Javascript

Lecture 2

Eng. Sahar E. Hassan

Javascript: Arrays

- Arrays are a collection of values grouped together.
- To define an array:

```
var students=[ ];          //initialize a new empty array
var students = new Array();
var fruit = ["apple", "orange", "banana" ];
```

- You can access the elements of the array by their index (zero-based)

Ex:

```
alert(fruit[0]); //this will alert "apple"
```

Javascript: Arrays (continue)

- Arrays have built in methods and properties to facilitate using them.
- To access any property or method you can precede it with a dot "."

Properties:

- Length:
  - The property "length" of any array will hold the total count of array elements.
  - Ex:

```
var fruits=["apple", "orange", "banana"];
alert(fruits.length);                    //this will alert 3;
```

Javascript: Arrays (continue)

Function	Description
indexOf(item)	Return the index of the specified item.
lastIndexOf(item)	Return the last index of the specified item.
reverse()	Reverse the array
push(item)	Add element to the end of the array, and return the new length of the array.
unshift(item)	Add element to the start of the array, and return the new length of the array.
pop()	Remove last element in the array, and return the removed element.
shift()	Remove first element in the array, and return the removed item.
concat(array)	Concatenate two arrays.
join(delimiter)	Concatenate arrays’ elements and separate them with delimiter(defalt: ”,”) and return string.

Javascript: Arrays (continue)

- **Splice method:** splice(index, count, item1, item2, …)

  1- Remove specific element:

  ```
  Var fruits = [‘apple’, ‘orange’, ‘mango’];
  fruits.splice(1,1);      // will remove from index 1 , 1 item. (orange)
  ```

  - Will return the removed element.

  2- Add elements at specified index:

  ```
  Var fruits = [‘apple’, ‘orange’, ‘mango’];
  fruits.splice(2, 0, ‘orange’);      //will add ‘orange’ element at index of 2
  ```

  - Will return an empty array.
  - In case of adding new items we put the count = 0

Javascript: String

- You can define a string using single quotes or double quotes.
- To include a double quote inside a string , you must escape it:

  Ex: var quote= "Ahmed said \"How are you?\" ";

- **String** can be treated like a "read-only" array (i.e you can access its elements using brackets but you cannot alter it)

```
var str="Hello World";

alert(str[1]);
```

Javascript: String (continue)

Function	Description
indexOf(item)	Return the index of the specified character or string.
lastIndexOf(item)	Return the last index of the specified character or string.
replace(text1,text2)	Replace string with another string, and return the new string. (i.e. the original string is not changed).
split(delimiter)	Split string into array of strings using a delimiter.
subStr(startIndex,length)	Extract part of string starting from index through the length.
subString(startIndex,endIndex)	Extract part of string from start index to end index.
toLowerCase()	Return the string in lowercase characters.
toUpperCase()	Return the string in uppercase characters.

Javascript: Loops

- You can use loops to repeat certain lines of codes for specific number of times.
- This can be used to iterate through collections of values (such as items in an array)
- Loops should always be limited (i.e. there is a point where loop ends) otherwise, it will cause the application to crash.

Javascript: for loop

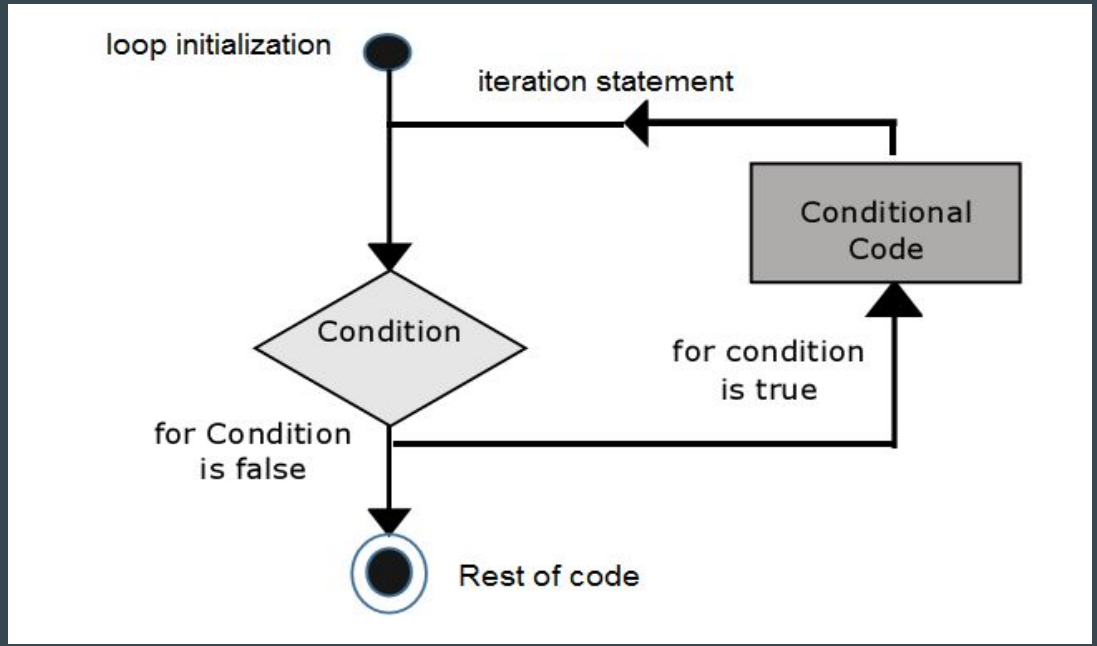
- For loops are used to iterate through a known number of iterations.

Syntax:

```
for(var i=0; i < limit; i++){
    //code to be repeated goes here
}
```

- It consists of 3 parts:
  - Loop initialization.
  - Test condition.
  - Iteration statement.

Javascript: for loop flow chart



Javascript: while loop

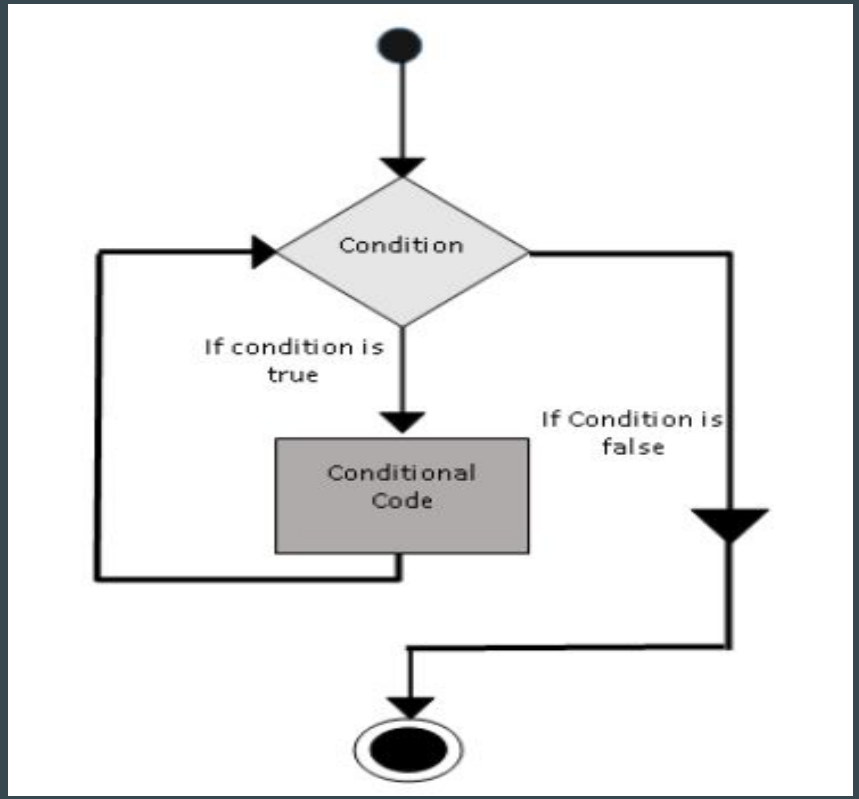
- While loop is usually used when the number of iteration is unknown.

Syntax :

```
while(condition){
    //code to be repeated goes here
    Increment or decrement statement
}
```

- Note that to prevent infinite loops, the code that runs inside the while loop should make the condition equal to false at some point.

Javascript: while loop flow chart

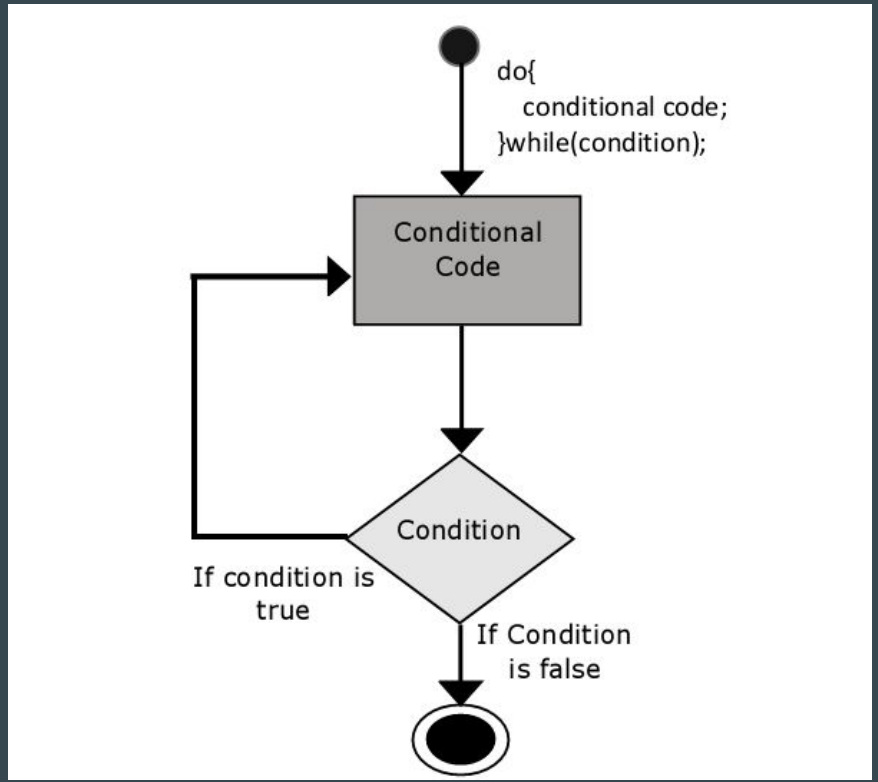


Javascript: do..while loop

- Do while is used to execute the code at least 1 time before checking for the condition.
- It operates as while loop.
- Syntax:

```
do{
        //code here is executed at least 1 time.
  Increment or decrement statement
}while(condition);
```

Javascript: do..while loop flow chart



Javascript: break & continue statements

- break is used to exit the current loop regardless of the condition.

```
Ex: while(a<100){
        b++;
            if(b==2){
                break;
            }
        }
```

- continue statement is used to exit only the current iteration and continue with the next iteration.

Javascript: functions

- You can create custom functions to group reusable code and be able to call those functions from anywhere in your program.
- To define a function:
  - Function definition:

```
function  myFunction(){

        //function code goes here

}
```

  - Function expression:

```
var  myFunctionVariable = function ( ) {
        // my function script
}
```

Javascript: functions (continue)

- To call a function:

  myFunction();

- Note that you can call the function before the definition of the function.
- Functions can also return a value.
  - Ex:

```
function  myFunction (){
        var result = "hello world";
        return result;
}
```

Javascript: functions (continue)

- Functions can accept parameters to be passed to it.
- To define a function that takes parameters:

```
Ex: function  add(a,b){
        var result = a+b;
        return result;
    }
    alert(add(3,4));
```

- Javascript will not complain if you pass an extra parameter (it will be ignored).
- If you did not pass an argument, its value will be undefined;

Javascript: scope

- All variables in javascript until ECMAscript5 are belong to global scope.
- Functions are the only thing that can define new variables scope.
- Variables defined inside functions belong to local scope of the function.
- Inside functions you can still access the global variables.

Ex:

```
var myText = "Hello World";

add(3,4);
function add(a,b){
        alert(myText);
        var result = a+b;
        alert(result);
}
alert(result);        //error
```

Javascript: Hoisting

- Javascript will automatically move the variables and functions declarations to the start of their scope which called ‘Hoisting’.
- Hoisting allows using of variables and functions before they are declared.
- But note that it will only move the declarations but NOT initialization.
- Function declarations are hoisted, however functions that are assigned to variables are not hoisted.
- Ex:

```
alert(name); //this will alert undefined

var name="Ahmed"

alert(age); //this will throw an error

showHello(); //this will successfully call the function and alert "Hello";

function showHello(){

        alert("Hello");

}
```

Javascript: Lab 2

- Exercise 1:
  - Ask the user to enter 5 names (using prompt) and store them into an array.
  - Ask the user to enter a name to search for.
  - If the name is found, alert the order of that name (ex: first, second, third, fourth, or fifth).
  - If the name is not found, show a message that "The name is not found“.

- Exercise 2 :
  - Ask the user to enter a string.
  - Count the number of words the string has.

    (ex: “Ahmed goes to school by bus” □ 6 words)

Javascript: Lab 2

- Exercise 3:
  - Ask the user to enter a string.
  - Ask the user to enter 2 characters.
  - If both characters exist on the string:
    - Display the string between those 2 characters in the first string.
  - Otherwise: display an message that the character does not exist.

    Ex: the user enters :”Javascript is awesome”. Then he enters “v” and “w”.

    The output should be “ascript is a”

Javascript: Lab 2

- Exercise 4:
  - Ask the user to enter a number
  - Write code to calculate the factorial of the number (factorial 5 = 5*4*3*2*1)
  - Alert the result to the user
  - Ask the user if he wants to check for another number ,keep asking the user to enter the number until he clicks no
- Exercise 5:
  - Create a function that accepts a string and returns the number of vowels in that string (vowel characters are : a, e, o, u, i)
  - Ask the user to enter a string
  - Count the number of vowels in that string using the created function.

Javascript: Lab 2

- Exercise 6:

  Task 1:

  - Ask the user to enter names of employees in a comma separated format.
  - Ask the user to enter the salaries of the employees in a comma separated format.
  - If the number of employee names does not match the number of salaries entered, show an error that the count does not match.

  Task 2:

  - Alert the names of the employees and their salaries in this form:

    Name: Ahmed , Salary: 500
  - Alert the data of the employee with the highest salary

  Task 3

  - Ask the user to insert a name of an employee
  - Search for that employee in the list and return the data of that employee
  - Bonus: Make the search case-insensitive