* **Javascript( page 2)**
* **Where we will use javascript(page 2)**
* **Arrays**
* **Cookies (page6)**
* **Data types(page 5** primitive and non primitive **)**
* **Display Possibilities(page 3** innerHTML, document.write(),window.alert(),console.log(
* **Events(page7)**
* **Functions(page5)**
* **Uses of function**
* **Advantages of function**
* **Function with argument**
* **Hoisting(page8)**
* **Nan(page8)**
* **Statements in javascript(page3)**
* **strict mode(page4)**
* **setInterval( )(page7)**
* **typeof operator(page8)**
* **Variables(page3)**
* **closures**

**What is javascript?**

* **Javascript** is a dynamic computer programming language. ... It is an interpreted programming language with object-oriented capabilities. **JavaScript** was first known as LiveScript, but Netscape changed its name to **JavaScript**, possibly because of the excitement being generated by Java.Case Sensitivity:
* JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

**JavaScript is a solution** of client side dynamic pages.

JavaScript is an object-based scripting language that is lightweight and cross-platform.

JavaScript is not compiled but translated. The JavaScript Translator (embedded in browser) is responsible to translate the JavaScript code.

Where JavaScript is used

**JavaScript is used** to create interactive websites. It is mainly used for:

* Client-side validation
* Dynamic drop-down menus
* Displaying date and time
* Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
* Displaying clocks etc.

**JavaScript Can Change HTML Styles (CSS)**

* Changing the style of an HTML element, is a variant of changing an HTML attribute:

**JavaScript Display Possibilities:**

JavaScript can "display" data in different ways:

* Writing into an HTML element, using **innerHTML**.
* Writing into the HTML output using **document.write()**.
* Writing into an alert box, using **window.alert()**.
* Writing into the browser console, using **console.log()**.

**Statements in java scripting:**

A **computer program** is a list of "instructions" to be "executed" by the computer.

In a programming language, these program instructions are called **statements**.

JavaScript is a **programming language**. JavaScript statements are separated by **semicolons**:

The line which is ended with ; is called statement

**Js math objects:**

Allows us to to perform mathematical task on numbers.

**Math.min() , Math.max.() ,Math.round(),Math.ceil()**

**Javascript variables:**

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore( \_ ), or dollar( $ ) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

**javaScript local variable:**

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only

**<script>**

function abc(){

var x=10;//local variable

}

**</script>**

**Javscript global variable:**

A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable

**<script>**

var value=50;//global variable

function a(){

alert(value);

}

function b(){

alert(value);

}

**</script>**

To declare JavaScript global variables inside function, you need to use **window object**

function m(){

window.value=100;//declaring global variable by window object

}

function n(){

alert(window.value);//accessing global variable from other function

}

**JS strict mode:**

The “use strict” directive is new in javascript 1.8.5

With strict mode you can’t use undeclared variables

Strict mod edeclared by adding “use strict ” beginning of the script or a function.

* If we declare beginning of the script it has **global scope(**all code in the script will execute in strict mode**)**
* F we declare inside a function , it has local scope(only the code inside the function is strict mode)
* Using a variable , without declaring it is not allowed.
* Using a object , without declaring it is not allowed.
* Deleting a variable or object is not allowed
* Deleting a function is not allowed
* Duplicating a parameter name is not allowed
* Octal numeric literals are not allowed
* Writing to a read-only property is not allowed
* Deleting an undeletable property is not allowed
* The string "eval" cannot be used as a variable

# Data Types:

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because **it is dynamically used by JavaScript engine**. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

**primitive data types:**

There are five types of primitive data types in JavaScript. They are as follows

**String:** represents sequence of characters

**Number:** represents numeric values e.g. 100 Boolean, Undefined, Null

**Boolean:** represents boolean value either false or true

**Undefined:** represents undefined value

**Null:** represents null i.e. no value at all

**non-primitive data types**

The non-primitive data types are as follows:

**Object,array,RegExp**

# JavaScript Functions

**JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.

**Advantage of JavaScript function**

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don’t need to write many lines of code each time to perform a common tas

**Function with argument**:

Let’s see the simple example of function in JavaScript that does not has arguments.

**<script>**

function msg(){

alert("hello! this is message");

}

**</script>**

**<input** type="button" onclick="msg()" value="call function"**/>**

**Function with argument:**

**<script>**

function getcube(number){

alert(number\*number\*number);

}

**</script>**

**<form>**

**<input** type="button" value="click" onclick="getcube(4)"**/>**

**</form>**

**Cookies:** Cookies are small text files that a browser stores in the visitors computer. A cookie is a basically a string of name-value pairs separated by semi-colons

Cookies were invented to solve the problem "how to remember information about the user":

* When a user visits a web page, his name can be stored in a cookie.
* Next time the user visits the page, the cookie "remembers" his name.

**Create a Cookie with JavaScript**

* JavaScript can create, read, and delete cookies with the **document.cookie**  property.

Ex: document.cookie = "username=John Doe";

**add an expiry date** (in UTC time). By default, the cookie is deleted when the browser is closed:

document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";

**Read a Cookie with JavaScript:**

var x = document.cookie;

* **document.cookie** will return all cookies in one string much like: cookie1=value; cookie2=value; cookie3=value;

**Delete a Cookie with JavaScript:**

Just set the expires parameter to a passed date:

document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

**SetInterval( ):** The setInterval() method calls a function or evaluates an expression at specified intervals (in milliseconds).

The setInterval() method will continue calling the function until [clearInterval()](https://www.w3schools.com/jsref/met_win_clearinterval.asp) is called, or the window is closed.

The ID value returned by setInterval() is used as the parameter for the clearInterval() method.

**Ex:**

myVar = setInterval(alertFunc, 3000);

**Events:**

HTML events are **"things"** that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

**1.onclick:** The user clicks an HTML element

**2 .onchange:** An HTML element has been changed

**3.onmouseover:** The user moves the mouse over an HTML element

**4.onmouseout:** The user moves the mouse away from an HTML element

**5.onkeydown:** The user pushes a keyboard key

**6.Onload:** The browser has finished loading the page

**oncontextmenu, ondbclick, onmouseenter, onmouseleave, onmouseleave**

**typeof operator:** The typeof operator is used to get the data type (returns a string) of its operand. The operand can be either a literal or a data structure such as a variable, a function, or an object. The operator returns the data type.

**Syntax:**

var index = 8;

var result = (typeof index === 'number');

**NaN function( ):**

The **isNaN**() **function** determines whether a value is an illegal number (Not-a-Number). This **function** returns true if the value equates to **NaN**. Otherwise it returns false. This **function** is different from the Number specific Number.**isNaN**() method.

Syntax:

var res = "";

res = res + isNaN(123) + ": 123<br>";

**Arrays:**

An array is a special variable, which can hold more than one value at a time.

JavaScript arrays are used to store multiple values in a single variable.

**Join():**The **join()** method also joins all array elements into a string.

**toString()** : The JavaScript method **toString()** converts an array to a string of (comma separated) array values.

**pop()** :The **pop()** method removes the last element from an array:

**push()** :The **push()** method adds a new element to an array (at the end)

**shift()** :The **shift()** method removes the first array element and "shifts" all other elements to a lower index

**delete**::elements can be deleted by using the JavaScript operator **delete**

**concat():**The **concat()** method creates a new array by concatenating two arrays:

**Hoisting:**

Hoisting is JavaScript's default behavior of moving declarations to the top.

In JavaScript, a variable can be declared after it has been used.

In other words; a variable can be used before it has been declared.

Hoisting is (to many developers) an unknown or overlooked behavior of JavaScript.

If a developer doesn't understand hoisting, programs may contain bugs (errors).

To avoid bugs, always declare all variables at the beginning of every scope.

Since this is how JavaScript interprets the code, it is always a good rule.

**Closures:** JavaScript variables can belong to the **local** or **global** scope.

Global variables can be made local (private) with **closures**.

Exam:

|  |
| --- |
| function showName (firstName, lastName) { |
|  | ​var nameIntro = "Your name is "; |
|  | // this inner function has access to the outer function's variables, including the parameter​ |
|  | ​function makeFullName () { |
|  | ​return nameIntro + firstName + " " + lastName; |
|  | } |
|  | ​ |
|  | ​return makeFullName (); |
|  | } |
|  | ​ |
|  | showName ("Michael", "Jackson"); // Your name is Michael Jackson |