# 8289 - DBS Word Doc Letterhead FA_CMYK

**REACT JS ASSIGNMENT**

**Amisha Kumari**

**1] how to filter the array data in javascript react**

**->** const names = ['Eshan', 'Shawn', 'Ariana', 'Maluma', 'Guru'];

function App() {

return (

<div>

{names.filter(name => name.includes('S')).map(filteredName => (

<li>

{filteredName}

</li>

))}

</div>

);

}

**2] how get parent element javascript**

**->** ParentID = node.parentNode;

**3] add all elements in array javascript**

**->** function arraySum(a){

var total=0;

for(var i in a) {

total += a[i];

}

return total;

}

var arr = [123,155,134, 205, 105];

var Total = arraySum(arr);

**4] add an element to an array javascript**

**->** arr = [1, 2, 3];

arr.push(4);

**5] add 10 seconds to date javascript**

**->** Date.setSeconds(date.getSeconds() + 10);

**6] add 10px to width js**

**->** var img = document.getElementById("image");

var currWidth = img.clientWidth;

if(currWidth < 500){

img.style.width = (currWidth + 10) + "px";

}

**7] add 2 class names react**

**->** //To specify CSS class

var classes= “class1 class2”;

<li className={classes}></li>

**8] add 2 for hours in date timestamp js**

**->** var date = new Date();

date.setHours(date.getHours + 2);

**9] add 7 days in date using jquery**

**->** var date = new Date();

date.setDate(date.getDate() + 7);

**10] add a route to a buttoin in angular**

**->** <Link to=”/register”>

<button>

Click here

</button>

</Link>

**11] add a slash to string in javascript**

**->** str = “amisha”;

function addSlash(str){  
 str = str.substring(0,2) + ‘/’ + str.subString(2,6);

return str; }

**12] how to run a filter having object [] in reactjs**

**->** <div>

{arrayOfObject.filter(el1 => el1.age<18).map(teens=> teens.name)}

</div>

**13] how to use filter when returning jsx**

**->** function fil(){

Return (

{arrayOfObject.filter(el1 => el1.age<18).map(teens=> teens.name)}

);

**14] Render JSX**

**->** Render() {

<div>//add statements or call function </div> };

**15] How to Use JavaScript in JSX**

**->** Enclose the code in ‘{ }’ to use javaScript in JSX. The curly braces simply tell JSX that it is javaScript expression.

**16] How to Use array in JSX**

-> const arr = dataArray.map((data) => {return ( <div> {data}</div>); } );

**17] How to define a component**

**->** export default class App extends React.Component{  
 render() {

return ( <h1>Hey there!</h1>);

} }

**18] What is the use of this.props.children**

**->** It is used to render the content in the component. Means it simply display whatever included b/w

the opening and closing tags, when invoking a component.

**19] Finding a DOM node**

**->** Refs provide a way to access DOM node or react element created in the render method.

However, it is not advisable to use refs too often. But still if we want to use refs then use

the callback patterns.

**20] Write example to use this.state**

**->** The state is built-in object in React components. In the state object we store property values that

belongs to the component.

EXAMPLE:-

import React from 'react';

class App extends React.Component {

constructor(props) {

super(props);

this.state = {

title: "React state example",

};

}

**21] Component Lifecycle**

**->** Lifecycle of Components Each component in React has a lifecycle which you can monitor and

manipulate during its three main phases. The three phases are:

1. Mounting
2. Updating
3. Unmounting

//EXAMPLE

|  |
| --- |
| import React from 'react'; |
|  | class App extends React.Component { |
|  | constructor(props) { |
|  | super(props); |
|  |  |
|  | this.state = { |
|  | data: 0 |
|  | } |
|  | this.setNewNumber = this.setNewNumber.bind(this) |
|  | }; |
|  | setNewNumber() { |
|  | this.setState({data: this.state.data + 1}) |
|  | } |
|  | render() { |
|  | return ( |
|  | <div> |
|  | <button onClick = {this.setNewNumber}>INCREMENT</button> |
|  | <Content myNumber = {this.state.data}></Content> |
|  | </div> |
|  | ); |
|  | } |
|  | } |
|  | class Content extends React.Component { |
|  | componentWillMount() { |
|  | console.log('Component WILL MOUNT!') |
|  | } |
|  | componentDidMount() { |
|  | console.log('Component DID MOUNT!') |
|  | } |
|  | componentWillReceiveProps(newProps) { |
|  | console.log('Component WILL RECIEVE PROPS!') |
|  | } |
|  | shouldComponentUpdate(newProps, newState) { |
|  | return true; |
|  | } |
|  | componentWillUpdate(nextProps, nextState) { |
|  | console.log('Component WILL UPDATE!'); |
|  | } |
|  | componentDidUpdate(prevProps, prevState) { |
|  | console.log('Component DID UPDATE!') |
|  | } |
|  | componentWillUnmount() { |
|  | console.log('Component WILL UNMOUNT!') |
|  | } |
|  |  |
|  | render() { |
|  | return ( |
|  | <div> |
|  | <h3>{this.props.myNumber}</h3> |
|  | </div> |
|  | ); |
|  | } |
|  | } |
|  | export default App; |