Day 2

Yogesh Yadav

Converting a string to uppercase : Use toUpperCase() method

Example :
```
var upperCaseString = "JavaScript";
alert(upperCaseString.toUpperCase());
```

Output : JAVASCRIPT

Converting a string to lowercase : Use toLowerCase() method

Example :
```
var lowerCaseString = "JavaScript";
alert(lowerCaseString.toLowerCase());
```

Length of string javascript : Use length property

Example : alert("JavaScript".length);

Output : 10

Example :
```
var myString = "Hello JavaScript";
alert(myString.length);
```
Output : 16

Remove whitespace from both ends of a string : Use trim() method

Example :

```
var string1 = " AB ";
var string2 = " CD ";
var result = string1.trim() + string2.trim();
alert(result);
```

Output : ABCD

Replacing strings in javascript : Use replace() method. This method searches a given string for a specified value or a regular expression, replaces the specified values with the replacement values and returns a new string. This method does not change the original string.

Example : Replaces JavaScript with World

```
var myString = "Hello JavaScript";
var result = myString.replace("JavaScript", "World");
alert(result);
```

Output : Hello World

Example : **Perform a case-sensitive global replacement.** In this example, we are using a regular expression between the 2 forward slashes(//).
The letter g after the forward slash specifies a global replacement.
The match here is case sensitive. This means Blue(with capital B) is not replaced with green.

```
var myString = "A Blue bottle with a blue liquid is on a blue table";
var result = myString.replace(/blue/g, "green");
alert(result);
```

Output : A Blue bottle with a green liquid is on a green table

Example : **Perform a case-insensitive global replacement.**
The letters gi after the forward slash indicates to do global case-insensitive replacement.
Notice that the word Blue(with capital B) is also replaced with green.

```
var myString = "A Blue bottle with a blue liquid is on a blue table";
var result = myString.replace(/blue/gi, "green");
alert(result);
```

Output : A green bottle with a green liquid is on a green table

Substrings in JavaScript

- The following are the 3 methods in JavaScript that can be used to **retrieve a substring from a given string**.
  substring()
  substr()
  slice()

- **substring() method :** This method has 2 parameters start and end. start parameter is required and specifies the position where to start the extraction. end parameter is optional and specifies the position where the extraction should end. The character at the end position is not included in the substring. If the end parameter is not specified, all the characters from the start position till the end of the string are extracted. If the value of start parameter is greater than the value of the end parameter, this method will swap the two arguments. This means start will be used as end and end will be used as start.

Extract the first 10 characters

```
var str = "JavaScript Tutorial";
var result = str.substring(0, 10);
alert(result);
```

Output : JavaScript

If the value of start parameter is greater than the value of the end parameter, then start will be used as end and end will be used as start

```
var str = "JavaScript Tutorial";
var result = str.substring(10, 0);
alert(result);
```

Output : JavaScript

substr() method : This method has 2 parameters start and count. start parameter is required and specifies the position where to start the extraction. count parameter is optional and specifies the number of characters to extract. If the count parameter is not specified, all the characters from the start position till the end of the string are extracted. If count is 0 or negative, an empty string is returned.

Extract the first 10 characters

```
var str = "JavaScript Tutorial";
var result = str.substr(0, 10);
alert(result);
```

Output : JavaScript

If the count parameter is not specified, all the characters from the start position till the end of the string are extracted

```
var str = "JavaScript Tutorial";
var result = str.substr(11);
alert(result);
```

Output : Tutorial

slice() method : This method has 2 parameters start and end. start parameter is required and specifies the position where to start the extraction. end parameter is optional and specifies the position where the extraction should end. The character at the end position is not included in the substring. If the end parameter is not specified, all the characters from the start position till the end of the string are extracted.

Extract the first 10 characters

```
var str = "JavaScript Tutorial";
var result = str.slice(0, 10);
alert(result);
```

Output : JavaScript

If the end parameter is not specified, all the characters from the start position till the end of the string are extracted

```
var str = "JavaScript Tutorial";
var result = str.slice(11);
alert(result);
```

Output : Tutorial

What is the difference between substr and substring methods

The difference is in the second parameter. The second parameter of substring() method specifies the index position where the extraction should stop. The character at the end position is not included in the substring. The second parameter of substr() method specifies the number of characters to return.

Another difference is substr() method does not work in IE8 and earlier versions.

- **What is the difference between slice and substring**
  If start parameter is greater than stop parameter, then substring will swap those 2 parameters, where as slice will not swap.

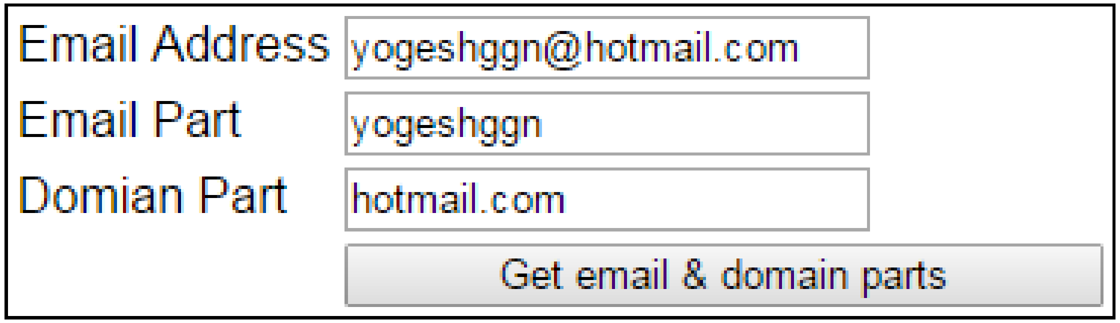
  Another method that is very useful when extracting a substring is indexOf() method. This method returns the position of the first occurrence of a specified value in a string. If the specified value is not present then -1 is returned.

  Example : Retrieve the index position of @ character in the email

```
var str = "yogesh@gmail.com";
var result = str.indexOf("@");
alert(result);
```

indexOf(), lastIndexOf() and substring()

Get email & domain parts



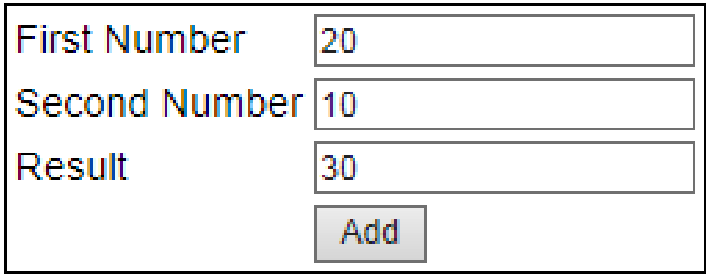
Email Address	yogeshggn@hotmail.com
Email Part	yogeshggn
Domian Part	hotmail.com

lastIndexOf() method returns the position of the last occurrence of a specified value in a string. Since it's job is to return the last index of the specified value, this method searches the given string from the end to the beginning and returns the index of the first match it finds. This method returns -1 if the specified value is not present in the given string.

```
var url = "http://www.google.com";
alert(url.lastIndexOf("."));
```

JavaScript to convert strings to numbers.

- 1. **parseInt()**
  2. **parseFloat()**
  3. **isNan()**

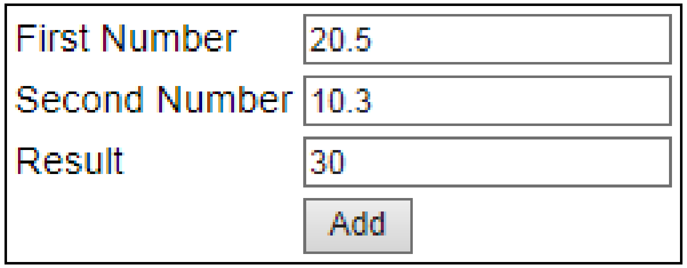


First Number	20
Second Number	10
Result	30
	Add

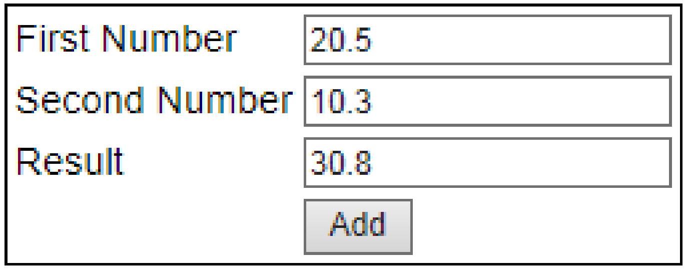
- function addNumbers()
- {
- var firstNumber = **parseInt**(document.getElementById("txtFirstNumber").value);
- var secondNumber = parseInt(document.getElementById("txtSecondNumber").value);
- document.getElementById("txtResult").value = firstNumber + secondNumber;
- }

- To retain the decimal places, use **parseFloat() function**.

- function addNumbers()
- {
-     var firstNumber = parseFloat(document.getElementById("txtFirstNumber").value);
-     var secondNumber =parseFloat(document.getElementById("txtSecondNumber").value);
-     document.getElementById("txtResult").value = firstNumber + secondNumber;
- }



First Number	20.5
Second Number	10.3
Result	30
	Add



First Number	20.5
Second Number	10.3
Result	30.8
	Add

- If you leave first number and second number textboxes blank or if you enter text instead of a number, and when you click the Add button, **NaN** is displayed in result textbox.

- **NaN in JavaScript stands for Not-a-Number.** In JavaScript we have isNaN() function which determines whether a value is an illegal number. This function returns true if the value is not a number, and false if not.

- if (isNaN(firstNumber))
- {
- alert("Please enter a valid number in the first number textbox");
- return;
- }

Conditional statements in javascript

- **JavaScript code is executed in a linear fashion from the first line to the last line.** If for some reason you want to interrupt this flow and execute certain statements, only, if certain condition is met, then we use conditional statements.

- **JavaScript has the following conditional statements**
  if
  if else
  if else if else
  switch
  ternary operator - shortcut for an if...else statement

If example

```
var userInput = Number(prompt("Please enter a number", ""));



if (userInput == 1)
{
    alert("You number is One");
}
if (userInput == 2)
{
    alert("You number is Two");
}
if (userInput == 3)
{
    alert("Your number is Three");
}
if (userInput != 1 && userInput != 2 && userInput != 3)
{
    alert("Your number is not between 1 and 3");
}
```

If else if example

```
var userInput = Number(prompt("Please enter a number", ""));


if (userInput == 1)
{
    alert("You number is One");
}
else if (userInput == 2)
{
    alert("You number is Two");
}
else if (userInput == 3)
{
    alert("Your number is Three");
}
else
{
    alert("Your number is not between 1 and 3");
}
```

Switch Statement

```
var userInput = Number(prompt("Please enter a number", ""));
switch (userInput)
{
    case 1:
        alert("You number is One");
        break;
    case 2:
        alert("You number is Two");
        break;
    case 3:
        alert("You number is Three");
        break;
    default:
        alert("You number is not between 1 and 3");
        break;
}
```

In general we need to have break statement after each case statement to ensure the program breaks out of the switch statement after executing statements present in a specific case.

What happens if there is no break in a switch statement

If there is no break statement the execution falls automatically to next case until a break statement is encountered or end of the program is reached.

When would you combine multiple case statements together

If you want the same piece of code to be executed for multiple cases you can combine them together as shown below. Case statement with no code in-between creates a single case for multiple values. A case without any code will automatically fall through to the next case.

```
var userInput = Number(prompt("Please enter a number", ""));
switch (userInput)
{
    case 1:
    case 2:
    case 3:
        alert("You number is " + userInput);
        break;
    default:
        alert("You number is not between 1 and 3");
        break;
}
```

Ternary operator

Boolean Expression ? Statements to execute if true : Statements to execute if false

```
var userInput = Number(prompt("Please enter a number"));
var message = "";
if (userInput % 2 == 0)
{
    message = "Your number is even";
}
else
{
    message = "Your number is odd";
}
```

```
var userInput = Number(prompt("Please enter a number"));
var message = userInput % 2 == 0 ? "Your number is even“
                                              : "Your number is odd";
```

Loops in JavaScript

- Here are the basic **loops in JavaScript**
  while
  do...while
  for

- let us print even numbers starting from ZERO till a number provided by the user at runtime.

- With for loop all three can be done in one place. To make this clear, look at the syntax of the for loop below.

```
for(initialization; boolean Condition; update variable)
{
   statements;
}
```

While loop

prints all even numbers from 0 till the target number that the end user has provided.

```
var targetNumber = Number(prompt("Please enter your target number", ""));
var start = 0;
while (start <= targetNumber)
{
    document.write(start + "<br/>");
    start = start + 2;
}
```

How does the while loop work

1. While loop checks the condition first
2. If the condition is true, statements with in the loop are executed
3. This process is repeated as long as the condition evaluates to true.

Loop Break

```
var targetNumber = Number(prompt("Please enter your target number", ""));
var start = 0;
while (start <= targetNumber)
{
    document.write(start + "<br/>");
    start = start + 2;


    if (start > 100) {
        break;
    }
}
```

What is the use of break statement

If break statement is used in a switch statement, it causes the execution to break out of that switch statement. In a similar fashion, if used in a loop, the loop ends.

loop continue

```
var start = 0;
while (start < 10)
{
    start = start + 1;


    if (start % 2 == 0)
    {
        continue;
    }


    document.write(start + "<br/>");
}
```

JavaSript while loop continue example
continue statement tells the JavaScript interpreter to skip remaining code that is present after this statement and start the next iteration of the loop.

do while loop

- **while loop :**
  1. while loop checks the condition first
  2. If the condition is true, statements with in the loop are executed
  3. This process is repeated as long as the condition evaluates to true.

  do while loop :
  1. do while loop checks its condition at the end of the loop
  2. This means the do...while... loop is guaranteed to execute at least one time.
  3. In general, do...while... loops are used to present a menu to the user

  What is the difference between while and do while in javascript
  1. while loop checks the condition at the beginning, where as do...while... loop checks the condition at the end of the loop
  2. do...while... loop is guaranteed to execute at least once, where as while loop is not.

```
var userChoice = "";
do
{
    var targetNumber = Number(prompt("Please enter your target number", ""));
    var start = 0;
    while (start <= targetNumber)
    {
        document.write(start + "<br/>");
        start = start + 2;
    }


    do
    {
        userChoice = prompt("Do you want to continue - Yes or No").toUpperCase();
        if (userChoice != "YES" && userChoice != "NO") {
            alert("Invalid choice. Please say, Yes or No");
        }
    } while (userChoice != "YES" && userChoice != "NO");


} while (userChoice == "YES");
```

- var targetNumber = Number(prompt("Please enter your target number", ""));
- for (var start = 0; start <= targetNumber; start = start + 2)
- {
-     document.write(start + "<br/>");
- }

- **Notice that all the 3 things, i.e**
  1. Variable initialization
  2. Boolean condition check and
  3. Updating the variable participating in the boolean expression **are all done at one place in the for loop. All these 3 things are optional in a for loop.**

- Variable initialization is optional in a for loop. Notice that we have moved variable intialization (var start = 0;) outside of the for loop head.

  var targetNumber = Number(prompt("Please enter your target number", ""));
- var start = 0;
- for (; start <= targetNumber; start = start + 2)
- {
-     document.write(start + "<br/>");
- }

- In this example we have moved the expression that updates the variable participating in the boolean expression (start = start + 2) from the head of the for loop, into it's body.

  var targetNumber = Number(prompt("Please enter your target number", ""));
- var start = 0
- for (; ;)
- {
-     if (start > targetNumber)
-     {
-         break;
-     }
-     document.write(start + "<br/>");
-     start = start + 2;
- }

Arrays

- **Arrays are collections and are ZERO indexed.** This means the first element is at index ZERO and the last element is at index arrayObject.length - 1. length property of the array object returns the size of the array.

- **The following JavaScript code creates an empty array.** length property returns 0 in this case.

```
var emptyArray = [];
alert(emptyArray.length);
```

Output : 0

- **Another way to create an array is by busing the Array constructor as shown below.** In this example we are setting the length of the array to 10.

  ```
  var myArray = new Array(10);
  alert(myArray.length);
  ```

  Output : 10

- **Retrieving first and last elements from the array using the array index**
  var myArray = [10, 20, 30];
- document.write("First element = " + myArray[0] + "<br/>");
- document.write("Last element = " + myArray[myArray.length - 1] + "<br/>");
-

  Output :
  First element = 10
  Last element = 30

- for (var i = 0; i < evenNumbersArray.length; i++)
- {
-     document.write(evenNumbersArray[i] + "<br/>");
- }

- **Populating an array in JavaScript :** There are several ways to populate an array in JavaScript.

Declaring an array first and then populating using the array index

```
var myArray = [];

myArray[0] = 10;
myArray[1] = 20;
myArray[2] = 30;

alert(myArray);
```

Output : 10, 20, 30

Declaring and populating the array at the same time

```
var myArray = [10, 20, 30];
alert(myArray);
```

Output : 10, 20, 30