JavaScript - Used for controlling behavior of web page.

Uses:

- Event Handling
- Accessing DOM elements
- Modifying/Adding/Deleting DOM elements
- Controlling CSS
- Client side validation

Where to write Javascript code?

```
<html>
    <head>
        <script type="text/javascript">
            function add( ) { … }
        </script>
    </head>
</html>
```

2. Writing js code separately and including that in .html file

1.js file

```
function f1( ) { … }
function f2( ) { … }
```

1.html file

```
<html>
    <head>
        <script type="text/javascript" src="1.js"> </script>
    </head>
</html>
```

Uncaught ReferenceError: welcome is not defined

at HTMLButtonElement.onclick (1.html:8)

Regular expression:

+ : 1 or more occurrences
* : 0 or more occurrences
? : 0 or 1 occurrence
{m} - exact m occurrences
{m, n} - range of occurrences
^: beginning of input
$: end of input

[0-9] or \d
[a-z]
[a-zA-Z]+
[0-9]{10} \d{10}

[a-z]+ => 1 or more occurrences of a-z

[A-Z][a-z]* => 1st letter capital and rest of in lower case

RegEx for mobile number:

^\d{3}\s?\d{3}\s?\d{4}$

https://regexr.com/

JQuery:

AJAX : Asynchronous JavaScript & XML

Ajax Request : XHR(XML HTTP Request)

{"results":[{"gender":"male","name":
{"title":"mr","first":"cesar","last":"soto"},"location":
{"street":"3254 avenida de américa","city":"jerez de la
frontera","state":"cantabria","postcode":
13167,"coordinates":
{"latitude":"-21.5632","longitude":"-159.5392"},"timezo
ne":
{"offset":"+5:45","description":"Kathmandu"}},"email":"
cesar.soto@example.com","login":
{"uuid":"3ec50630-2280-4d45-
a5df-8ee051fe706d","username":"tinyzebra591","pass
word":"flint","salt":"jkGd1qt5","md5":"3fc5051f25ecd4b
2599de96b398d2633","sha1":"d283bf8795d4231175e8
bf38cd5bcec6b673e5e7","sha256":"4f07524ae2cd725c
71623dcf91bc37076afe9932446909c0295f2743a4220b
96"},"dob":{"date":"1992-06-05T09:01:55Z","age":
26},"registered":{"date":"2016-10-21T20:15:48Z","age":
1},"phone":"931-994-048","cell":"631-873-185","id":
{"name":"DNI","value":"41584888-Z"},"picture":
{"large":"https://randomuser.me/api/portraits/men/
0.jpg","medium":"https://randomuser.me/api/portraits/
med/men/0.jpg","thumbnail":"https://randomuser.me/
api/portraits/thumb/men/0.jpg"},"nat":"ES"}],"info":
{"seed":"4f2d291603bef09d","results":1,"page":
1,"version":"1.2"}}

```
$.ajax({
  url: 'https://randomuser.me/api/',
  dataType: 'json',
  success: function(data) {
    console.log(data);
  }
```

});

JDBC (Java Database Connectivity)

DBMS: Database Management System
It is a s/w which manages database.
(MySQL)

Database : Collections of related tables. (college)

Table : Collection of records.(student)

CRUD operations
C: Create (insert)
R: Retrieve (select)
U: Update (update)
D: Delete (delete)
DAO : Data access object
CRUD operations of Student will be defined
“StudentDao.java” file.

POJO class : Plain Old Java Object

```
class Student {
    private int rollNo;
    private String name;
    private float marks;
    //constructors (default + parameterized)
    //public getter + setter methods
    public int getRollNo() { return this.rollNo; }
    public void setRollNo(int r) { this.rollNo = r; }
}
```

https://dev.mysql.com/get/Downloads/

MySQLInstaller/mysql-installer-community-8.0.11.0.msi

JDBC Driver (.jar) (Jdbc connector)

create table student(id integer primary key, name varchar(20), marks float);

insert into student values(2,'John',44);
select * from student;
delete from student where id=2;
update student marks=marks+5 where id = 2;

JDBC Steps:

1. Importing “java.sql.*” package.
2. Load JDBC driver class. (This class belongs to JDBC jar file which is included in project)
   Driver class name varies from DBMS to DBMS.
   e.g. for MySQL it is “com.mysql.jdbc.Driver”
   for PostgreSQL : “org.postgresql.Driver”
   for Oracle : “oracle.jdbc.driver.OracleDriver“
   Syntax of loading a class:
   Class.forName(“com.mysql.jdbc.Driver”)
3. Establishing connection with DBMS server. This requires 3 fields(a.JDBC URL,b.userName, c.password)
   a. JDBC URL: This is again specific to DBMS.
   for mysql: “jdbc:mysql://<dbms-ip>/<db-name>”
   b. UserName (user name of server)
   c. password (password of server)
   Syntax: Connection con = DriverManager.getConnection(url,user,password);
4. Creating Statement object.

Statement can be of 3 types.

a. Statement (static queries)

b. PreparedStatement(dynamic queries)

c. CallableStatement(stored procedures)

It is created from connection object created in last step.

```
Statement st = con.createStatement( );
PreparedStatement ps = con.prepareStatement(query);
CallableStatement cs = con.prepareCall(query);
```

5. Executing query on statement object.

executeUpdate( ) => Used for Update,Delete, Insert

ResultSet rs = executeQuery( ) = > Used for select query

6. Processing result set (required only for 'select' query)

```
while(rs.next( )) {
    String n =rs.getString('name');// getting data of 'name' column
    float m = rs.getFloat(3);//getting data of 3rd column
}
```

7. Closing connection

```
st.close( );//closing statement
con.close( );//closing connection
```

Singleton Design pattern:

- Only one copy of object should be created
- Constructor should be private
- We need to define separate method which returns an object

(e.g. getInstance( ))

```
class A {
        static A obj = null;
        private A( ) { }
        synchronized static A getInstance( ) {
            if(obj == null) {
                obj = new A( );
            }
            return obj;
        }
    }
```

Getting copy of A's object:

```
A obj1 = A.getInstance();
A obj2 = A.getInstance();
```

https://www.eclipse.org/windowbuilder/download.php (swing plugin)

Metadata: (Data about data- Extra information about data)

- ResultSetMetaData
- DatabaseMetaData

JSP tags:

1. Scriptlet
   <% //java code %>
2. Expression tag
   <%= %>
3. Declaration tag
   <%! %>

This gets executed only once. We can define functions inside this.

JSTL - JSP standard Tag library (.jar)
http://localhost:8080/jstl/saveStudent.jsp?id=10&sName=agc&sMarks=99

JSP Actions:
<jsp:include>
<jsp:forward>
<jsp:param>
<jsp:useBean>
<jsp:getProperty>
<jsp:setProperty>
<jsp:useBean class="com.pga.Student" var="s"/>
http://localhost:8080/jspActions/oddEven.jsp?num=10
http://localhost:8080/jspActions/odd.jsp?NUM=101

jsp:include	To include a resource at runtime, can be HTML, JSP or any other file
jsp:useBean	To get the java bean object from given scope or to create a new object of java bean.
jsp:getProperty	To get the property of a java bean, used with jsp:useBean action.
jsp:setProperty	To set the property of a java bean object, used with jsp:useBean action.
jsp:forward	To forward the request to another resource.

<jsp:forward> v/s response.sendRedirect( )

Used to forward request to internal resource of same website	Used to send request to outside resources (e.g. google site)
Browser is NOT involved.	Browser is involved in redirection.
No. of request is less.	One extra request is sent
Faster	Slow due to extra request
Actual URL is not shown (hence secure)	Actual URL is seen.

Servlet:

- Server side program.
- It is a Java class which extends “HttpServlet” class
- Servlet must be mapped with an URL
- There are 2 ways of defining/mapping URL for servlet

1. Using Deployment descriptor file (web.xml)

```
<servlet>
    <servlet-name>first</servlet-name>
    <servlet-class>com.pga.GetMarksServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>first</servlet-name>
```

```
    <url-pattern>/getMarks</url-pattern>
  </servlet-mapping>
```

2. Using @WebServlet annotation

```
@WebServlet(url="/getData")
```

public class GetDataServlet extends HttpServlet { }

- It can consist of "doGet()" and/or "doPost()" method

Session tracking:

- HTTP is stateless protocol, means each request is considered as new request and there is no any track of states of previous requests.
- Many times we need to keep track of previous request.
- e.g. On shopping website we visit various pages on website and at the end we do billing, so there would be lot of HTTP requests made before we reach to billing page. Now billing page need to know all items you added to cart from various web pages. In such scenarios we need to track session.
- Session tracking is also needed to identify users on website.
- So whenever client connects to server, server creates unique session for that client.

Session timeout setting: (web.xml)

```
  <session-config>
    <session-timeout>5</session-timeout>
  </session-config>
```

Programmatically we can destroy the session using

“session.invalidate()” method.

- Options of session tracking:
- 1. Hidden form fields
- 2. URL re-writing
- 3. Cookies
- 1. Using hidden field in <form> tag
- <form>
- ….
- <input type=‘hidden’ value=‘12345’ name=’sessionID’/>
- </form>
- limitation: This field can be used only for <form> tag and not with <a> tag.
- 2. Using URL re-writing
- Append session ID to each and every request
- 3. Using Cookie

user1 -> 1234Xyz
user2 -> 6799abc
user3 -> 9911pqr

Request

/*

Filter1 -> Filter2 -> Request

sop(in filter1 request)

doFilter()
sop(in filter 1 response)

Request processed

sop(in filter2 request)
doFilter()
sop(in filter 3 response

in filter1 request -> in filter2 request -> request processed ->Filter 2 response -> f1 response

- Cookie:
- Cookie is small piece of information sent by server and stored in client's browser.
- It is browser and site specific.
- So cookie for gmail.com under Firefox browser can not be seen under chrome browser.
- It can be used for session tracking.
- Cookie data is always stored as "string"

- Who creates cookie? - Server
- Where cookie are stored? On client side(inside browser)
- Cookies are created and sent in request/response? - response
- What happens when browser receives response with cookies?
- browser reads those cookies and store them inside it's storage.
- What happens with cookies when request is sent from browser?
- browser sends all the active(valid) cookies for that

site in subsequent requests to server.

- What max amount of data can be stored in cookie?
- 4 kb max (per site)

- 1kb - 1024 bytes (512 characters)
- Cookie lifetime is set by server (It can be in seconds, hours, years etc)
- Typical use of Cookie : Session Tracking