**Java Script Complex Assignment**

**Array Function Assignments**

16. Create a function named reversePlusOne. This function should:

* + Take one argument, an array of at least two numbers.
  + This function should return:
    - the array *reversed* with a 1 added at the front

For example:

reversePlusOne([1,2]); // returns [1,2,1]

reversePlusOne([5,4,3,2]); // returns [1,2,3,4,5]

<!DOCTYPE html>

<html>

<h3>Quantity Plus One</h3>

<body>

<p id="demo"></p>

<script >

var arr=[5,4,3,2];

document.getElementById("demo").innerHTML = reverseplusone(arr);

function reverseplusone(array){

array.reverse();

array.unshift(1);

return array;

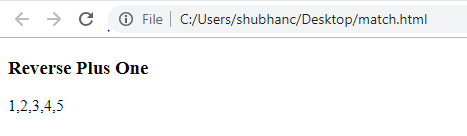
}

reverseplusone(array);

</script>

</body>

</html>



* Create a function named plusesEverywhere. This function should:
  + Take one argument, an array of at least two numbers.
  + This function should return:
    - a String made of all the values in the array separated by +

For example:

plusesEverywhere([1,2,3]); // returns "1+2+3"

plusesEverywhere([18,24]); // returns "18+24"

<!DOCTYPE html>

<html>

<h3>Pluses Everywhere</h3>

<body>

<p id="demo"></p>

<script >

var array=[53,63,45,85];

document.getElementById("demo").innerHTML = plusesEverywhere(array);

function plusesEverywhere(array){

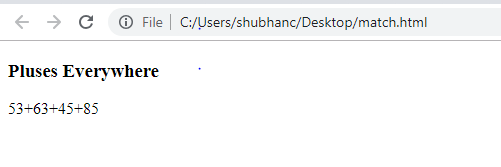
return array.join("+");

}

</script>

</body>

</html>



* Create a function named arrayQuantityPlusOne. This function should:
  + Take one argument, an array of numbers.
  + This function should return:
    - one greater than the number of items in the array

For example:

arrayQuantityPlusOne([0,0,1,0,2,1]); // returns 7

arrayQuantityPlusOne([42]); // returns 2

<!DOCTYPE html>

<html>

<h3>Quantity Plus One</h3>

<body>

<p id="demo"></p>

<script >

document.getElementById("demo").innerHTML = quantityplusone([53,63,45,85]);

function quantityplusone(array){

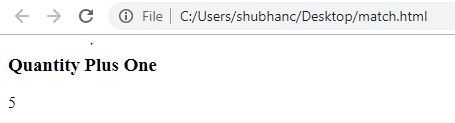
return (array.length+1);

}

</script>

</body>

</html>



# Object Oriented Basics

17. Complete the createCourse function. This function should:

* + take three arguments that will define course properties
    - courseTitle (string)
    - courseDuration (string)
    - courseStudents (array)
  + return an object that has each property assigned its proper value

For example:

createCourse('Bloc Front-End Engineering', '4 weeks', ['Joe', 'Tim', 'Rob'])

// should return {title: 'Bloc Front-End Engineering', duration: '4 weeks', students: ['Joe', 'Tim', 'Rob']}

<!DOCTYPE html>

<html>

<h3>Create Course</h3>

<body>

<p id="demo"></p>

<script >

var txt="";

var x;

function createCourse(courseTitle, courseDuration, courseStudents)

{

this.courseTitle=courseTitle;

this.courseDuration=courseDuration;

this.courseStudents=courseStudents;

}

var obj= new createCourse('Bloc Front-End Engineering', '4 weeks', ['Joe', 'Tim', 'Rob']);

for(x in obj)

{

txt+=obj[x];

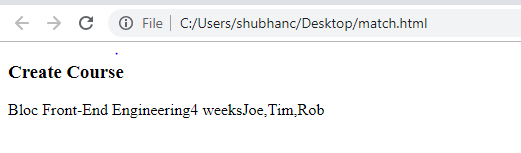
}

document.getElementById("demo").innerHTML = txt;

</script>

</body>

</html>



* Complete the addProperty function. This function should:
  + Take three arguments:
    - object: an object to add a properties to
    - newProp: a property that we want to add to the object
    - newValue: a value that we want the new property to have
  + If object doesn't already have a property named newProp, then add newProp with value of newValue to object
  + If object already has newProp, return the object argument.

For example:

addProperty({}, 'firstName', 'Jim') // should return { firstName: 'Jim' }

addProperty({firstName: 'Rob'}, 'firstName', 'Jim') // should return {firstName: 'Rob'}

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>

var txt="";

var x;

function addProperty(object, newProp, newValue){

if(!object.hasOwnProperty(newProp)){

object[newProp] = newValue;

}

return object;

}

var obj=new addProperty({}, 'firstName', 'Jim')

for(x in obj)

{

txt+=obj[x];

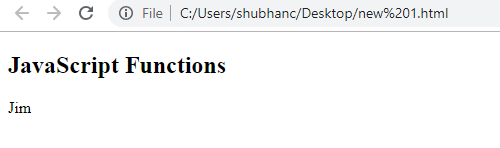
}

document.getElementById("demo").innerHTML = txt;

</script>

</body>

</html>



* Complete the formLetter function. This function should:
  + take one argument, a letter, which has three properties recipient, sender, and msg
  + combine the three properties into a single string with an additional greeting and closing
  + insert additional new lines between the greeting, message, and signature.

For example:

formLetter({ recipient: "James", sender: "Richard", msg: "Things are well." })

// should return "Hello James,\n\nThings are well.\n\nSincerely,\nRichard"

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>

function formLetter(recipient, sender,msg) {

this.recipient= recipient;

this.sender=sender;

this.msg=msg;

}

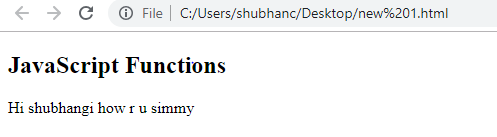
var s = new formLetter("shubhangi","simmy","how r u")

document.getElementById("demo").innerHTML = "Hi"+" "+s.recipient+" "+s.msg+" "+s.sender;

</script>

</body>

</html>



* Complete the canIGet function. This function should:
  + Take two arguments:
    - item: represents what the user wants to buy
    - money: represents how many dollars a user has
  + return true if a user can afford a given item according to the price chart below, and false otherwise:
    - 'MacBook Air' - $999
    - MacBook Pro' - $1299
    - 'Mac Pro' - $2499
    - 'Apple Sticker' - $1
  + Return false if the item is not in the above list of Apple products

Do this with 0 'if' conditions! (Hint: Place the above price table in an object).

For example:

canIGet('MacBook Air', 100) // returns false

canIGet('MacBook Air', 1000) // returns true

<!DOCTYPE html>

<html>

<h3>Create Course</h3>

<body>

<p id="demo"></p>

<script >

function canIGet(item, money)

{

var itemList = {

'MacBook Air': 999,

'MacBook Pro': 1299,

'Mac Pro': 2499,

'Apple Sticker': 1

};

return itemList[item]<=money?true:false;

}

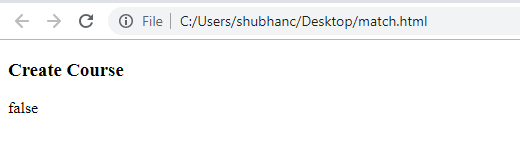
var obj=canIGet('MacBook Air', 100)

document.getElementById("demo").innerHTML = obj;

</script>

</body>

</html>



**String Assignments**

We'll be working on five functions for this exercise.

18. Complete the formLetter function. This function should:

* + Take three strings as arguments: the first name of the recipient, the sender's signature name, and the message of the letter
  + combine the three into a single string with additional greetings and closings
  + insert additional new lines between the greeting, message, and signature

For example:

Ex.formLetter("James", "Richard", "Things are well.");

...should return:

"Hello James,\n\nThings are well.\n\nSincerely,\nRichard"

<!DOCTYPE html>

<html>

<h3>Java Functions</h3>

<body>

<p id="demo"></p>

<script >

function formLetter(recipient,sender,msg)

{

return "Hello "+recipient+" , \\ n \\ n"+msg+" \\ n \\ n Sincerely ,\\n "+sender;

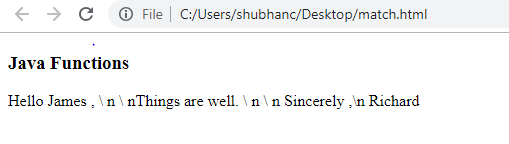
}

document.getElementById("demo").innerHTML = formLetter("James", "Richard", "Things are well.");

</script>

</body>

</html>



* Complete the sliceItAndCombineIt function. This function should:
  + take a string and four indices (numbers)
  + return a new string which is the concatenation of two substrings marked by the first and second index of each pair of indices. For example:

sliceItAndCombineIt("This is a Test", 0, 4, 5, 7) // returns "Thisis"

sliceItAndCombineIt("This is a Test", 0, 4, 1, 2) // returns "Thish".

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>

var sliceItAndCombineIt=function(myString, a,b,c,d){

return myString.substring(a,b)+myString.substring(c,d);

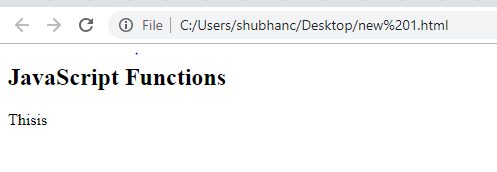
}

document.getElementById("demo").innerHTML = sliceItAndCombineIt("This is a Test", 0, 4, 5, 7);

</script>

</body>

</html>



* Complete the findFirstMatch function. This function should:
  + Take two strings as arguments. The first string is the one to search, the second is the one to search for.
  + Return the position (i.e. index) of the first match of string being searching for

For example:

findFirst("Roses are red", "re") // returns 7 (the position of the "re" in "are")

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>

var findFirstMatch =function(myString,str){

return myString.indexOf(str);

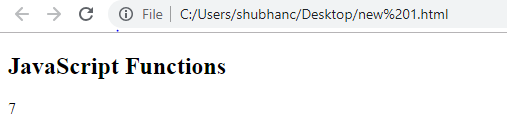
}

document.getElementById("demo").innerHTML = findFirstMatch("Roses are red", "re");

</script>

</body>

</html>



* Complete the findLastMatch function. This function should:
  + Take two strings as arguments. The first string is the one to search, the second is the one to search for
  + Return the position (a.k.a. the index) of the last match of string we're searching for For example:

findFirst("Roses are red", "re") returns 10 (the position of the "re" in "red")

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>

var findFirstMatch =function(myString,str){

return myString.lastIndexOf(str);

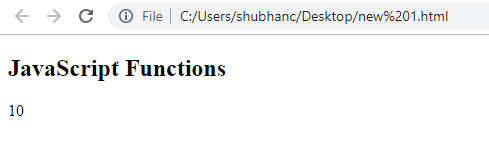
}

document.getElementById("demo").innerHTML = findFirstMatch("Roses are red", "re");

</script>

</body>

</html>



* Complete the substringBetweenMatches function. This function should:
  + Take two strings as arguments. The first string is the one to search, the second is the one to search for
  + Return the substring between the first match and the last match
  + Not include the first match or the last match in the returned substring For example:

findFirst("Roses are red, apples are really red.", "red") // returns ", apples are really "

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>

var substringBetweenMatches =function(myString,str){

var strlength=str.length;

var firstsub=myString.indexOf(str)+strlength;

var lastsub=myString.lastIndexOf(str);

return myString.substring(firstsub,lastsub);

};

document.getElementById("demo").innerHTML = substringBetweenMatches("Roses are red, apples are really red.", "red") ;

</script>

</body>

</html>

