Friday, August 19, 2022 9:07 AM

Different Parts of UI

HTML - widgets

To enhance HTML --- CSS

1. inline , internal , external

Inline

<tag style="" > </tag>

Internal

<head>

<style>

selectors

</style>

</head>

<tag class="" or id ="" > </tag>

How to give selectors ? Tag based,id based , class based

In the selectors we give CSS attributes

External CSS

.outer

{

}

How to include the CSS in html <link >

Bootstrap

Javascript ------

In html = we can write code in the <script>

DOM manipulation

Identify a particular TAG in html =

document.getElementById("someId")

READ value of that tag

OR

We can change the body /value of that TAG

let x = document.getElementById("someId").value

WRITING

document.getElementById("someId").innerHTML = provide something here

<script>

St1

St2

</script>

Instead of putting in script directly we use function

<script>

```
 function(){
      St1
      St2
      }
</script>
```

Event Handling in Javascript --- what events ----

ES6 standard for JS   syntax ----

using let or var or const

```
if ( x ==  or  === "" )
else
```

for
while
do while
switch case

Different data types ----

number, boolean ,

string ---- string functions

splice, trim, toUpperCase, toLowercase , concat , substring, charAt

Strings are immutable !!!

MDN documentation

arr ------- array functions

push , pop, insert,

sort ((a,b)=>{ ....   ; return +ve, -ve or 0})

filter((ele)=>{ … ; return boolean})

map ((ele)=>{..... ; return the new element}   )

foreach ((ele) =>{ print ele})

Slice ---   -ve index

functions

1. passing a function to another function
2. Return a function and call the returned function
3. Arrow functions , named function, anonymous function

Destructuring of ARRAY , Object

```
functional react components

  function Com()
  {
        let [x,setX] = useState(initial value)
  }
```

Spread operator

… arr

Deep copy of objects

Rest parameter

```
function f1(… nums)   //variable number of parameter
{
}
```

Nums must be the last parameter
One function can have only 1 rest parameter

Prototype pattern ---- every javascript object has a prototype
The programmer can split state and functions
State = current object
Functions = prototype
Advantage = memory saved ---- functions are shared between all objects

---

Date --- how to create date
How to get a date from HTML and use it to create a date object
once you get a date object
Use it to get day, month, year

Promises ------ delayed execution [main Stack , callback queue, promise queue]
Promises may be resolved or rejected
how do we get the resolved value ---- then((resolved)=>{ use the value }
how do we get the rejected value ---- catch

```
new Promise().then( (resolved )=>{ return val}   ) . then(  (val)=>{}  )
```

Async callbacks ----
Difference between

simple function	Async function
returns whatever you return OR undefined	returns Promise
cannot call await	we can call await

A **library** on top of Vanilla Javascript = JQUERY

1. How to include jquery in my html

```
<script src=""    />
```

2. How does the Jquery code begin

```
<script>
    $(document).ready( ()=>{

    //registering the callbacks for events
    $("#b1").click(  ()=>{  //what to do}   )
```

```
        })
</script>

<body>

        <button id="b1"> </button>
</body>
```

	Javascript	Jquery	React	
Get element from HTML	document.getElementById("id")	Selector	onChange={(event)=> { setX(event.target.valu e) }}	
Event handling	onClick="handler() "	Register callback $("#b1").click(()= >{ //what to do})	onClick ={ handler } Register callback	
How to set the html in tag	document.getElementById("id").innerHTML =" Hi "	$("#id").html(" hi ")	onChange={(event)=> { setJSXVar(hi) }} < div> { jsxvar} 	
AJAX	XMLHttpRequest	$.ajax(	fetch( )	
give them URL + DATA				
How to process response				

React ----------

Component

How to create component and Render it

index.js

App.js

MyCompF

MyCompC

Important --- component names should start with capital , exported properly

Function	Class

return JSX	render() { return jsx }

C1 wants to pass data to C2 } props

How to pass

c1	return (<c2 n={here you pass} />)

How to receive

C2	function C2(props)
	{
	let v = props.n
	}

	class C2
	render()
	{
	let v = this.props.n
	}

If some data changes then can we see the change on the HTML

state

How to declare state ?	let [x,setX] =useState	this.state={ fn:'',ln:''}
How to change state ?	setX(.....)	this.setState({ fn: newval})
How to use state ?	{x}	{this.state.fn}

Event handling ----- handler should be arrow function in a class component

Each handler will get event object

Text field	event.target.value
Check box	event.target.value , event.target.checked
radio button	event.target.value == value of the selected one
drop down select	event.target.value

Life cycle

Class	function
componentDidMount	useEffect (()=>{.....} ,[ ])
componentDidUpdate	useEffect (()=>{.....} ,[props.message])

	useEffect (()=>{……..} ,[x])
componentWillUnMount	useEffect (()=>{ return ()=>{} } ,[ ])

Renderering Lists/tables

```
cities= [  pune, mumbai,kolhapur, merut, jhansi ,…..  ]
Let [jsxoptions ,setJ]  = useState()
Handler()
{
    let temp =   cities.map( (city)=>{
                    return  <option value={city}  key={"city"}>{city} </option>
                }
     )

    setJ(temp)
}


 return(
    <select> {  jsxoptions } </select>
)
```

Server Side ----------

express !!! Web server

Listen

dynamic html = hbs

rest api =

How to share data between Redux components

1. Props ------- number, obj , string === Parent to Child
2. Props ---------- function = Lifting state UP = Child to Parent
3. Redux Store --------- communicate between any components ---may be parent child/may be siblings
    1. reducer

        State , action

        Reducer is a function that returns the state

    2. using the reducer we create a store !!!

    3. make the store available to components

        WRAP the components in <Provider store={exported store}> <App></App> </Provider>

    4. C1

        useSelector() to access current state

        useDispatch() to get the dispatch(pass action here)

Routes

--- install the react-router-dom

1. Define Routes ---- MAP the path <--------->component
2. Use the routes in the <Link to="path" > …..
3. Place the <Outlet> tag where you want to render the component
4. Passing parameters to the link
    a. Accessing link parameter using useParams hook
