7AM 12-07-2023

SrinivasRao

60 hours (7 AM to 8 AM - MON to SAT)

HTML
====

1. Why do we go for HTML?

2. Inline vs Block-level elements

3. Attributes

4. Input controls

5. Table

6. List

7. Developer console window

8. Elements Behaviour

CSS
===

1. Why do we go for CSS

2. How many ways can we apply CSS ?

3. Selectors

4. Media queries

5. Positions

6. Box model

7. Bootstrap

JavaScript:
=======

1. Introduction

2. Variables

3. Functions

4. Array Methods

5. String Methods

6. JSON Object

7. Get Reference

8. Apply Styles

9. Events

10. Debouncing

11. Throttling

12. Async vs defer

13. OOPS

14. Set Timeout

15. Set Interval

16. Browser Storages

17. BOM

18. Call, apply, bind

19. Closure

20. Event Bubbling

21. Event Capturing

22. Exception Handling

23. Callbacks

24. Promises

25. Async Await

26. AJAX

27. Async functions

28. Arrow functions

29. Spread operator vs Rest Parameter

30. Array and Object destructuring.

31. Event Looping

32. Generators

33. Regular Expressions

34. var, let , const

35. Variable and Method Hoisting.

React:
====

1. Why do we go for React ?

2. How to prepare the templates in react (functional components) and JSX

3. How to handle the events(click, focus, blur ...) ?

4. How to take the data from inputControls(textbox,password,radio,checkbox,textarea,fileupload,dropdown)?

5. State vs Props

6. State value update, prop value update

7. Communication b/w components (parent-child , child-parent)

8. Context API

9. How to handle AJAX(XMLHttpRequest, fetch, Axios)

10. Lifecycle phases

11. Lists and keys

12. Reusable components

13. How to apply styles dynamically (inline, class)

14. Handle Routing in React

15. Conditional Rendering

16. Integrate bootstrap, react-bootstrap, material UI design

17. Module CSS, styled-components, SASS.

18. DefaultProps , PropTypes

19. Higher Order Components

20. React Testing(jest+RTL)

21. Cypress(JavaScript Automation Testing)

22. Create and publish your own lib in npmjs.

React Hooks:
=========

1. useState

2. useEffect

3. useRef

4. useContext

5. useReducer

6. useMemo

7. useCallback

8. useTransistion

9. Custom Hook

Build and Deploy:
==========

Godaddy

AWS

Vercel

Version Control:
==========

GIT

Backend:
======

1. node js

DataBase:
=======

1. MongoDB

Redux(state management at application level):
==========

1. How to implement Redux in your React Application ?

2. actions, reducers, store

Data Visualization:
==========

1. Visualization(line chart)

EtoE Application(Online Test) :
==========

1. client(React +Redux) + Server(node with express) + MongoDB

GraphQL:
======

1. client(React with GraphQL) + Server(node with Graphql) + MongoDB

JWT:
====

1. client(React) + Server(node with express)

Others:
=====

1. Next JS --

2. React Native --

3. React With Typescript --

4. Webpack

Softwares need to install:
==========

1. Visual Studio Code : https://code.visualstudio.com/

2. git : https://git-scm.com/downloads

3. DesktopGit : https://desktop.github.com/

4. MongoDB : https://www.mongodb.com/try/download/community

5. MongoDB compass : https://www.mongodb.com/products/compass

6. Node s/w or nvm : https://nodejs.org/en/download or https://github.com/coreybutler/nvm-windows/releases

7. postman : https://www.postman.com/downloads/

Create Accounts:
==========

1. https://github.com/

2. https://www.vercel.com/

3. https://mongodb.com/

4. https://aws.amazon.com/

5. https://www.npmjs.com/

Learning Websites:
==========

1. www.uijavakit.com

2. www.kalamschools.com

3. www.yourpost.in

4. www.writetest.in

Email:
====

chenchala.srinu@gmail.com

14-07-2023

Fee 5000 with videos 8000

NOTE PLEASE ATTEND CLASS REGULAR

THANK YOU NARESH IT TEAM SYAM NIT ADMIN 8247601157 SEND MESSAGE DONT

CALL PLEASE REQ THANK YOU

This is Not For Full stack Java. If you any doubts contact 8179191999

Day-1 https://youtu.be/C5Ems01BLYc
Day-2 https://youtu.be/KXWUo81wFv0
Day-3 https://youtu.be/hJf8C0z8WHU
Day-4 https://youtu.be/lXmhRFqAbqs
Day-5 https://youtu.be/tkucYE8YpVY
Day-6 https://youtu.be/dFJ8IF3ydLo
Day-7 https://youtu.be/oFzBAvmywrM
Day-8 https://youtu.be/9-JvuFEQdbQ

15-07-2023

What is React - React is an open-source client-side library , developed by a Facebook organization.

Why do we go for React -

1. Using React we can develop UI(interacts directly with end-user)

2. As part of web application(static, dynamic) development, we can develop client-side applications usingn React.

HTML - Hyper Text Markup Language

Programming Language VS Markup Language

Programming Language - hello.c - write source code - compile - run - output

Markup Langualr - hello.html- write source code - run - output

Open Developer console window - right-click on window(context menu) - select inspect option

1. Elements

2. console

3. source

4. network

5. application

1. source code (.html) - developer console window - element tab- body

2. body(source code) - rendered inside the browser - visible to the end user

17-07-2023

1. Using HTML we can develop static web pages which are running inside the browser
2. HTML elements divided into 2 sections
3. inline elements - which takes content width as it is own width - span, b, i
4. block-level elements - which take the parent width as it is own width - div, p, h1, h5, h6

18-07-2023

1. Elements behavior completely depends on the styles which are applied to an element.
2. We can change the block-level element to an inline element, and an inline element to a block level.
3. We can change the behavior of span like h1, by applying h1 style properties to span.

How do we see the content(loaded in the browser) related code in Browser ?

19-07-2023

1. Terminalogy- Developer console window, context Menu
2. Element Tab- contains , contenet related source code and styles applied to the elements
3. Console Tab- To log messages in the console while running the application, To see the error/info/warning , Use like JavaScript working environment.
4. Network Tab- Can track browser-sent requests and responses
5. Source Tab- Can see the successful request-related responses in the file explorer format, and use it for JavaScript code debugging purposes.
6. Application Tab- Can view the browser storage physically like local storage, session storage, and cookies.

20-07-2023

1. In HTML, the attribute is the key, value pair.
2. Attributes placed in the open tag.
3. Can place any no of attributes, differentiate with space.
4. Attributes tell something about an element.

21-07-2023

1. Complete all the HTML topics from uijavakit.com
2. Why <input type="password" > textbox show the content with the mask ? because -Webkit-text-security: disc style property applied.
3. How many ways can apply CSS ? - inline, internal, external
4. How we can apply inline styling ? - using the style attribute

22-07-2023

1. static

a. it is the default position.
b. we are not able to apply t,r,l,b

2. absolute

a. we can apply t,r,l,b
b. element takes content width.(block to inline)
c. it removes the space from DOM
d. it takes parent as a reference(if my parent has other that static)

3. relative
a. we can apply t,r,l,b
b. no change in the element width(block-block, inline-inline)
c. it not removes the space from DOM
d. it takes my current position as reference.

4. fixed
a. we can apply t,r,l,b
b.element takes content width.(block to inline)
c. it removes the space from DOM
d. it takes viewport/widnow/html as a reference
e. irrespecitve of scroll, element always fixed to that particular position

5. sticky
a. we can apply t,r,l,b
b.no change in the element width(block-block, inline-inline)
c. it not removes the space from DOM.

27-07-2023

1. look and feel
2. performance
3. responsive
4. resolution - width of the device

<h1>Sachin</h1>

```
h1{
    color:"black"
}
```

<700 - h1 -red

700-900 - h1 -blue

900-1200 - h1 - yellow

>1200 - h1 -green

media queries - based on the resolution we can apply CSS properties.

title tag vs title attribute

style tag vs style attribute

favicon

deployment

ctrl+shift+n - incognito mode open browser

28-07-2023

1. HTML -
2. CSS -
3. Java Script -
4. contextMenu
5. debugging - source tab

Variables-

JS:

variable name=initial value
amount=100 -valid
amount="Sachin"-valid
name=“sachin” -valid

dataType varaibelName=initialValue

var/let/const amount=100

error msg- amount is not defined

fix - amount=1000

error msg - loc is not defined
fix:
loc="Hyd"
var/let/const loc="Mumbai"
var loc;

Default value for the variable - undefined

Dynamically Typed Programming Language

Java:

dataType varaibelName=initialValue

int amount=100
int marks="Sachin" - Invalid

Functions

29-07-2023

1. in console age - enter - age is not defined - fix age=10 / var age / var age=30
2. default value for the variable - undefined
3. Do we have data types(int,string) in js ? no
4. Dynamically typed programming language

Functions

```
function(keyword) <function-name>(){
  // function body
}

function add(){
   var n1=10;
   var n2=20;
   var sum=n1+n2;
   console.log(sum);
}

add -  f add(){}
add()-30

a() - a is not defined
var a=10;
a() - a is not a function

1st person
add()

2nd person
add()

3rd person
add()
```

31-07-2023

```
a() - a is not defined
var a=10
a() - a is not a function
======================
function add(n1,n2){
   var sum=n1+n2
   console.log(sum)
}
person 1:
add(1,2) // 3
person 2:
add(100,200) // 300
====================
function fn(a){
a()
}
fn() - a is not a function
fn(10) - a is not a function
====================
function f1(a){
a()
}
function f2(){
console.log("f2 called")
}
```

```
f1(f2)// f2 called
f2 - callback
f1- Higher Order Function
======================
function fn(){
//5k lines of code
var n1=10
var n2=20
var sum=n1+n2
console.log(sum)
}
fn() = 5004 lines code executed // 30
fn() = 5004 lines code executed //30
fn() = 5004 lines code executed //30

function fn(){
//5k lines of code
var n1=10
var n2=20
var sum=n1+n2
return sum
}
var result=fn() // 5004 lines code executed
console.log(result)
console.log(result)
console.log(result)
================================
function(){
}
1. var a=function(){
}
a()

2. (function(){
   })()

3. function fn(a){
    a()
    }
fn(function(){
  })
========================
function fn(){
  return function(){
    console.log("called return function")
  }
}
var a=fn()
a() // called return function
======================

function fn(a,b){
    a("Sachin");
    b();
```

```
    return function(){
        alert("Kohli")
    }
}
```
while calling fn output should be Sachin in the alert box and then Dhoni in the alert box and then Kohli in the alert box

fn- function
arguments - 2
type of arguments - functions
return type- function
return value-function(){
alert("Kohli")
}

```
var x=fn(
 function(data){
    alert(data)
 },
function(){
  alert("dhoni")
}
)
x()
```

or

```
fn(
 function(data){
    alert(data)
 },
function(){
  alert("dhoni")
}
)()
```

==========

02-08-2023

Hellow World
react-lib-facebook-opensorce- client side appliction - browser
window-super object
DOM - Document Object Modal
using html - no
using js - no
using react ?
How to prepare content ?
using components
1. functional components

```
function A(){

}
```

A() = A is a function - no

new A() = A is acting like a class -no
<A></A> = A is action like a component
<A />

2. class components

a. what is lib ?
it is js file - react.js - created by some other person (facebook)
contains - variables and methods
load that lib
use it(variable,methods)

03-08-2023

```
function App(){
   return  <h1>Hellow World</h1>
}
```

App() - js function
new App() - class

<App></App>
<App />

<h1>Hellow World</h1>
JSX - JavaScript And XML (HTML + XML + JS)

HTML - no compile
JSX - compile
https://react.dev/

print your name 5 times
display 4 boxes

https://gist.githubusercontent.com/gaearon/0275b1e1518599bbeafcde4722e79ed1/raw/db72dcbf3384ee1708c4a07d3
https://web.proctur.com/

04-08-2023

1. min lib - react, reactDOM, babel
2. react- reconginzed components
3. reactDOM-render content inside the container
4. babel- compile JSX
5. JSX - JavaScirpt and XML
6. JSX - HTML + XML + JS

Warning: The tag <app> is unrecognized in this browser
ReferenceError: App is not defined
Error: createRoot(...): Target container is not a DOM element.
SyntaxError: Unexpected token "<"
TypeError: ReactDOM.createroot is not a function
TypeError: Cannot read properties of undefined (reading
"__SECRET_INTERNALS_DO_NOT_USE_OR_YOU_WILL_BE_FIRED")

Write some Java Script error messages?

05-08-2023

var amout=10
var name="sachin"
var isHePass=true
var players=["SAchin","Dhoni"]
var obj={}
properties- 0 - key,value pair
Insert:

key	value
name	Sachin

1. dot notation - obj.name="Sachin"
2. [] notation - obj["loc"]="Mumbai"
obj
{name: "sachin", loc: "Mumbai"}

obj.full name="Sachin Ramesh Tendulkar" - Uncaught SyntaxError
obj["full name"]="Sachin Ramesh Tendulkar"
obj
{name: “sachin”, loc: "Mumbai", full name: "Sachin Ramesh Tendulkar"}

Updation:
obj.name="Dhoni"
Delete
delete obj.name
retrieve
obj.loc
obj["loc"]

Assignment

My name is Sachin, am from Mumbai.
My name is Dhoni, am from Ranchi.
My name is Kohli, am from Delhi.

07-08-2023

JSX - HTML + XML + {JS}
JSX - compile - babel
lib - 3 - react , reactDOM, babel
When we can pass the data to the component ? while loading
How many ways we can pass the data to the component while loading - 2ways - as an attribute , as a children

Hi My self s1, rno is 1, marks are 500.
Hi My self s2, rno is 2, marks are 400.
Hi My self s3, rno is 3, marks are 300.

1. create a template using react
2. handle click event on button
3. get the data from input controls
4. do some operation
5. render content dynamically
6. Apply Styles dynamically

08-08-2023

When we can pass the data to the component ? while loading <App/> <A></A>
How many ways - 2 ways -> <App name="sachin" /> <A name="Sachin" >Mubai</A>

```
function A(props){
  props.name
  props.children
}
```

functional components - before 16.8 - dumb , state less

```
function App(){
 return <div>Sachin</div>
}
```

class component   - before 16.8

```
class App extends React.Component{  // state full component
    render(){
     return <div>
           <p>
             <b>No1:</b><input />
           </p>
            <p>
             <b>No2:</b><input />
            </p>
           <p>
             <button>Sum</button>
           </p>
           <h1>5</h1>
         </div>

   }
}
```

16.8 - hooks + functional components
from 16.8 version functional component === class components

09-08-2023

useState, useRef, useEffect, useMemo , useCallback , useContext, useReducer, useTransistion
All JavaScript events we can use in react.
js - onclick
react-onClick

in JS the listener - onclick="fnPrintName()"
in react the listene - onClick={fnPrintName}
in React while calling listener function how many arguments it is passed

```
fuPrintName(eve){
}
```

convert into number - Number(), parseInt()

10-08-2023

```
function fn(){
console.log("fn called")
}

const fn=()=>{
}

function(){
console.log(" called")
}

()=>{
console.log(" called")
}

function fn(){
console.log("fn called")
}
const fn=()=> console.log("fn called")

function fn(){
return 10
}
const fn=()=>10


var name="sachin"
var amount=100
================
function fn(){
var loc="Mumbai"
}
fn()
console.log(loc) // loc is not defined
================

function fn(){
loc="Mumbai"
}
fn()
console.log(loc) // Mumbai
================
function fn(){
 var no=10
 if(no==10){
  let loc="Mumba";
}
console.log(loc);
}
fn() // loc is not defined
====================
function fn(){
 let no=10
 if(no==10){
```

```
  var loc="Mumbai";
}
console.log(loc);
}
fn() //Mumbai
====================
function fn(){
 let no=10
 if(no==10){
  const loc="Mumbai";
}
console.log(loc);
}
fn() //Mumbai
====================
function fn(){
 const loc="Ranchi"
 if(true){
   loc="Mumbai";
}
console.log(loc);
}
fn()// Assignment to constant variable.
====================
function fn(){
 console.log(this)
}
fn() // window
==========
function fn(a,b){
 console.log(a+b+this.no)
}
fn(10,20) // NaN
=============
function fn(){
  console.log("fn called")
}
fn()
fn.call()
fn.apply()
fn.bind()()
====================
```

11-08-2023

```
function fn(){
console.log("fn")
}
fn() //fn
fn.call()//fn
fn.apply()//fn
fn.bind()()//fn
=============
```

```
function fn(x,y){
 console.log(x+y+this.i)
}
fn(10,20) // NaN
==============
function fn(x,y){
 console.log(x+y+this.i)
}
var obj={i:100}

fn.call(obj,10,20) // 130
===============
function fn(){
console.log(this)
}
fn() // window
fn.call({x:100}) // {x:100}
fn.apply({x:100}) // {x:100}
===============
function fn(x,y){
console.log(x+y+this.i)
}
obj={i:100}
fn-
ƒ fn(x,y){
 console.log(x+y+this.i) // this is window
}
fn.bind(obj,10,20)
ƒ fn(x,y){
 console.log(x+y+this.i) // this is {i:100}
}
fn.bind(obj,10,20)() // 130
============================
function sum(...nos){
  console.log(nos)
}
sum()- []
=======================
function sum(...nos){
  let sum=0;
  for(let i=0;i<nos.length;i++){
        sum=sum+nos[i]
  }
  console.log(sum)
}
================
function fn(...nos,name){
}
fn(10,20,"Sachin") //  SyntaxError: Rest parameter must be last formal parameter
===============
function fn(name,...nos,){
}
fn("Sachin",10,20)
```

name-Sachin
nos-[10,20]
==========
let obj1={n1:10,n2:20}
let obj2={n3:30,n4:40}
let obj3={...obj1,...obj2}
{n1:10,n2:20,n3:30,n4:40}
==========
const obj1={n1:10,n2:20}
const obj2={n3:30,n2:100}
const obj3={...obj1,...obj2} // {n1:10,n3:30,n2:100}
const obj3={...obj2,...obj1} // {n1:10,n3:30,n2:20}
==========
let obj1={n1:10,n2:20}
let obj2={n3:30,n4:40}
var obj3={...obj1,...obj2,n5:50,n6:60} // {n1:10,n2:20,n3:30,n4:40,n5:50,n6:60}
var obj3={...obj1,...obj2,n2:50,n4:60} // {n1:10,n2:50,n3:30,n4:60}
==========
const arr1=[1,2,3]
const arr2=[3,3,3]
const arr3=[...arr1,...arr2] // [1,2,3,3,3,3]
const arr4=[...arr1,...arr2,1,2,3] // [1,2,3,3,3,3,1,2,3]
==========
let x=10
function fn(){
  let y=20
  return function(){
      let z=30
      console.log(x+y+z)
  }
}
fn()() // 60   Laxicolscope

12-08-2023

call, apply, bind
rest parameter
spread operator
lexical scope
==========
function fn(){
console.log(this)
}
fn() // window
fn.call({x:100}) // {x:100}
fn.apply({i:10}) // {i:10}
fn.bind({x:10}) // fn function
fn.bind({x:10})() // {x:10}
==========
const fn(){
 console.log(this)
}
fn() // window

```
fn.call({x:10}) // window
fn.apply({x:10}) // window
fn.bind({x:10}) // fn function
fn.bind({x:10})() // window
because of arrow function having lexical scope behavior
==========================
function fn(){
  console.log("outer fn", this)
   return function(){
       console.log("inner fn", this)
   }
}
const x=fn() // outer fn , window
x // inner function
x() // inner fn, window
x.call({x:1000})  // inner fn, {x:1000}

var x=fn.call({i:10}) // ourter fn, {i:10}
x() // inner fn, window
==============================
function fn(){
  console.log("outer fn", this)
   return ()=>{
       console.log("inner fn", this)
   }
}
const x=fn()  // outer fn, window
x // inner arrow function
x() // inner fn, window
x.call({x:10}) // inner fn, window

var x=fn.call({i:10}) // ourter fn, {i:10}
x() // inner fn, {i:10}
============================
var obj={
   name:"SAchin",
    runs: 2000,
  loc: {
      area : {
      pin:53434,
         town:"Mumbai"
      }
  }
}
const {pin,town}=obj.loc.area
pin // 53434
town// Mumbai
================================

var arr=["sachin",20000,"Mumbia"]
arr[0] // sachin
arr[1] // 20000
arr[2]// Mumabi
```

```
const [a,b,c]=["sachin",20000,"Mumbia"]
a // sachin
b // 20000
c // Mumabi
============================
const a=10
const b=20
const obj={
    a:10,
      b:20
}
obj // {a:10,b:20}
const obj={
    a:a
      b:b
}
obj // {a:10,b:20}
const obj={a,b}
============================
function fn(){
  var a=10;
  var b=20
   return a,b;
}
fn() // 20
function fn(){
  var a=10;
  var b=20
   return [a,b]
}
fn() // [10,20]
function fn(){
  var a=10;
  var b=20
   return {a,b}
}
fn() // {a:10,b:20}
=========================
function useState(init){
  let val=init
  let f=function(newVal){
    val=newVal
  }
 return [val,f]
}

const [marks,setMarks]=useState(80)
=======================
```

19-08-2023

when we can pass the data to the component - while loading
How we can pass - attribute, children

How to pass the data from parent to child component - while loading can pass the data as an attribute, children
How to pass the data from child to parent component - callback
where we need to define the callback - parent compnent
==========
App - A - B - C - D- E
pass the data from App -E
1. throght attribute / childen all hirarichy levels
2. context API
3. redux
==========
created by -me
```
const App=(props)=>{
  return <div>{props.name}</div>
}
```
used by - me
```
<App name="Sachin" /> // Sachin
```
==========
created by - react people
```
const Provider=(props)=>{
  return <div>{props.value}</div>
}
```
used by - me
```
<Provider value="Sachin" /> // Sachin
```
==========

21-08-2023

Why we go for useContext hook ? - To consume the data from context

Why we go for context API ? parent to all childrens
1. create context -
```
const ctx=React.createContext()
```
2. make available to all components
```
<ctx.Provider>
    components...
</ctx.Provider>
```
3. provide data to context
```
<ctx.Provider value={}>
    components...
</ctx.Provider>
```
4. consume data from context
```
const ctxData=useContext(ctx)
```

How many ways we can pass the data from child to parent ?
1. using callbacks
Where do we need to define the callback ?
parent / child -> parent
2. Redux

How may ways we can pass the data from parent to child ?
1. using attributes/children while loading the components
2. context API
3. Redux

useState- manage the state of the variable at the component level
useRef- create one reference
useContext- consume the data from the context
useEffect - To handle lifecycle phases(mounting,updating,unmounting)

==========
```
const App=()=>{
  return <div>Sachin</div>
}
```

<App /> - mounting phase
before
after

- updating phase

- unmounting phase

22-08-2023

1. Mounting -- first time load
Before:
```
const App=()=>{
    console.log("5k lines")
  return <div>nit</div>
}
After:
React.useEffect(()=>{
  console.log("call after content loading completed inside the browser")
})
```
==========
```
React.useEffect(()=>{

})
```
no dependencies: first time load and for every state/prop change , callback method called by useEffect
```
React.useEffect(()=>{

},[])
```

empty dependencies: firsttime load , callback method called by useEffect

2. Updating

```
React.useEffect(()=>{

},[cnt])
```
having dependencies: firsttime load ,for every "cnt" change , callback method called by useEffect
```
React.useEffect(()=>{

},[cnt,name])
```
having dependencies: firsttime load ,for every "cnt" and "name" change , callback method called by useEffect
3. Unmounting

```
React.useEffect(()=>{
          return ()=>{
              console.log(" A is going to unmounting...")
          }
      },[])
```

---

1. using function keyword

```
function Bus(){
    this.fw=2;
    this.bw=4;
}

var obj=new App()
obj.fw
obj["bw"]
```

2. using class keyword

```
class Bus{
   fw=2;
   bw=2;
}
var obj=new App()
obj.fw
obj["bw"]
```

23-08-2023

How many ways can we create the class
- using function keyword

```
function Bus(){
    this.fw=2
    this.bw=4
    this.totalWheels=function(){
        console.log(this.fw+this.bw)
      }
}
var obj=new Bus()
obj.fw
obj["bw"]
obj.totalWheels()
```

- using class keyword

```
class Bus{
    fw=2
    bw=4
    totalWheels(){
          console.log(this.fw+this.bw)
       }
}
var obj=new Bus()
```

```
obj.fw
obj["bw"]
obj.totalWheels()
============================
function Bus(){
    this.fw=2
    this.totalWheels=()=>{
        console.log(this.fw+this.bw)
    }
}
var obj=new Bus()
obj={fw:2,totalWheels:f}
obj.totalWheels() // NaN
Bus.protoType.bw=4
obj.totalWheels()//6
obj.getColor()// obj.getColor is not a function
Bus.prototype.getColor=()=>{
   return "red"
}
obj.getColor() //red
============================
class Bus{
    fw=2
    totalWheels(){
          console.log(this.fw+this.bw)
       }
}
var obj=new Bus()
obj={fw:2,totalWheels:f}
obj.totalWheels() // NaN
Bus.protoType.bw=4
obj.totalWheels()//6
obj.getColor()// obj.getColor is not a function
Bus.prototype.getColor=()=>{
   return "red"
}
obj.getColor() //red
====================
class A{
  bw=4
}
class Bus extends A{
    fw=2

    totalWheels(){
          console.log(this.fw+this.bw)
       }
}
var obj=new Bus()
obj.totalWheels()//6
==================
function A(){
 this.bw=4
```

```
}
class Bus extends A{
    fw=2

    totalWheels(){
          console.log(this.fw+this.bw)
        }
}
var obj=new Bus()
obj.totalWheels()//6
================

var o1={
 n1:10
}

o1.n1 // 10
==============
class A{
  n1=20
}
var o1=new A()
o1.n1 //20
================
```