

Javascript:

Functions:

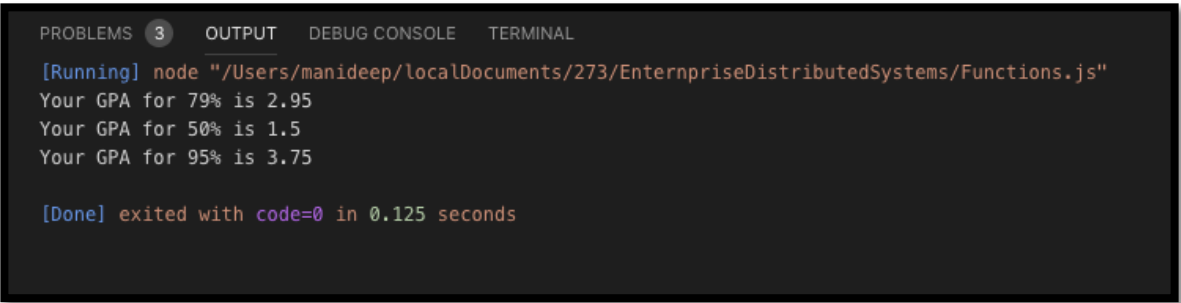
Introduction:

Functions refers to piece of code performing specific functionality it is defined for. The main purpose of defining functions is to avoid code repetition and just invoke the function wherever needed. A function also intakes values known as parameter and returns a value using return statement. If the return statement is not defined, the function will return a default value.

Programming Question: Write a javascript program to calculate gpa from percentage
Code:

```
function percToGPA(percentage) {  
    var GPA;  
    GPA = (percentage / 20) - 1;  
    console.log("Your GPA for "+percentage+"% is " + GPA);  
}  
  
percToGPA(79);  
percToGPA(50);  
percToGPA(95);
```

Output:



```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL  
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/Functions.js"  
Your GPA for 79% is 2.95  
Your GPA for 50% is 1.5  
Your GPA for 95% is 3.75  
  
[Done] exited with code=0 in 0.125 seconds
```

Events:

Introduction: Events are specific action items like say, click, hover, change etc., and javascript here does execute the respective code associated with that html element event.

Programming Question: Using the events, design a Valentine Greetings webpage specific to each gender, responding with appropriate message accordingly.

Code:

```
<!DOCTYPE html>  
<html>  
  
<body>  
    <script>
```

```
function welcomelImage() {
    var image = document.getElementById("wlclImage").src;
    if (image.includes("boy") || image.includes("girl")) {
        document.getElementById("wlclImage").src = "both.gif";
        document.getElementById("wlclImage").style.display = "block";
        document.getElementById("gender").style.display = "none";
        document.getElementById("end").style.display = "block";
    } else if (image.includes("both")) {
        // do nothing
    } else {
        document.getElementById("wlclImage").style.display = "none";
        document.getElementById("gender").style.display = "block";
    }
}
```

```
function messageGif() {
    if (document.getElementById('gender').value == "Queen") {
        document.getElementById("wlclImage").src = "boy.gif";
        document.getElementById("wlclImage").style.display = "block";
        document.getElementById("gender").style.display = "none";
    } else if (document.getElementById('gender').value == "King") {
        document.getElementById("wlclImage").src = "girl.gif";
        document.getElementById("wlclImage").style.display = "block";
        document.getElementById("gender").style.display = "none";
    }
}
```

```
</script>
```

```
<div style="margin-left: auto;margin-right: auto;width: 50%;display: block;">
```

```
<u>
```

```
<h1 id="start" style="display: block;">Valentine Greetings</h1>
```

```
</u>
```

```

```

```
<select id="gender" onchange="messageGif()" style="display: none;margin-left: auto;margin-right: auto;width: 50%;margin-top: 50%">
```

```
<option value="Youare">You are</option>
```

```
<option value="King">King</option>
```

```
<option value="Queen">Queen</option>
```

```
</select>
```

```
<h1 id="end" style="display: none">Takecare, Good Bye.</h1>
```

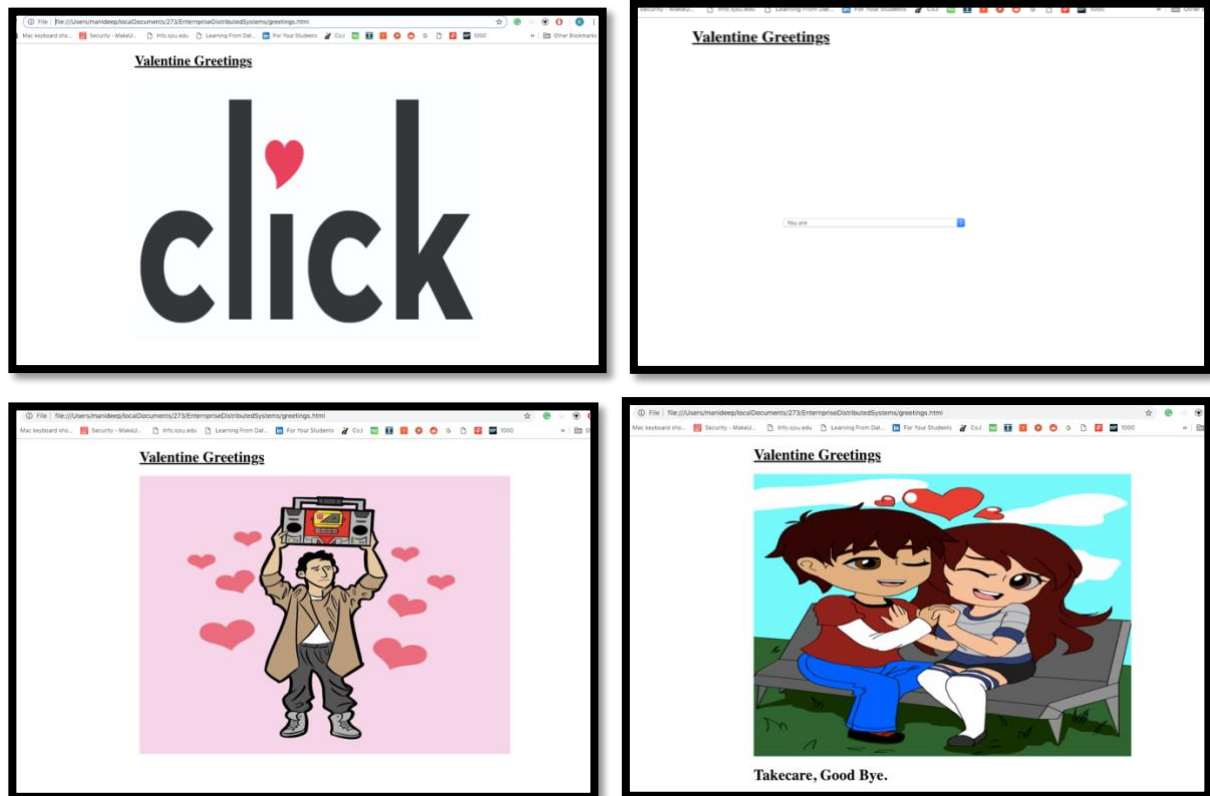
```

</div>
</body>

</html>

```

Output:



Arrays

Introduction: Arrays in JavaScript, is a special object that allows you to store more than one value at a time.

Programming Question: The four most common cars in San Jose, CA are Toyoto Camry, Nissan Altima, Ford Explorer and Subra Legacy. Their cost (USD)-MPG-Horsepower are 23845-29-301, 23750-28-236, 32365-19-365 and 22345-25-256 respectively. Write the code to represent this code in a table using Arrays.

Code:

```

<!DOCTYPE html>
<html>

<body>

  <script>

    function displayArray() {

```

```

let row1 = ["Toyoto Camry", 23845, 29, 301];
let row2 = ["Nissan Altima", 23750, 28, 236];
let row3 = ["Ford Explorer", 32365, 19, 365];
let row4 = ["Subra Legacy", 22345, 25, 256];
let finalArray = [row1,row2,row3,row4];
document.write("<table bsale=1>");
document.write("<th>Car-Model</th><th>Cost</th><th>MPG</th><th>Horsepower</th>");
for(var i=0; i<=3; i++) {
    document.write("<tr>");
    for(var j=0; j<=3; j++){
        document.write("<td>",finalArray[i][j],"</td>");
    }
    document.write("</tr>");
}
document.write("</table>");
}
</script>
<p>The four most common cars in San Jose, CA are Toyoto Camry, Nissan Altima, Ford Explorer and Subra
Legacy.

    Their cost(USD)-MPG-Horsepower are 23845-29-301, 23750-28-236, 32365-19-365 and 22345-25-256
    respectively.

    Write the code to represent
    this code in a table using Arrays.</p>
<input type="button" onclick="displayArray()" value="Output">
<p id="arrayTable"></p>

</body>

</html>

```

Output:

← → ↻ ⓘ File | file:///Users/manideep/localDocuments/273/EnterpriseDistributedSystems/Arrays.html

Apps Mac keyboard sho... Security - MakeU... Info.sjsu.edu Learning From Dat... For Your Stud

Car-Model	Cost	MPG	Horsepower
Toyoto Camry	23845	29	301
Nissan Altima	23750	28	236
Ford Explorer	32365	19	365
Subra Legacy	22345	25	256

Regular Expressions:

Introduction: Regular Expressions are the patterns that can be defined generic using predefined modifiers like say /i for case-insensitive pattern verification. Regular Expressions helps in find the match, replace the text in a file or text.

Programming Question: Write a program that finds the occurrence of a given word in the text. And also replace the text with new words when required.

Code:

```
<!DOCTYPE html>
<html>
<body>
  <script>
    function countOcc() {
      inputText = document.getElementById("actualText").value;
      findText = document.getElementById("keyword").value;
      var patt = new RegExp(findText, "g");
      var n = (inputText.match(patt) || []).length;
      document.getElementById('noc').innerHTML = n+" of Occurences.";}
    function replaceText(){
```

```

        inputText = document.getElementById("actualText").value;
        findText = document.getElementById("replaceKeyword").value;
        replaceText = document.getElementById("replaceWith").value;
        var patt = new RegExp(findText, "g");
        var str = inputText.replace(patt,replaceText);
        document.getElementById("actualText").value = str;
        document.getElementById("result").value = str;
        document.getElementById("result").style.display = "block";
        document.getElementById("actualText").style.display = "none";
        document.getElementById("heading").style.display = "none";}
    </script>
<div style="margin-left: auto;margin-right: auto;width: 50%;display: block;">
    <u>
        <h1>Occurence Counter and Replace tool</h1>
    </u>
    <h3 id="heading">Please paste you text here:</h3>
    <textarea id="actualText" rows="15" cols="80" style="width:500px;height:200px;display: block"></textarea><br
/><br />
    Keyword to Find:&nbsp;&nbsp;&nbsp;<input type="text" id="keyword" /><br /><br>
    <input type="button" id="noOfOcc" value="Find No. of occurence" onclick="countOcc()" /><br><br>
    <p id="noc"></p>
    <hr><br>
    Replace:&nbsp;&nbsp;&nbsp;<input type="text" id="replaceKeyword"
/>&nbsp;&nbsp;&nbsp;With:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    <input type="text" id="replaceWith" /><br /><br />
    <input type="button" id="replaceButton" value="replace" onclick="replaceText()" /><br><br>
    <textarea id="result" rows="15" cols="80" style="width:500px;height:200px;display: none"></textarea>
</div>
</body>
</html>

```

Output:

file:///C:/Users/Manideep/Local Documents/273/Enterprise Distributed Systems/Regex.html

MakeU... Info.sjsu.edu Learning From Dat... For Your Students CoJ TC T Y P

Occurence Counter and Replace tool

Please paste you text here:

Shawn Matthews

Damageplan was an American heavy metal supergroup from Dallas, Texas, formed in 2003. Following the demise of their previous group Pantera, brothers Dimebag Darrell and Vinnie Paul Abbott wanted to start a new band. The pair recruited former Diesel Machine and Halford guitarist Pat Lachman on vocals, and later Bob Zilla on bass. Damageplan released their only studio album New Found Power in the United States on February 10, 2004, which debuted at number 38 on the Billboard 200, selling 44,676 copies in its first week.

While Damageplan was promoting the album at a concert on December 8, 2004, at the Alrosa Villa in Columbus, Ohio, a man named Nathan Gale climbed on stage, killed guitarist Dimebag Darrell and three others and wounded another seven before being fatally shot by a police officer. Although no motive was found, some witnesses claimed Gale blamed the brothers for Pantera's breakup and believed that they had stolen his lyrics. The band's manager confirmed there are unreleased Damageplan recordings, although they have not surfaced, and the band has been inactive since the incident. Paul and Zilla had joined the band Halvao (until the former's death in 2018), and Lachman joined the Mercy Clinic.

Keyword to Find: in

Find No. of occurrence

154 of Occurences.

Replace: With:

replace

file:///Users/manideep/localDocuments/273/EnternpriseDistributedSystems/Regex.html

board sho... Security - MakeU... Info.sjsu.edu Learning From Dat... For Your Students CoJ TC T Y P

Occurence Counter and Replace tool

Keyword to Find: in

Find No. of occurrence

0 of Occurences.

Replace: in With: KKKK

replace

Damageplan
Damageplan.jpg
From left to right: Bob Zilla, KKKKnie Paul, Pat Lachman, Dimebag Darrell.
Background KKKKformation
Also known as New Found Power
OrigKKKK Dallas, Texas, U.S.
Genres
Nu metaltrash metalgroove metal
Years active 2003-2004
Labels Elektra
Associated acts
Pantera Halvao
Website damageplan.com
Past members
Nathan Gale, Paul

Strict Mode:
Introduction:

There are numerous issues which slow down the performance and use extra memory like duplicate variable declarations, vague declarations, not to access read only memory. Such kind of situations are handled by using strict mode which has paved its way in ECMAScript version 5.

Programming Question: Demonstrate the difference between strict use and normal execution.
Code:

```
"use strict";  
var eval = 12;  
var arguments = 89;
```

Output:

```
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/abc.js"  
/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/abc.js:2  
var eval = 12; //In non-strict mode eval is overwritten. Exception thrown in strict mode.  
    ^^^^^  
  
SyntaxError: Unexpected eval or arguments in strict mode  
    at new Script (vm.js:79:7)  
    at createScript (vm.js:251:10)  
    at Object.runInThisContext (vm.js:303:10)  
    at Module._compile (internal/modules/cjs/loader.js:657:28)  
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:700:10)  
    at Module.load (internal/modules/cjs/loader.js:599:32)  
    at tryModuleLoad (internal/modules/cjs/loader.js:538:12)  
    at Function.Module._load (internal/modules/cjs/loader.js:530:3)  
    at Function.Module.runMain (internal/modules/cjs/loader.js:742:12)  
    at startup (internal/bootstrap/node.js:283:19)  
  
[Done] exited with code=1 in 0.335 seconds
```

Errors:

Introduction:

The events which result in termination of program execution in run-time are called errors.

Programming Question: Focus on Error Handling in javascript using an example.

Code:

```
var array1 = ["Android", "IOS", "Symbian", "Blackberry", "Windows"];  
try {  
    for (var i = 0; i <= 7; i++) {  
        if (i >= array1.length) throw new Error("Array out of bounds");  
        console.log(array1[i]);  
    }  
} catch (err) {  
    console.log(err);  
}
```

Output:


```
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/errors.js"
Android
IOS
Symbian
Blackberry
Windows
Error: Array out of bounds
    at Object.<anonymous> (/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/errors.js:6:39)
    at Module._compile (internal/modules/cjs/loader.js:689:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:700:10)
    at Module.load (internal/modules/cjs/loader.js:599:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:538:12)
    at Function.Module._load (internal/modules/cjs/loader.js:530:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:742:12)
    at startup (internal/bootstrap/node.js:283:19)
    at bootstrapNodeJSCore (internal/bootstrap/node.js:743:3)

[Done] exited with code=0 in 0.412 seconds
```

Default Params:

Introduction:

Default params are the arguments that are given to the functions in javascript through which a default value is assigned to the function which can be used inside it. If we pass the values to the functions, the values of the default params are over-rided.

Programming Question:

Code:

```
var lease = (cost = 4500, months = 12 ) => cost*months;

console.log("Lease for 10months with 5000/month: "+lease(5000,10));
console.log("default Lease for 87months: "+lease(undefined,87));
console.log("Lease with 10000/month: "+lease(10000));
```

Output:

```
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/defaultParams.js"
Lease for 10months with 5000/month: 50000
default Lease for 87months: 391500
Lease with 10000/month: 120000

[Done] exited with code=0 in 0.454 seconds
```

Includes and typeof:

Introduction:

Type of is used to fetch the data type of value inside it. Includes is used to check the values if element is inside it or not.

Programming Question:

Code:

```
var FreshMan={Course : ["273","240","220","180C","180A","180D", "257"],
    Class:["ENG189", "ENG337", "CLARK335", "CLARK225", "BOCARD0223", "BOCARD0230",
"BOCARD0125"]}

var Output1=[];
var Output2=[];
var Output3=[];
for(var i=0;i<FreshMan.Course.length;i++){
    if((typeof FreshMan.Class[i]!=null) && FreshMan.Class[i].includes("ENG")){
        Output1.push(FreshMan.Course[i]);
    }
}
(Output1.length>0)? console.log("Classes in Engineering Building are: "+Output1):console.log("No classes")
for(var i=0;i<FreshMan.Course.length;i++){
    if((typeof FreshMan.Class[i]!=null) && FreshMan.Class[i].includes("CLARK")){
        Output2.push(FreshMan.Course[i]);
    }
}
(Output2.length>0)? console.log("Classes in Clark Hall are: "+Output2):console.log("No classes")
for(var i=0;i<FreshMan.Course.length;i++){
    if((typeof FreshMan.Class[i]!=null) && FreshMan.Class[i].includes("BOCARD0")){
        Output3.push(FreshMan.Course[i]);
    }
}
(Output3.length>0)? console.log("Classes in Bocardo Building are: "+Output3):console.log("No classes")
```

Output:

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/jscode.js"
Classes in Engineering Building are: 273,240
Classes in Clark Hall are: 220,180C
Classes in Bocardo Building are: 180A,180D,257

[Done] exited with code=0 in 0.137 seconds
```

Use of import and export

Introduction:

Export is used to throw functions from one file to another and import will use require built in function to catch the function.

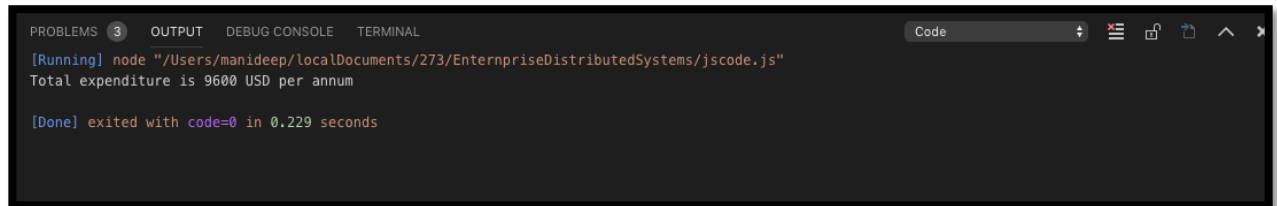
Programming Question:

Code:

```
var leaseAmount = require('./defaultParams');
rent = (leaseAmount.lease(5100,11))/77;
var expenditure = (utilities = 150, rent=500) => (utilities+rent)*12;

console.log("Total expenditure is "+expenditure(200,600)+" USD per annum");
```

Output:



```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL Code
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/jscode.js"
Total expenditure is 9600 USD per annum

[Done] exited with code=0 in 0.229 seconds
```

Type Conversions

Introduction:

Type conversion is used to convert one data type to another such as string to int, int to string and so on

Programming Question:

Code:

```
var dates=["Eigth","15",31,"null"]
var a=dates[0];
var b=dates[1];
var c=dates[2];
var d=dates[3];

console.log("First-value:"+a+"\t\tSecond-value: "+(b)+"\t\t\tThird-value: "+(c)+"\t\t\tFourth-value: "+(d));
console.log("Type of First:"+typeof(a)+"\tType of Second:"+typeof(b)+"\tType of Third:"+typeof(c)+"\tType of Fourth:"+typeof(d));

console.log("Type of b before conversion: "+b);
console.log("Type of d before conversion: "+d);
b = typeof(Number(b));
```

```
d = typeof(Number(d));
console.log("Type of b after conversion: "+b);
console.log("Type of d after conversion: "+d);
```

Output:

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/jscode.js"
First-value:Eighth      Second-value:15      Third-value:31      Fourth-value:null
Type of First:string    Type of Second:string  Type of Third:number  Type of Fourth:string
Type of b before conversion: 15
Type of d before conversion: null
Type of b after conversion: number
Type of d after conversion: number

[Done] exited with code=0 in 0.11 seconds
```

JSON

Introduction:

JSON is javascript object notation that is used to share information between the servers and clients.

Programming Question:

Code:

```
var kous={
  "peopleName":["Ranji","Harshi"],
  "peopleSchool":["MIT","Stanford"],
  "studentCourses":["Software Engineering","Information Engineering"]
}
for(var i=0;i<kous.peopleName.length;i++){
  console.log(kous.peopleName[i]+" is studying in "+kous.peopleSchool[i]+" studying course "+kous.studentCourses[i])
}
```

Output:

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/json.js"
Ranji is studying in MIT studying course Software Engineering
Harshi is studying in Stanford studying course Information Engineering

[Done] exited with code=0 in 0.154 seconds
```

Object and Class

Introduction:

Class is a user defined type which is a collection of variables and functions. It can define objects which are used for manipulation. Variables and functions can be added dynamically using Prototype.

Programming Question:

Code:

```
class Stylus {
  constructor(name, manufacturer, cost, type, color, width) {
    this.name = name;
    this.manufacturer = manufacturer;
    this.cost = cost;
    this.type = type;
    this.color = color;
    this.width = width;
  }
  changeColor(inkColor) {
    this.color = inkColor;
    return this.color;
  }
  paintNow(brushWidth) {
    this.width = brushWidth;
    return this.width;
  }
}

let surfaceStylus = new Stylus("SurfaceBookStylus", "Microsoft", "130$", "feather-touch", "white", "0.5mm");
console.log(surfaceStylus.changeColor("Blue"));
console.log(surfaceStylus.paintNow("4mm"));
```

Output:

```
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/jscode.js"
Blue
4mm

[Done] exited with code=0 in 0.407 seconds
```

Object.assign

Introduction:

It is used to copy information from one object to other. Also, there can be many sources but one target.

Programming Question:

Code:

```
class Stylus {
  constructor(name, manufacturer, cost, type, color, width) {
    this.name = name;
```

```

        this.manufacturer = manufacturer;

        this.cost = cost;

        this.type = type

        this.color = color;

        this.width = width;
    }

    changeColor(inkColor) {
        this.color = inkColor;

        return this.color;
    }

    paintNow(brushWidth) {
        this.width = brushWidth;

        return this.width;
    }
}

class mobileStylus {
    constructor(name, manufacturer, ) {
        this.name = name;

        this.manufacturer = manufacturer;
    }
}

let surfaceStylus = new Stylus("SurfaceBookStylus", "Microsoft", "130$", "feather-touch", "white", "0.5mm");
let noteNineStylus = new mobileStylus("Note9", "Samsung");
let bothPurposeStylus = Object.assign({}, surfaceStylus, noteNineStylus);
console.log(bothPurposeStylus);

```

Output:

```
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/jscode.js"
{ name: 'Note9',
  manufacturer: 'Samsung',
  cost: '130$',
  type: 'feather-touch',
  color: 'white',
  width: '0.5mm' }

[Done] exited with code=0 in 0.447 seconds
```

Static Method

Introduction:

Static variables or methods are the ones which are used to create them globally. They can be accessed by using class name.

Programming Question:

Code:

```
class Vehicle {
  constructor(VehicleName, VehiclemodelNo, PreferredColour, speedofTheCar=0){
    this.VehicleName=VehicleName;
    this.VehiclemodelNo=VehiclemodelNo;
    this.PreferredColour=PreferredColour;
    this.speedofTheCar=speedofTheCar;
  }
  velocamoro(speedMentioned){
    this.speedofTheCar+=speedMentioned;
    return this.speedofTheCar;
  }
  braking(brakingPower){
    this.speedofTheCar-=brakingPower;
    if(this.speedofTheCar==0){
      Vehicle.lightsoffplease();
    }
    return this.speedofTheCar;
  }
  static lightsoffplease(){
    console.log("Lights were offed!");
  }
}
```

```
let royMustang = new Vehicle("camoro","Chevrolet","red",120);
console.log(royMustang.velocamoro(40));
royMustang.braking(160);
console.log(royMustang.speedofTheCar);
```

Output:

```
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/static.js"
160
Lights were offed!
0
[Done] exited with code=0 in 0.492 seconds
```

Inheritance using sub-class in Javascript

Introduction:

The process of sending the functionality from parent to child is known as inheritance. Child classes can override the methods of parent class.

Programming Question:

Code:

```
class Computer {
  constructor(brand, memory, display, player, cost) {
    this.brand = brand;
    this._memory = memory;
    this.display = display;
    this.player = player;
    this.cost = cost;
  }
  get memory() {
    return this._memory;
  }
  basicSimple() {
    console.log("Purchase a basic computer");
  }
}
class upgradedComputer extends Computer {
  constructor(brand, memory, display, player, cost, bluetooth, gps, hdmi) {
    super(brand, memory, display, player, cost);
    this.bluetooth = bluetooth;
    this.gps = gps;
    this.hdmi = hdmi;
  }
}
```



```

    }
    connectOther() {
        console.log("Connecting bluetooth devices");
    }
}
class mobileComputer extends upgradedComputer {
    constructor(brand, memory, display, player, cost, bluetooth, gps, hdmi, serviceNetwork, mobileInternetModule,
dualcamera) {
        super(brand, memory, display, player, cost, bluetooth, gps, hdmi);
        this.serviceNetwork = serviceNetwork;
        this.mobileInternetModule = mobileInternetModule;
        this.dualcamera = dualcamera;
    }
    videoConference() {
        console.log("Conferencing over videos");
    }
}
var alienware = new upgradedComputer("Dell", "1024", "1376", "yes", "900", "yes", "yes", "yes")
alienware.connectOther();
console.log(alienware.memory);
var ios = new mobileComputer("Apple", "128", "Retina", "yes", "1500", "yes", "yes", "yes", "yes");
ios.videoConference();

```

Output:

```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/jscode.js"
Connecting bluetooth devices
1024
Conferencing over videos

[Done] exited with code=0 in 0.108 seconds

```

Method overriding

Introduction:

When there is an inheritance from parent class to child class, the method having same name in child class will override the method in parent class.

Programming Question:

Code:

```

class Computer {
    constructor(brand, memory, display, player, cost) {
        this.brand = brand;
        this._memory = memory;
    }
}

```

```

        this.display = display;
        this.player = player;
        this.cost = cost;
    }
    get memory() {
        return this._memory;
    }
    basicSimple() {
        console.log("Purchase a basic computer");
    }
}

class upgradedComputer extends Computer {
    constructor(brand, memory, display, player, cost, bluetooth, gps, hdmi) {
        super(brand, memory, display, player, cost);
        this.bluetooth = bluetooth;
        this.gps = gps;
        this.hdmi = hdmi;
    }
    connectOther() {
        console.log("Connecting bluetooth devices");
    }
}

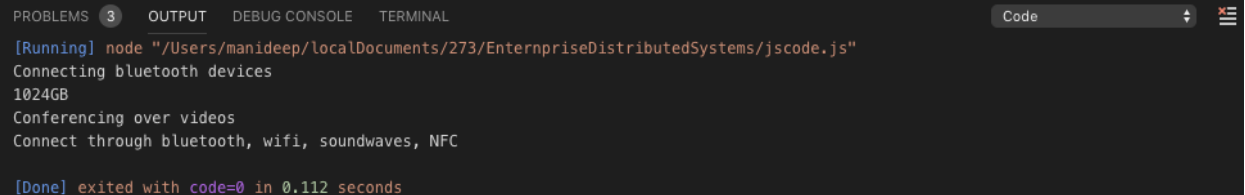
class mobileComputer extends upgradedComputer {
    constructor(brand, memory, display, player, cost, bluetooth, gps, hdmi, serviceNetwork, mobileInternetModule,
    dualcamera) {
        super(brand, memory, display, player, cost, bluetooth, gps, hdmi);
        this.serviceNetwork = serviceNetwork;
        this.mobileInternetModule = mobileInternetModule;
        this.dualcamera = dualcamera;
    }
    connectOther() {
        console.log("Connect through bluetooth, wifi, soundwaves, NFC");
    }
    videoConference() {
        console.log("Conferencing over videos");
    }
}

var alienware = new upgradedComputer("Dell", "1024GB", "1376", "yes", "900", "yes", "yes", "yes")
alienware.connectOther();
console.log(alienware.memory);

```

```
var ios = new mobileComputer("Apple", "128GB", "Retina", "yes", "1500", "yes", "yes", "yes", "yes", "yes");
ios.videoConference();
ios.connectOther();
```

Output:



```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL Code
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/jscode.js"
Connecting bluetooth devices
1024GB
Conferencing over videos
Connect through bluetooth, wifi, soundwaves, NFC

[Done] exited with code=0 in 0.112 seconds
```

Use of get

Introduction:

Get is used as a method to retrieve value from the object of class that is private.

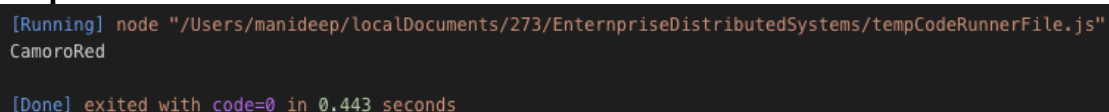
Programming Question:

Code:

```
class transportbyCar {
    constructor(carname, vehiclemodelNo, vehiclepaintcolor, travellingbySpeed=0){
        this.carname=carname;
        this.vehiclemodelNo=vehiclemodelNo;
        this._vehiclepaintcolor=vehiclepaintcolor;
        this.nowspeed=travellingbySpeed;
    }
    get vehiclepaintcolor(){
        return this._vehiclepaintcolor;
    }
    velocity(userspeed){
        this.nowspeed+=userspeed;
        return this.nowspeed;
    }
}

let kous = new transportbyCar("Camaro","Petrol","CamoroRed",120);
console.log(kous.vehiclepaintcolor);
```

Output:



```
[Running] node "/Users/manideep/localDocuments/273/EnterpriseDistributedSystems/tempCodeRunnerFile.js"
CamoroRed

[Done] exited with code=0 in 0.443 seconds
```

fetch

Introduction:

Fetch is used to retrieve the information from another application on a server using an external API.

Programming Question:

Code:

```

<!DOCTYPE html>
<html>
<head>
  <script>
    const url = 'http://www.recipepuppy.com/api/?i=onions,garlic&q=omelet&p=3';
    function getdata(){
      fetch('http://www.clashapi.xyz/api/arenas')
      .then(function(res){
        var a=res.json();
        console.log(a)
        return a;})
      .then(function(data){
        let output=`<table style="width:100%">
          <tr>
            <th>Victory Gold</th>
            <th>Name</th>
            <th>MinTrophies</th>
          </tr>`;
        data.forEach(function(arena){
          output += `
          <tr>
            <td>${arena.victoryGold}</td>
            <td>${arena.name}</td>
            <td>${arena.minTrophies}</td>
          <tr>`;
        })
        document.getElementById("output").innerHTML=output});
      }
    </script>
  </head>
  <body onload="getdata()">
    <div id="output"></div>
  </body>
</html>

```

Output:

	Victory Gold	Name	MinTrophies
0		Training Camp	0
5		Goblin Stadium	0
7		Bone Pit	400
9		Barbarian Bowl	800
11		P.E.K.K.A.'s Playhouse	1100
12		Spell Valley	1400
14		Builder's Workshop	1700
15		Royal Arena	2000
16		Frozen Peak	2300
18		Jungle Arena	2600
20		Hog Mountain	3000
22		Electro Valley	3400
24		Legendary Arena	3800

HTML5

Local Storage

Introduction: Since the HTTP transport layer is stateless, the data gets reset everytime we close and open. And for this, data used to be stored previously using cookies, where it stores to a maximum of 4KB, sent with every request leading to the load on server. Hence, for this overhead, HTML5 came up with HTML Local storage. These store data in the form of key-value pairs and is available through two versions, Session Storage and Local Storage.

-Session Storage data is available for that complete session (till the window is closed and then get deleted). The session storage is not available to multiple windows even from same domain unlike Local Storage.

-Local Storage is available to all the windows of same origin (Domain). The Local Storage data is available even after the browser is closed.

Programming Question: Store the details of Books using HTML Local Storage

Code:

```
<!DOCTYPE html>
<html>
<title>HTML Local Storage </title>

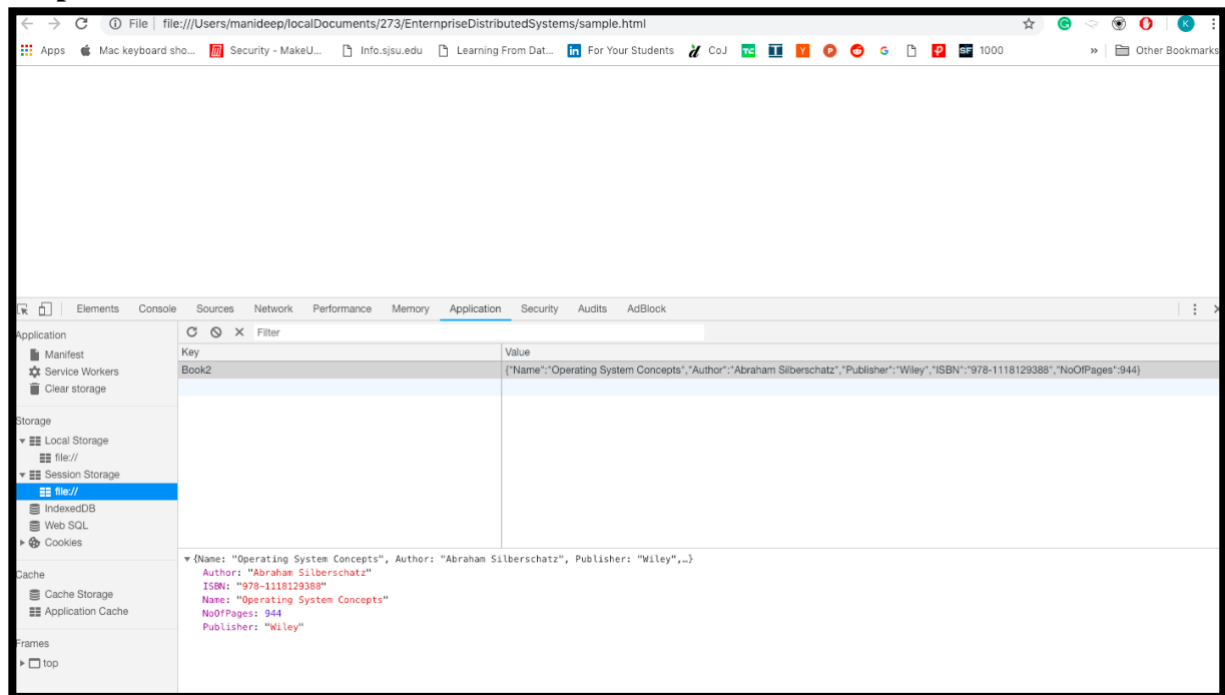
<body>
  <h1>HTML Local Storage</h1>
  <script>
    var book1 = {};
    book1.Name = 'Fundamentals of Computer Architecture and Design';
    book1.Author = 'Ahmet Bindal';
    book1.Publisher = 'Springer International Publishing';
    book1.ISBN = '978-3-319-25811-9';
```

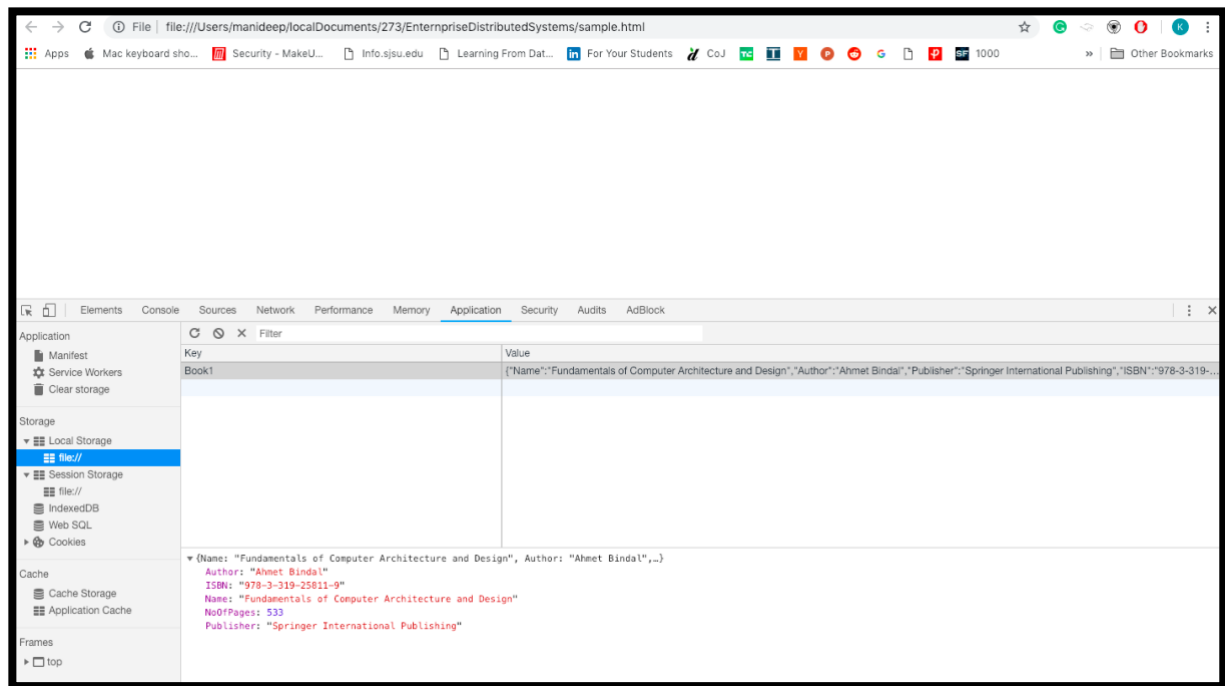
```

book1.NoOfPages = 533;
var book2 = {};
book2.Name = 'Operating System Concepts';
book2.Author = 'Abraham Silberschatz';
book2.Publisher = 'Wiley';
book2.ISBN = '978-1118129388';
book2.NoOfPages = 944;
localStorage.setItem('Book1', JSON.stringify(book1));
console.log(JSON.parse(localStorage.getItem('Book1')));
sessionStorage.setItem('Book2', JSON.stringify(book2));
console.log(JSON.parse(sessionStorage.getItem('Book2')));
</script>
</body>
</html>

```

Output:





Media (Video and Audio)

Introduction: The support for media i.e., video and audio has been added from HTML5 providing different features like player controls for video and audio, loading the content, muting the video, add subtitles, auto play and several different features.

Programming Question: Display a video with multiple dimensions on webpage that starts playing automatically with voice muted. Make the audio and video replay continuously. Enable support for audio with .mp3 and .ogg formats.

Code:

```
<!DOCTYPE html>
<html>

<body style="background-color:rgb(154, 215, 240)">
    <h2>Video in different dimensions</h2>
    <video width="600" height="500" autoplay loop muted controls>
        <source src="fishes.mp4" type="video/mp4">
        <track src="fishes.srt" kind="subtitles" srclang="en" label="English">
        Browser not supporting the video.
    </video>
    <video width="320" height="240" autoplay loop muted controls>
        <source src="fishes.mp4" type="video/mp4">
        <track src="fishes.srt" kind="subtitles" srclang="en-US" label="English" />
        Browser not supporting the video.
```

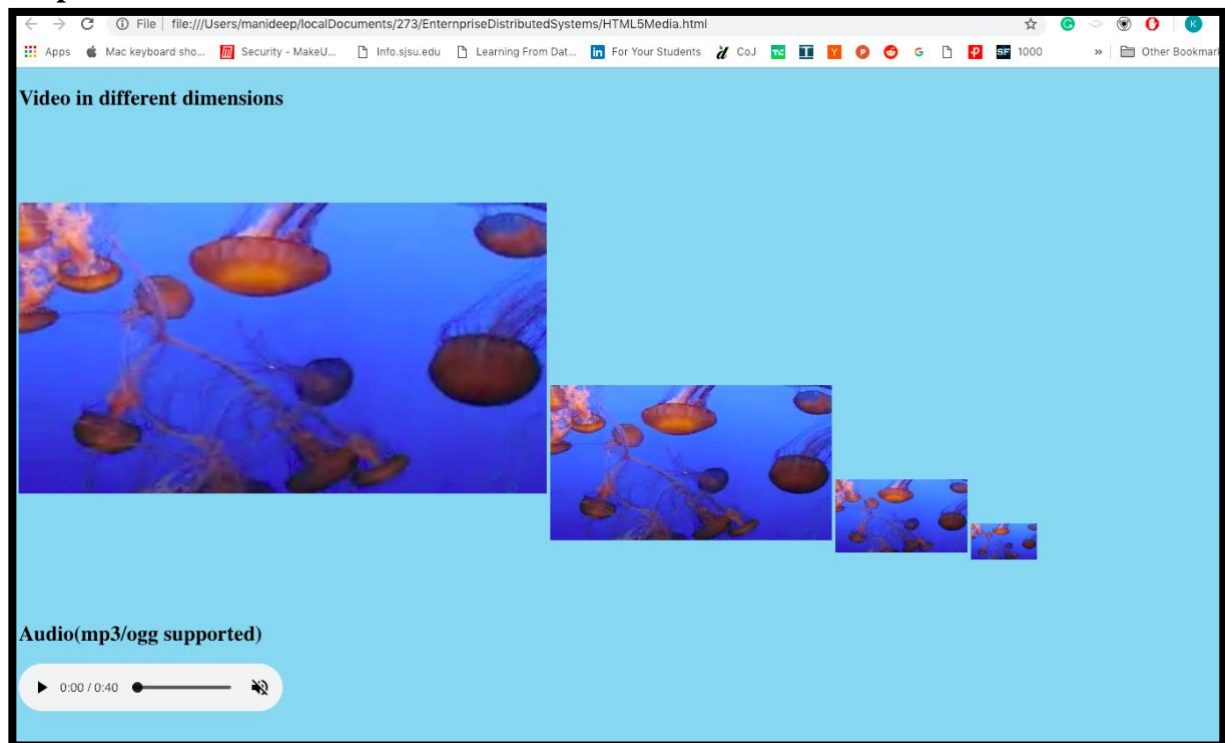


```

</video>
<video width="150" height="120" autoplay loop muted controls>
  <source src="fishes.mp4" type="video/mp4">
  <track src="fishes.srt" kind="subtitles" srclang="en-US" label="English" />
  Browser not supporting the video.
</video>
<video width="75" height="60" autoplay loop muted controls>
  <source src="fishes.mp4" type="video/mp4">
  <track src="fishes.srt" kind="subtitles" srclang="en-US" label="English" />
  Browser not supporting the video.
</video>
<br><br><br>
<h2>Audio(mp3/ogg supported)</h2>
<audio loop muted controls>
  <source src="viper.ogg" type="audio/ogg">
  <source src="viper.mp3" type="audio/mpeg">
  Browser not supporting the audio.
</audio>
</body>
</html>

```

Output:



Introduction

most common types of input are text, submit, reset, password etc. HTML5 has added some more input types like email, requires, search, url, number, tel etc.

Programming Question: Create a html page for saving cooking recipes and details.

Code:

[illegible]

```

Rating:<input type="number" name="Rating" min="1" max="5"><br><br>
Email contact:&nbsp;<input type="email" name="email"><br><br>
<input type="submit" value="Save">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="reset"><br><br>
</div>

</div>
</body>

</html>

```

Output:

The screenshot shows a web browser window with the address bar displaying a local file path. The browser's taskbar at the top shows various open applications. The main content area displays a form titled "Recipe Entry" with the following fields and controls:

- Recipe:** A text input field containing "Mutton Biryani".
- Cooked Date:** A date input field showing "02/06/2019".
- Photo Upload:** A button labeled "Choose file" followed by the filename "20180815_160612.jpg".
- Type of Cuisine :** A dropdown menu with "Thai" selected.
- Spices:** Three checkboxes labeled "spice1" (checked), "spice2", and "spice3".
- Process:** A text area containing the following text:

Drain and wash the mutton under cold running water. Pat dry and add garam masala, salt, ginger-garlic paste, red chilli powder and two cups of curd.

Put it in a clean film plastic bag and keep it in the refrigerator to marinate overnight. Cook rice with salt and oil till it is almost done. Keep it aside. Fry thinly sliced onions until golden brown. Add 1/3 of the golden-brown onion to the mutton marination and keep the rest aside.

Now in a bandj (deep pan), add the marinated mutton at
- Rating:** A number input field with the value "5".
- Email contact:** A text input field containing "abc@abc.com".
- Buttons:** "Save" and "Reset" buttons at the bottom of the form.

Geolocation

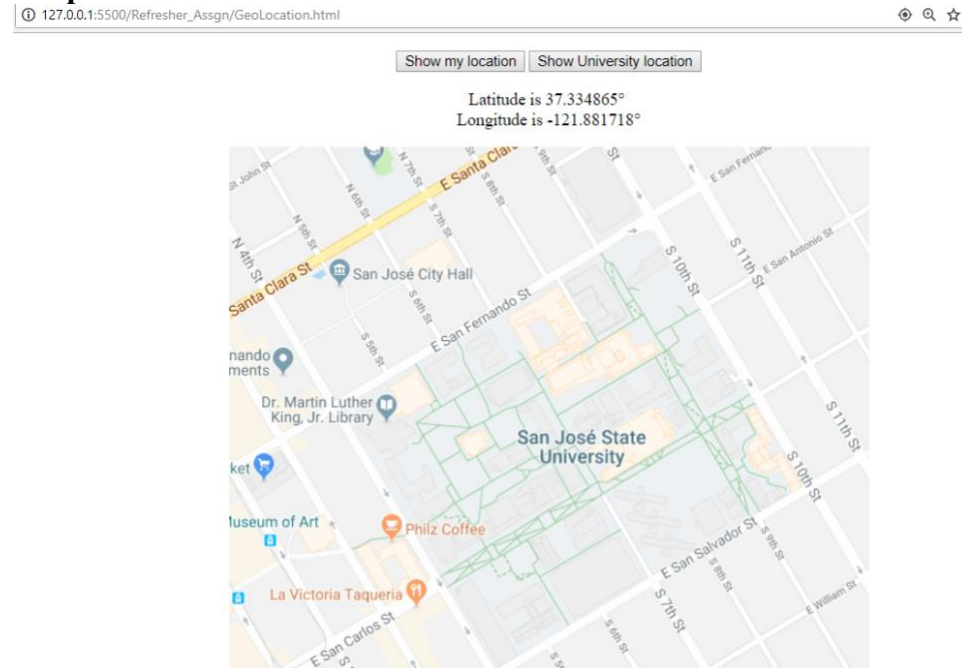
Introduction: This refers to the feature of getting the geographic position using the API navigator.geolocation object. This object retrieves the position using two main functions, getCurrentPosition() and watchPosition(). Along with location data, this object also provides other details like altitude, accuracy, heading, speed from metadata.

Programming Question: Write a html page that displays your location and points the position on a map.

Code:

```
<html lang="en">
<body style="text-align:center;">
  <p><button onclick="MyLoc()">Show my location</button>
    <button onclick="ClgLoc()">Show University location</button></p>
  <div id="out"></div>
</body>
<script>
  function MyLoc() {
    var output = document.getElementById("out");
    function success(position) {
      var lat = position.coords.latitude;
      var long = position.coords.longitude;
      output.innerHTML = '<p>Latitude is ' + lat + '° <br>Longitude is ' + long + '°</p>';
      var img = new Image();
      img.src = "https://maps.googleapis.com/maps/api/staticmap?center=" + lat + "," + long +
"&zoom=16&size=600x600&sensor=true";
      output.appendChild(img);
    }
    function error() {
      output.innerHTML = "Unable to retrieve your location";
    }
    navigator.geolocation.getCurrentPosition(success, error);
  }
  function ClgLoc() {
    var output = document.getElementById("out");
    function success(position) {
      var lat = 37.334865;
      var long = -121.881718;
      output.innerHTML = '<p>Latitude is ' + lat + '° <br>Longitude is ' + long + '°</p>';
      var img = new Image();
      img.src = "https://maps.googleapis.com/maps/api/staticmap?center=" + lat + "," + long +
"&zoom=16&size=600x600&sensor=true";
      output.appendChild(img);
    }
    function error() {
      output.innerHTML = "Unable to retrieve your location";
    }
    navigator.geolocation.getCurrentPosition(success, error);
  }
</script>
</html>
```

Output:



Java:

Arrays:

Introduction to the topic:

Arrays is a collection of data which is of similar data type.

Programming Question:

Calculate a company's average transactions.

Code:

Amazon.java → Class definition

package learn;

```
public class Amazon {
    private Integer transactions;
    private String sector;

    public Amazon(Integer transactions, String sector) {
        this.transactions=transactions;
        this.sector=sector;
    }

    public Integer getTransactions() {
        return transactions;
    }

    public String getSector() {
        return sector;
    }

    public void setTransactions(Integer transactions) {
        this.transactions = transactions;
    }
}
```

```

    }

    public void setSector(String sector) {
        this.sector = sector;
    }
}

```

AverageTransactions.java → Function definition

```
package learn;
```

```

class AverageTransactions {
    public float calavgTransactions(Amazon[] sectores) {
        float avgTransactions=(float) 0;
        Integer Transactions=0;
        for(Amazon sector : sectores) {
            Transactions+=sector.getTransactions();
        }
        avgTransactions=(float) (Transactions/sectores.length);
        return avgTransactions;
    }
}

```

AverageSalesTest.java→ Function call and testing

```
package learn;
```

```

import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Test;

```

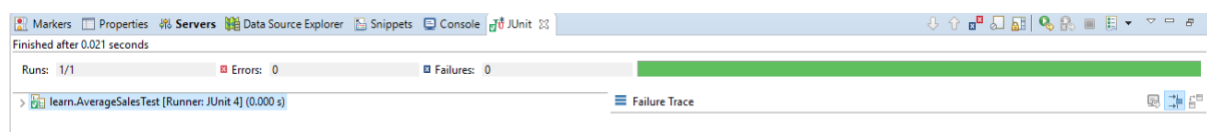
```

public class AverageSalesTest {
    private AverageTransactions classUnderTest;
    @Before
    public void setUp() throws Exception{
        classUnderTest = new AverageTransactions();
    }
    @Test
    public void testCal() {
        float expectedValue=(float) (109/6);
        Amazon s1=new Amazon(14, "AZ");
        Amazon s2=new Amazon(19, "CA");
        Amazon s3=new Amazon(21, "NY");
        Amazon s4=new Amazon(11, "TX");
        Amazon s5=new Amazon(18, "VA");
        Amazon s6=new Amazon(26, "IN");
        Amazon sectores[]= {s1,s2,s3,s4,s5,s6};

        assertTrue(expectedValue.equals(classUnderTest.calavgTransactions(sectores)));
    }
}

```

Output:



Interface:

Introduction to the topic:

Interface is group of methods with only method definitions and variables are public, static and final.

Programming Question:

Write a program to demonstrate the usage of credit card interface

Code:

Interfaces.java→

```
package learn;

interface CcCard{
    Float points = (float) 100;

    Float redeemRewards(Integer pointsToRedeem);
    Float addRewards(Integer points);
}
```

Interface_main.java→

```
package learn;

public class Interface_main implements CcCard{
    Float pts=points;
    @Override
    public Float redeemRewards(Integer points1) {
        if(pts>200) {
            pts-=points1;
            System.out.println("Points redeemed!");
        }

        return (float)pts;
    }
    @Override
    public Float addRewards(Integer points1) {
        pts+=points1;
        return pts;
    }
}
```

Interface_mainTest.java→

```
package learn;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

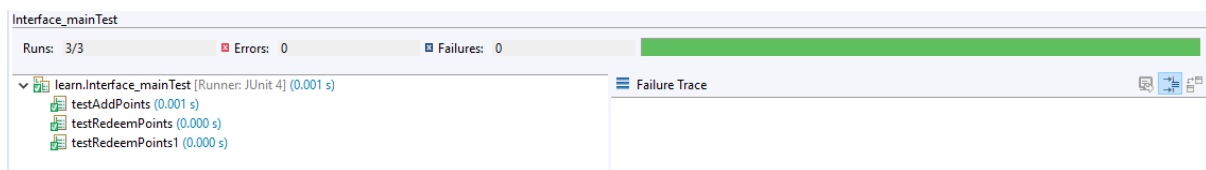
public class Interface_mainTest {
    private Interface_main classUnderTest;
```

```

@Before
public void setUp() throws Exception{
    classUnderTest = new Interface_main();
}
@Test
public void testAddRewards() {
    Float expectedValue= (float) 340.0;
    assertTrue(expectedValue.equals(classUnderTest.addRewards(240)));
}
@Test
public void testRedeemRewards() {
    Float expectedValue=(float) (100);
    assertTrue(expectedValue.equals(classUnderTest.redeemRewards(100)));
}
@Test
public void testRedeemRewards1() {
    Float expectedValue=(float) (139);
    classUnderTest.addRewards(240);
    assertTrue(expectedValue.equals(classUnderTest.redeemRewards(201)));
}
}

```

Output:



Generics:

Introduction to the topic:

Generics are declarations that are used to initialize the variables of not known types which means that generics can manage any kind of data that is given to it. Type casting is done in compile time. It is an added advantage.

Programming Question:

Code:

Sale.java→

```
package collections;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```

public class Sale<O,T> {
    private O saleId;
    private T saleName;
    public List l= new ArrayList<>();
    public O getSaleId() {
        return saleId;
    }
    public void setSaleId(O id) {
        this.saleId = id;
    }
    public T getSaleName() {
        return saleName;
    }
}

```



```

        public void setSaleName(T saleName) {
            this.saleName = saleName;
        }
        public <O,T> void addSale(O Id, T name) {
            Sale<O,T> O= new Sale<>();
            O.setSaleId(Id);
            O.setSaleName(name);
            l.add(O);
        }
        public <E> List getSales(){

            return l;
        }
    }
}

```

Sale_fun.java→

```

package collections;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Sale_fun {
    List l= new ArrayList<>();
}

```

Sale_funtest.java→

```

package collections;

import static org.junit.Assert.*;

import java.util.ArrayList;
import java.util.List;

import org.junit.Before;
import org.junit.Test;
import collections.Sale;

public class Sale_funTest {
    private Sale_fun classUnderTest;
    @Before
    public void setUp() throws Exception{
        classUnderTest = new Sale_fun();
    }
    @Test
    public <O,T> void test() {

        List l= new ArrayList<>();
        Sale<Integer, String> obj1= new Sale<>();
        obj1.setSaleId(101);
        obj1.setSaleName("Margarita");
        l.add(obj1);

        Sale<Integer, String> obj2= new Sale<>();
        obj2.setSaleId(102);
        obj2.setSaleName("Pepperoni");
        l.add(obj2);
    }
}

```

```

        Sale<Integer, String> a= new Sale<>();
        a.addSale(101, "Margarita");
        a.addSale(102, "Pepperoni");
        List l1=a.getSales();

        for(Object o:l1) {
            Sale ob1=(Sale)o;

        }

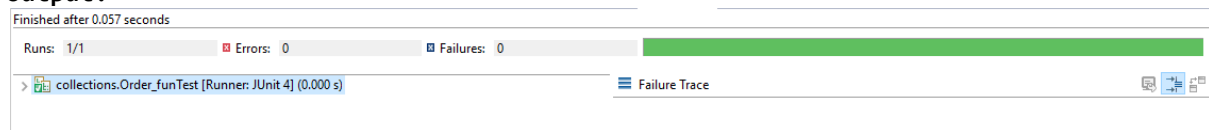
        for(Object o:l) {
            Sale ob1=(Sale)o;

        }

        assertTrue(l.equals(l1));
    }
}

```

Output:



Collections:

Introduction to the topic:

Collections is a stereotype which includes the interfaces and also its implementations. These are the data structures used to organize and play with data.

Examples: List, Set, Map

Programming Question:

Using a hashmap to handle the details of a player.

Code:

FreshMan.java → characteristics of FreshMan

```

package collections;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class FreshMan {
    private String Name;
    private Float score;
    public String getName() {
        return Name;
    }
    public void setName(String name) {
        Name = name;
    }
    public Float getScore() {
        return score;
    }
    public void setScore(Float score) {
        this.score = score;
    }
}

```

FreshMan_service.java→functions of FreshMan

```
package collections;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class FreshMan_service {
    public Map<String, Float> itemMap = new HashMap<String, Float>();
    public void addFreshMan(String name, Float score) {
        itemMap.put(name, score);
    }
    public Float getFreshManMarks(String name) {
        Float score = itemMap.get(name);
        return score;
    }
}
```

FreshMan_serviceTest→ Testing

```
package collections;

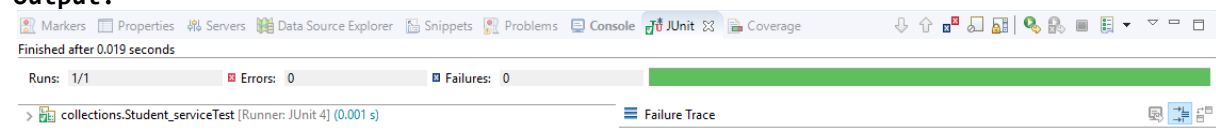
import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class FreshMan_serviceTest {
    private FreshMan classUnderTest;
    @Before
    public void setUp() throws Exception {
        classUnderTest = new FreshMan();
    }

    @Test
    public void test() {
        Float expectedValue = (float) 92.00;
        FreshMan_service obj1 = new FreshMan_service();
        obj1.addFreshMan("Koushik", (float) 92.00);
        Float currentValue = obj1.getStudenMarks("Vinay");
        assertTrue(expectedValue.equals(currentValue));
    }
}
```

Output:



Stack:

Introduction to the topic:

Stack is a data structure which is of user defined data type which follows Last in first out procedure which has insert and delete operations.

Programming Question:

A program to demonstrate a functionality of Jeans rack.

Jeanstack.java→

```
package learn;

public class Jeanstack {
    Integer size;
    String[] Jeans;
    Integer top;
    public Jeanstack(Integer Ssize) {
        top=-1;
        Jeans=new String[Ssize];
    }
    public Integer size() {
        return top+1;
    }
    public void push(String data) {
        Jeans[top+1]=data;
        top++;
    }
    public boolean is_empty() {
        return (top==-1);
    }
    public String pop() {
        if(!is_empty()) {
            return Jeans[top--];
        }
        return "";
    }
    public void display() {
        for(int i=0;i<=top;i++) {
            System.out.println(Jeans[i]);
        }
    }
}
```

JeanstackTest.java→

```
package learn;
```

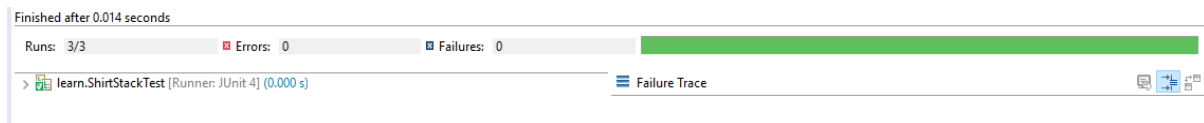
```
import static org.junit.Assert.*;
```

```
import org.junit.Test;

public class JeanstackTest {

    @Test
    public void is_empty() {
        Jeanstack Jeans=new Jeanstack(10);
        Jeans.push("Grey Jeans");
        Jeans.pop();
        assertTrue(Jeans.is_empty()==true);
    }
    @Test
    public void pop() {
        Jeanstack Jeans=new Jeanstack(10);
        assertTrue(Jeans.pop()=="");
    }
    @Test
    public void size() {
        Jeanstack Jeans=new Jeanstack(10);
        Jeans.push("Grey Jeans");
        Jeans.push("White Jeans");
        assertTrue(Jeans.size().equals(2));
    }
}
```

Output:



Queue:

Introduction to the topic:

Queue is also a data structure which is a user defined data type that implements first in first out.

Programming Question:

program to demonstrate the Queue.

Code:

Queueline.java

```
package learn;

public class Queueline {
    Integer rear=-1,front=0;
    String[] q= new String[10];
    public Integer size() {
        if(front>rear) {
            return 0;
        }
        return rear-front+1;
    }
    public void push(String data) {
        q[++rear]=data;
    }
    public boolean is_empty() {
        return (rear-front<=0);
    }
    public String pop() {
        if(!is_empty()) {
            return q[front++];
        }
        return "";
    }
    public void display() {
        for(String q:q) {
            System.out.println(q);
        }
    }
}
```

QueuelineTest.java→

```

package learn;

import static org.junit.Assert.*;

import org.junit.Test;

public class QueueLineTest {

    @Test
    public void is_empty() {
        QueueLine q=new QueueLine();
        q.push("Vinay");
        q.pop();
        assertTrue(q.is_empty()==true);
    }
    @Test
    public void pop() {
        QueueLine q=new QueueLine();
        assertTrue(q.pop()=="");
    }
    @Test
    public void size() {
        QueueLine q=new QueueLine();
        q.push("Vinay");
        q.push("Sandeep");
        assertTrue(q.size().equals(2));
    }
}

```

Output:

Finished after 0.017 seconds

Runs: 3/3	Errors: 0	Failures: 0
-----------	-----------	-------------

> learn.QueueTest [Runner: JUnit 4] (0.000 s)

Failure Trace

Multi-Threading:

Introduction to the topic:

Java supports the concept of multi-threading which executes multiple parts of a program at the same time. A series of statements in a program which are executed at the same time is called a thread. We manually override the process of multi threading by programming.

Programming Question:

Program that synchronizes the MoneyDeposits and also moneyWithdrawals of a bank.

Code:

BankAccount.java→

```
package learn;

class BankAccount {
    Float balance;
    BankAccount(Float balance) {
        this.balance = balance;
    }
    public void moneyWithdraw(Float amount) {
        this.balance -= amount;
    }
    public void MoneyDeposit(Float amount) {
        this.balance += amount;
    }
}

class MoneyWithdraw extends Thread {
    BankAccount bankAccount;
    Float amt;
    MoneyWithdraw(BankAccount bankAccount, Float amt) {
        this.bankAccount = bankAccount;
        this.amt = amt;
    }
    @Override
    public void run() {
        try {
            if(bankAccount.balance >= amt) {
                Thread.sleep(100);
                bankAccount.moneyWithdraw(this.amt);
                System.out.println("Balance after moneyWithdrawal: " +
bankAccount.balance);
            }
            else throw new Exception("Not enough balance");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

class MoneyDeposit extends Thread {
    BankAccount bankAccount;
    Float amt;

    MoneyDeposit(BankAccount bankAccount, Float amt) {
        this.bankAccount = bankAccount;
        this.amt = amt;
    }
}
```



```

@Override
public void run() {
    try {
        Thread.sleep(50);
        bankAccount.MoneyDeposit(this.amt);
        System.out.println("Balance after MoneyDeposit: " + bankAccount.balance);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
}

```

BankAccountTest.java→

package learn;

import static org.junit.Assert.*;

import org.junit.Test;

public class BankAccountTest {

@Test

public void test() throws Exception{

BankAccount myAcc = new BankAccount((float) 300);

MoneyWithdraw w= new MoneyWithdraw(myAcc, (float) 200);

MoneyDeposit d=new MoneyDeposit(myAcc, (float) 300);

w.start();

d.start();

Thread.sleep(1000);

assertTrue(400.0==myAcc.balance);

}

}

Output:

