WEB PROGRAMMING USING PHP

5TH SEMESTER BCA

UNIVERSITY OF CALICUT

MISNA VK
ASST.PROFESSOR
DEPT.OF CS
NASRA CAS,TIRURKA

MODULE 2: JAVASCRIPT

- Javascript: Introduction, Client side programming, script tag, comments, variables.
- Including JavaScript in HTML: head, body, external.
- Data types, Operators: Arithmetic, Assignment, Relational, Logical.
- Conditional Statements, Loops, break and continue.
- Output functions: write, writeln, popup boxes: prompt, alert, confirm.
- Functions: Built-in Global Functions: alert(), prompt(), confirm(), isNan(), Number(), parseInt().
- User Defined Functions, Calling Functions with Timer
- Events Familiarization: onLoad, onClick, onBlur, onSubmit, onChange,
- Document Object Model (Concept). Objects: String, Array, Date

Javascript

- Client-side object-based scripting language
- With JS we can create intelligent web pages that verify input , calculate it and make presentation decision based on that
- JS is an interpreted language
- Scripts are interpreted line by line by a JS interpreter , which is an integrated component of web browser

Functions of jS

- Read and write elements from documents
- Manipulate or move text
- Create pop-up windows
- Perform mathematical calculations on data
- Reacts to events like user’s rolling over an image or clicking a button
- Retrieve current date n time from user’s computer or last time a document was modified
- Determine user’s screen size,browser version,or screen resolution
- Alerting users if they entered wrong information into a form or if they press a wrong button

Origin

- Developed by ***Brendan Eich*** of Netscape under name MOCHA
- Later renamed into ***Livescript*** and released on september1995
- Renamed to ***javascript*** and released on december 4,1995
- It was deployed in the netscape browser version 2
- Microsoft introduced a clone of javascript called jscript in internet explorer 3.0
- JS is now the standard client side scripting language and supported by all web browsers available today
- JS is officially maintained by ECMA(European Computer Manufacturers Association) as ECMAScript
- ES2015-ECMAScript2015 (old name-ES6) is the latest version

Including JS in web pages

- In the <head> element
- In the <body> element
- In an external file

<script language="JavaScript" type="text/JavaScript">

Code here

</script>

JS in <head>

```
<!Doctype html>
<html>
<head>
<title>JS page</title>
<script language="JavaScript”  type="text/JavaScript">
alert("welcome to JS");
</script>
</head>
<body>
<h2>This scripts pops up a message box</h2>
</body>
```

</html>

JS in <body>

```
<!Doctype html>
<html>
<head>
<title>JS page</title>
</head>
<body>
<h2>This scripts pops up a message box</h2>
<script language="JavaScript”  type="text/JavaScript">
alert("welcome to JS");
</script>
</body>
```

</html>

JS in an External File

- **hello.js**

```
function sayHello(){
alert(“welcome!”);
}
```

- **Ext.html**

```
<html>
<head>
<script src=“hello.js”></script>
</head>
<body>
<button type=“button” id=“mb” onclick=“sayHello()”>click me</button>
```

```
</body>
</html>
```

<noscript>

- Javascript-aware browsers ignore contents of <noscript>
- Javascript- unaware browsers display enclosed message within<noscript></noscript> tags

JS comments

- Single line comments
  - //comment
- Multiline comments
  - /*comments*/

variables

- Declare variables using ***var*** keyword
- Syntax:
- **var varname=value;**
- var x=8;
- var y=x+2;
- var name=“alan”;
- var isMarried=false;
- Javascript is also known as untyped language.
- This means, that once a variable is created in javascript using the keyword var, we can store any type of value in this variable supported by javascript

variables

- ES6 introduces 2 new keywords ***let*** and **const**
- ***const*** is a variable type assigned to data whose value cannot and will not change throughout the script.
- *let name="harry";*
- *const PI=3.14;*
- *const and let* have block-level scope
- *var* declare function scoped variables

Naming conventions for variables

- Name must start with letter , underscore or $ sign
- Cannot start with a number
- Cannot contain whitespaces
- Name cannot be a js keyword or reserved word
- Variable names are case sensitive

variables

```
<html>
<head>
<title>variables</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
//Declaring multiple variables
var name="shyam", age=23,isMarried=false;
//printing variable values
document.write(name +"<br>");
document.write(age +"<br>");
</script>
</body>
</html>
```

```
document.write(isMarried );
```

Data types in JS

- **Primitive datatypes**
  - Can hold only one value at a time
    - ***String***
    - ***Number***
    - ***Boolean***
- **Composite Data types**
  - Can hold collections of values and more complex entities
    - ***Object***
    - ***Array***
    - ***Function***
- **Special data type**
  - ***Undefined***

- ***Null***

String

- Used to represent textual data
- Enclosed in single or double quotes
- var a=‘hi there’;
- var a=“hi there”;
- var a=“let’s have a cup of coffee”;
- var b=‘he said “hello” and left’;
- var c=‘we\’ll never giveup’;

Number

- Represent +ve or –ve numbers with or without decimal place
- Or numbers written using exponential notation
- var a=25;
- var b=90.8;
- var c=4.25e+6;
- var d=4.25e-6;
- Number data type includes special values
  - Infinity-(dividing a nonzero number by 0)
  - -infinity
  - NaN(Not a Number)-invalid or undefined mathematical operation

Boolean

- True or false
- var isReading=true;
- Boolean values come as result of comparisons in a program
- var a=5,b=2,c=10;

```
alert(b>a);//output:false
alert(c>b);//output:true
```

Undefined

- A variable is declared and not assigned a value
- var a;

  var b=“hello”;

  alert(a);//undefined

  alert(b);//hello

NULL

- Means that there is no value
- A variable can be emptied of its current contents by assigning it a null value
- var a=null;

  alert(a); //output:null;

Object

- Complex data type that stores collections of data
- Object contains properties defined as key-value pair
- Property key is a string
- value can be any data type like string,boolean,array etc
- var emptyObject={};
- var person={

```
        "name":"Amar",
        "surname":"Akbar",
        "Age":25
};
```

Array

- Type of object used for storing multiple values in single variable
- Each value in array has a numeric position(index)
- First array element is arr[0]
- var color=[“red”,”pink”,”blue”];

alert(color[0]); //output red

function

- Callable object that executes a block of code
- var greeting=function(){

  return “hello world”;

  }
- Functions can be stored in variables,objects and arrays
- Functions can be passed as arguments to other functions and can be returned from functions

JS Operators

- Arithmetic
- Assignments
- String
- Increment/decrement
- Logical
- Comparison

Arithmetic operators

- +
- -
- *
- /
- %

Assignment Operators

- =
- +=
- -=
- *=
- /=
- %=

String

- + concatenation
- += concatenation assignment

Increment/Decrement

- ++x pre-increment
- x++ post-increment
- --x pre decrement
- X-- post decrement

Logical Operators

- && AND
- || OR

- ! NOT

Comparison operators

- == equal
- === identical(Strict equality Operator)
  - var a=true;
  - var b=1;
  - a==b;//true
  - a===b//false
- !=
- !== not identical
- <
- >
- <=

- >=

JS conditional/selection statements

- **if**

Synta

x:

if(condition)

{ //code to be executed}

- **if..else**

if(condition) {//true code}

else{//false code}

JS conditional/selection statements

- **switch ..case**

```
switch(x){
case value1: //code to be executed if x===value1
break;
case value2: //code
break;
….
default:// code if x is different from all values
}
```

Ternary operator

- Shorthand of if else operator
- var result=(condition)? value1:value2;
- let result = (marks >= 40) ? 'pass' : 'fail';

loops

- while
- do..while
- for
- for..in
- for..of

loops

- **while**

```
while(condition){
//code
}
```

- **Do..while**

```
do{
//code
}while(condition);
```

loops

- **for**

```
for(initialization;condition;increment)
{
//code
}
```

for..in loop

- Special loop that iterates over properties of an object or the elements of an array
- for(variable in object){

//code to be executed

}

- Variable is a string that contains name of current property or index of current array
- var fruits=["apple","orange","grapes","banana"];
- for(var i in fruits)

{

document.write(fruits[i]+"<br>");

}

For..of Loop

- ES6 introduced for..of loop
- Iterate over arrays or other objects
- let letters=[“a”,”b”,”c”,”d”];

```
for(let i of letters){
document.write(i+”,”)};// o/p: a,b,c,d
```

- let greet=“hello”;

```
for(let i of greet){
document.write(i+”,”);  //o/p: h,e,l,l,o
}
```

Jump statements

- Break
  - Break or exit a loop
- Continue
  - Breaks one iteration and continues with next iteration
  - for(var i=0;i<5;i++){

```
if(i===3){continue;}
document.write(i +”,”);    // o/p:   0,1,2,4
}
```

JS functions

- Provide a way to create reusable code packages which are more portable and easier to debug
- **Reduces repetition of code within a program**
- Makes the code much easier to maintain
- **Makes it easier to eliminate errors**

Defining and calling a function

- function functionName() {

//code to be executed

}

function sayHello(){

alert(“hello”);

}

- Calling a function
  - functionName();

Adding parameters to Functions

- function functionName(parameter1,parameter2){

```
// code to be executed
}
```

- function displaySum(num1,num2){

```
var total=num1+num2;
document.write(total);
}
//calling function
displaySum(6,20); // o/p : 26
```

Returning values from function

- A function can return a value using **return** statement
- Value may be of any type including array or object
- return value;

```
function createGreeting(name){
return "hello, " + name;
}
var result=createGreeting("Amar");
alert(result);   // hello Amar
```

Calling functions with Timer

- Timer allows to execute JS functions after a specified period
- Using timer ,
  - we can run a command at specified intervals,
  - run loops repeatedly at a predefined time,
  - Synchronise multiple events in a particular time span

- To use timer, JS provides various methods
- The **setTimeout()** method : execute code at a specified interval
  - Syntax : setTimeout(function, delayTime)
    - Function specifies method that timer calls
    - delayTime specifies no of milliseconds to wait before executing function
- The **clearTimeout()** method: Deactivates or cancels the timer that is created using setTimeout() method
  - Syntax: clearTimeout(timer)
- The **setInterval()** method: executes a function after specified time interval
  - Syntax : setInterval(function, intervalTime)
    - intervalTime specifies time interval b/w fn calls
- The **clearInterval()** method: Deactivates timer that is set using setInterval()
  - clearInterval(timer)

Function/variable scope

- Scope:- Area within which a function and its variables are accessible
  - Global scope:- function can be accessed from anywhere in a program
  - Local:- accessed only within specific area of program
- By default variables declared within function have local scope

Global functions

Built-in global functions of JS

- **alert()**
- **Prompt()**
- **Confirm()**
- **isFinite():-** returns true or false indicating whether the argument passed is finite or infinite
- **isNaN():-** Determines whether or not a value is an illegal number

Global functions

- **parseInt:-** Extracts a number from the beginning of a string
- **parseFloat:-** parses the string and returns first floating point value in thestring
- **Number():-** converts a value of an object into a number

Javascript output

- **Writing o/p to browser console**
- Write data to console using ***console.log()***
- console.log("hello world");
- **Writing output using write() &writeln()**

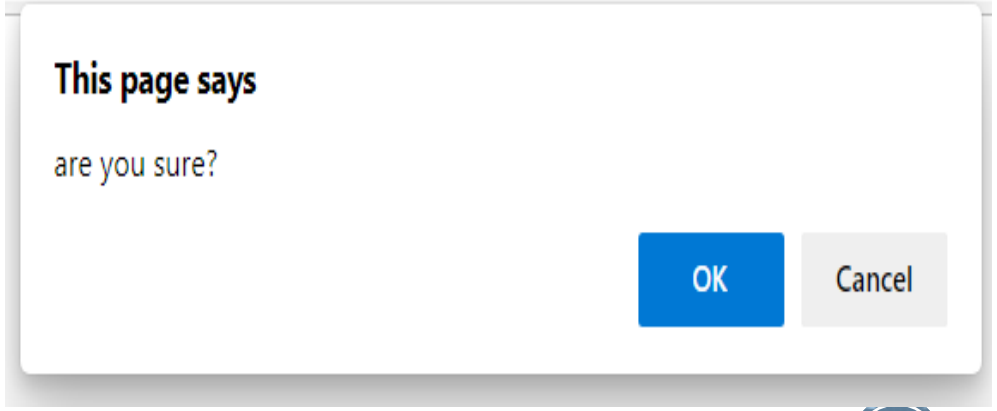
- **document .write()**
- document.write(“hello”);
- document.write(name+ “ ”)
- **document .writeln()- adds newline character**

JS Dialog or Popup Boxes

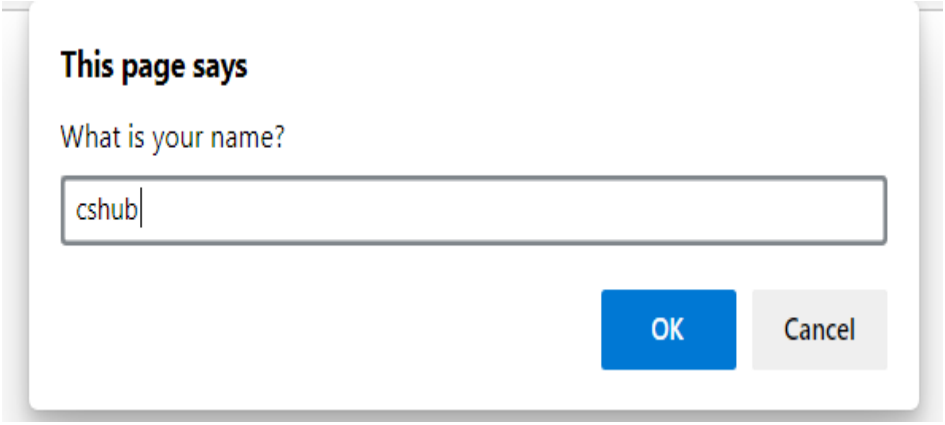
- Dialog box is a Window that displays a message along with buttons
  - Alert box
  - Confirm box
  - prompt box
- **Alert box**
  - Displays an alert message or error message
  - Contains an OK button
  - alert(“hello”);

- **Confirm box**
- Allows user to confirm or cancel an action
- It includes an OK and Cancel button
- It returns boolean value true or false depending on user clicks OK and Cancel button

```
var result= confirm("are you sure?");
if(result){
document.write("You clicked OK button");
}
else{
document.write("You clicked Cancel button");
}
```



- **Prompt Box**
- Used to prompt the user to enter information
- Includes a text input field ,an OK and a cancel button
- prompt(“what is your name?”);
- Value returned by prompt is always a string
- To use returned value as number,it must be converted
- var age=Number(prompt(“what is your age?”));



JS Events

- Event is something that happens when user interacts with the webpage
- When an event occur a JS event handler detects them and performs a specific task
- <button type=“button” onclick=“alert(‘Hello world’) “>click</button>
- Events classified into 4
  - Mouse events
  - Keyboard events
  - Form events
  - Document/window events

Mouse events

- Click event(**onclick**)
- Contextmenu event(**oncontextmenu**)
  - when user clicks right mouse button
- Mouseover Event(**onmouseover**)
  - When user moves mouse pointer over an element
- Mouseout Event(**onmouseout**)
  - User moves the mouse pointer out of an element

Keyboard Events

- Keydown Event**(onkeydown)**
  - User presses down a key
- Keyup Event(**onkeyup**)
  - User releases a key
- Keypress Event (**onkeypress**)
  - Press down a character key

Form Events

- Focus event (**onfocus**)
  - User gives focus to an element on a webpage
- Blur event(**onblur**)
  - Takes the focus away from a form element or a window
- Change event(**onchange**)
  - User changes the value of a form element
- Submit event(**onsubmit**)
  - Only occurs when a user submits a form on a web page

Document/Window Events

- The load event(**onload**):
  - When a webpage finished loading in the webbrowser
- The resize event (**onresize**):
  - User resizes the browser window

Built-in JS Objects

- **String Object**
  - Used to deal with strings of text
  - Instance of a string object can be created using
    - var myString = new String(‘Here is some big text’);
- document.write(myString.big())
- Main property of string object is **length**
- alert(myString.length)

Methods of string object

Method
anchor(name)
Big()
Bold()
charAt(index)
fontcolor(color)
Fontsize(size)
Italics()
Link(targetURL)
toLowerCase()
toUpperCase()
Substr(start,[length])

```
<!Doctype html>
<html>
<head>
<title>JS string object</title>
</head>
<body>
<script type="text/javascript">
var myString=new String('hello javascript string');
myString=myString.substr(6,[10]);
//myString=myString.substring(6,16);
myString=myString.toUpperCase();
document.write(myString);
</script>
</body>
</html>
```

Date object

- To work with date and time
- const d = new Date();
- It can use any of the four parameters
  - Milliseconds: value should be number of milliseconds since 01/01/1970
  - dateString: any date string recognised by parse() method
  - Yr_num,mo_num,day_num
  - Yr_num,mo_num,day_num,hr_num,min_num,seconds_num,ms_num:
- var bd=new Date(8298400000); //Tue Apr 07 1970 06:36:40 GMT+0530 (India Standard Time)
- const d = new Date(2018, 3, 24, 10, 33, 30, 0); // Tue Apr 24 2018 10:33:30 GMT+0530 (India Standard Time)
- const d = new Date("2022-03-25"); // Fri Mar 25 2022 05:30:00 GMT+0530

(India Standard Time)