component:

```
export const Greet = (props) => {
console.log(props);
return <h1>Hello {props.name}, you're great {props.profession} </h1>
}
```

_____

if you have to pass dynamic HTML content, pass in between component tags and in component defintion simply render the content using props.children.

```
<Greet name="Bruce Lee" profession="Martial Artist and Actor"><p>This is children props</p> </Greet>
```

as we can return only one HTML element from the component
so enclosing both in div tag as:

```
return (
    <div>
        <h1>Hello {props.name}, you're great {props.profession} </h1>
        {props.children}
    </div>
 )
```

_____

in Class Component , unlike the functional component where we specify the props parameter, in Class Component the properties are available through this.props which is reserved in Class Components.

props are immutable (means there value can't be changes).

Component State in React

props vs state

props get passed to the component, while state is managed within the component.
Function paramters vs Variables declared in the function body.
props are immutable , component has full control on the change of state.
in Functional components props can be accessed with using props paramter and in class component props can be accessed using this.props.

props - Functional Components
this.props - Class Components

useState Hook - Functional Components
this.state - Class Components

(props and state holds information that influences UI in the browser)

Now, How state can be used in Class Component?

EXAMPLE scenario: we want a button, subscribe button and when click on the button the text should change to Welcome Visitor! Thank you for subscribing.

as props are immutable so passing this as property won't work here.
the solution is to use component state.
1st step is to create state object and initialize it and this step is usually done inside the class constructor.

```
eg: class Message extends Component {

        constructor(){
                super()
                this.state= {
                        message: "Welcome Visitor"
        }
        }

//changeMessage method we're calling to alter the state of the component.
        changeMessage(){
                this.setState({
                        message: "Thank you for subscribing"
                })
        }
```

2nd step is to bind this state object into render method:

```
        render(){
                return(
                        <div>
                                <h1> this.state.message </h1>
                                <button onClick = {() =>
        this.changeMessage()}> Subscribe </button>
                        </div>
                )
```

So, state is basically an object that is privately maintains inside the component. A state can influence what is rendering inside the browser.State can be changes with in the component.

_____

Extensions: ES7+ React/Redux/React-Native snippets

We'll learn...dos and don'ts with state object:

shortcuts:
https://github.com/r5n-dev/vscode-react-javascript-snippets/blob/HEAD/docs/Snippets.md

rce
for:

```
import React, { Component } from 'react'

export class FileName extends Component {
  render() {
    return <div>$2</div>
  }
}

export default $1
```

_____

Destructuring props and state

ES6 feature that makes it possible to unpack values from Array or properties from object into distinct variables.

Destructuring in React improves code readability.

__> 2 Ways to destructure props in a functional component.
_____1st way is to destructure it in functional paramter itself.

eg:

```
export const Greet = ({name, profession}) => {

    return (
        <div>
```

```
            <h1>Hello {name}, you're great {profession} </h1>

    </div>
     ) //curly braces to evaulate JSX expressions.
    }

______ 2nd Way is to destructure it in fucnction body.
eg: // Destructuring in function body
export const Greet = props => {
    const {name, profession} = props
    return (
        <div>
            <h1>
                Hello {name} a.k.a {profession}
            </h1>
        </div>
    )
}

__> Destructuring in Class Component.
In class component we tend to destructure the props and state in render method.
eg: // /Destructuring in Class Component
    class Welcome extends Component{
        render(){
            const {name, profession} = this.props
//for state:  const {state1, state2} = this.state
            return <h1>
                Welcome {name} tumtoh {profession}
            </h1>
        }
    }
```

EVENT HANDLING

creating functional component using React Snippet rfce
In React events are named using camelCaseConvention.
        eg: onClick

common mistake:
        in onClick event we pass the function as event handler, no parenthesis, with parenthesis it'll become function call (that we don't want).

We want handler to handle function, not a function call.

If we leave the parenthesis, it won't work.

In class component the things worsed as the click handler changes the state

-> Event Handling in Functional Component
eg:

```
import React from 'react'

function FunctionClick() {

  function clickHandler(){
    console.log('Button Clicked');
  }

  return (
    <div>
        <button onClick = {clickHandler}>
            Click Here
        </button>
    </div>
  )
}

export default FunctionClick
```

_____

-> Event Handling in Class Component:

```
import React, { Component } from 'react'

export class ClassClick extends Component {

    clickHandler() {
        console.log('You clicked me');
    }

  render() {
    return (
      <div>
        <button onClick ={this.clickHandler}>
            ClickMe
        </button>
      </div>
    )
  }
}

export default ClassClick
```

_____

HOW TO BIND EVENT HANDLERS

-> Why to bind event handlers in React?
        it's bcz of how this keyword works in JS, not bcz of how React works.
(Read Basics of *this* keyword: https://www.w3schools.com/js/js_this.asp)

this keyword is undefined in Event Handlers.That's where binding came in picture.

1st way:-> To use bind keyword (binding in render method) (THIS APPROACH HAS PERFORMANCE IMPLICATIONS, so probably you don't wanna use it).

eg: <button onClick={this.clickHandler.bind(this)}>
Click
</button>
(this approach can work for small applications as everytime a new event handler is generating, so in large applications where nested child components are there ..it would be create issues)

2nd way:-> To use arrow functions in Render Method
(easiest way to pass parameter)
<button onClick={() => this.clickHandler()}>
ClickUsingArrowFunBind
</button>

//React Documentation suggests either approach no. 3 or 4. (3RD APPROACH BEST)
3rd way:-> binding the event handler in constructor instead of binding in Render Method.

```
  constructor(props) {
  super(props)

  this.state = {
     message: "Hello"
  }

//   Binding the *this* keyword in Event Handler inside constructor - 3rd approach of binding mentioned in official documentation of React.
  this.clickHandler = this.clickHandler.bind(this)
}
```

//React Documentation suggests either approach no. 3 or 4.
4th way:-> (CLASS PROPERTY AS ARROW FUNCTIONS) to use an arrow function as a class property.(basically change the way you define method in the class)

```
  clickHandler = () => {
  this.setState({
      message: "GoodBye Friends!"
  })
}
```

Video15: Methods as Props

(In earlier videos we've learned how a parent component can pass props to children component.)

WHAT if a CHILD component want to communicate it's parent component?
-> we'll use props