Unit III Notes

Scripting: Java script: Introduction, documents, forms, statements, functions, objects; introduction to AJAX,

Networking: Internet Addressing, InetAddress, Factory Methods, Instance Methods, TCP/IP Client Sockets, URL, URL Connection, TCP/IP Server Sockets, Datagram.

What is JavaScript?

JavaScript is a scripting language from NetScape based on Java used for dynamic web development at the client side and server side.

At the client side it is interpreted by web browsers and at the server side it is interpreted by special server side technologies like NodeJS.

It provides manipulation power to HTML using Document Object Model and Events.

Using **<script></script>** tag to merge the JavaScript code into HTML

Document Object Model (DOM) provides a communication system among the tags and other environmental objects like current web browser, current desktop, current web page etc.

DOM provides some predefined names also maintains a collection to elements inside a web page. Some of pre-defined names are

- screen
- navigator
- window
- document
- this

Some predefined collections are

- images
- forms
- frames

We can also define identifier to the tags to communicate with them use **ID** and **NAME** attributes.

These objects provide different methods to work with them.

Properties of **screen** object

- width

- height
- availWidth
- availHeight
- colorDepth

Properties of **navigator** object

- appName
- appCode
- appVersion

Methods of **window** object

- print()
- open()
- close()

Properties and methods of **document** object

- title
- getElementById()
- getElementByName()
- write()

Data types in JavaScript

JavaScript is a dynamic type language where we can store any type of data in same variable at different places in the program.

Use **var** keyword to such variables

Example

var num=45;

num="Hello";

num=true;

Dialogs in JavaScript

JavaScript provides three dialogs for user interaction

- alert()
    - To show a message dialog
- prompt()
    - To input some data from user

- confirm()
    - To take some confirmation from user
    - It displays Ok and Cancel Buttons
    - If we click on Ok, treats as true and Cancel button is treated as false

Syntaxes

```
alert(message);

variable=prompt(message);

if(confirm(message)){

	// statements for Ok

} else{

	// statements for Cancel

}
```

Example

```
<script>

	var name=prompt("Enter your name");

	alert("Welcome "+name);

</script>
```

Defining functions in JavaScript

Use **function** keyword to define the functions in JavaScript

Use **return** keyword to return some value. No return type need to define to return a value.

Syntax 1

```
function <functionname>(<arguments>){

	//statements

}
```

Syntax 2

```
function <functionname>(<arguments>){

        //statements

        return <value or variable>;

}
```

Working with Events

The events are moments on which some action is taken.

- load
  - When a form get loaded
- click
  - When we click on a control like button
- mouseover
  - When we move mouse over some control
- mouseout
  - When we move mouse outside the control
- keypress
  - When we press a key from keyboard
- submit
  - When we submit a form

These moments are defined as attribute to be used on different tags by prefixing **on** with them

- onload
- onclick
- onmousemouse
- onmouseout
- onkeypress
- onsubmit

Types of JavaScript

JavaScript code can be written using three ways

- Immediate Script
- Deferred Script
- Hybrid Script

In case of immediate script, the code get executed immediately once we run a web page

Example

```
<script>
    alert("Welcome to JavaScript");
</script>
```

In case of deferred script, we create a function and that function get called based on some specific event. When that event occurs only then the code get executed.

Example

```
<script>
    function demo(){
    alert("Welcome to JavaScript");
}
</script>
<input type="button" value="Click Me" onClick="demo()">
```

In case of hybrid script, some code get executed immediately while other code get executed later based on an event.

```
<script>
    alert("Welcome");
    function hello(){
        alert("Hello to JavaScript");
}
</script>
<input type="button" value="Say Hello" onClick="hello()">
```

Data Conversion functions

When we input some data using HTML tags, it gives textual data or string data. If we have to do some manipulation on that data we need to convert the data into numeric type.

To convert the data to numeric types we need conversion functions

- parseInt()
- parseFloat()

The **parseInt()** function is used to convert string data to integer type data

The **parseFloat()** function is used to convert string type data to float type data

Example

Create a web page having a text box to input a number and a button to show factorial of that number in a dialog.

Solution

```
<script>
    function factorial(){
        var num=document.getElementById("num");
        var n=parseInt(num.value);
        var f=1;
        for(i=1;i<=n;i++) f=f*i;
        alert("Factorial is "+f);
    }
</script>
<!DOCTYPE html>
<html>
<head></head>
<body>
Number <input type="text" id="num"> <input type="button" value="Factorial"
onclick="factorial()">
</body>
</html>
```

What are JavaScript Objects?

The JavaScript objects are special kind of variables that hold some reference rather than value. An object can have a set of values called properties and some functionality called as functions. E.g.

```
var book={bookno:101,title:"Java Programming",price:780.50};
```

Here book in an object that contains three properties name as bookno, title and price.

Use **key:value** pair model to define the property and use the dot (.) operator to access the properties

Example

```
<script>
    var book={bookno:101,title:"Java Programming",price:780.50};
    alert("Book Title is "+book.title);
</script>
```

Use **const** keyword if you do not want to allow to make any changes in the object properties

```
<script>
    const book={bookno:101,title:"Java Programming",price:780.50};
    alert("Book Title is "+book.title);
</script>
```

An object can have functions as well that are stored as property. Use built-in object **this** to refer the current instance and access the properties of current object.

```
const person = {
    firstName: "John",
    lastName : "Doe",
    id       : 5566,
    fullName : function() {
      return this.firstName + " " + this.lastName;
    }
  };
```

Use **object.methodname()** to call the method

Full Example

```
First Name <input type="text" id="firstName"><br>
Last Name <input type="text" id="lastName"><br>
Full Name <input type="text" id="fullName">
<script>
const person = {
    firstName: "John",
    lastName : "Doe",
    id       : 5566,
    fullName : function() {
      return this.firstName + " " + this.lastName;
    }
  };
  document.getElementById("firstName").value=person.firstName;
  document.getElementById("lastName").value=person.lastName;
  document.getElementById("fullName").value=person.fullName();
</script>
```

Introduction to AJAX

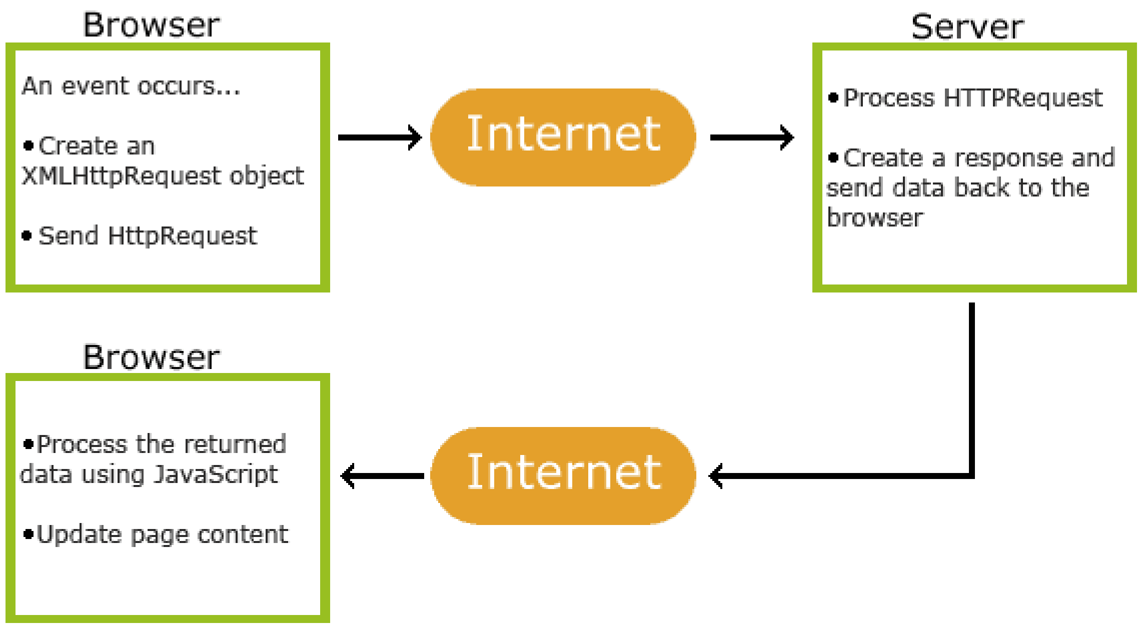
AJAX stands for Asynchronous JavaScript and XML

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

AJAX just uses a combination of two things

- A browser built-in **XMLHttpRequest** object
  - To request data from a web server
- JavaScript and HTML DOM
  - To display or use the data

How AJAX Works?


Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

An event occurs in a web page (the page is loaded, a button is clicked)

2. An XMLHttpRequest object is created by JavaScript

3. The XMLHttpRequest object sends a request to a web server

4. The server processes the request

5. The server sends a response back to the web page

6. The response is read by JavaScript

7. Proper action (like page update) is performed by JavaScript

Using XMLHttpRequest object

Create an XMLHttpRequest object

```
var xhttp = new XMLHttpRequest();
```

Define a callback function

A callback function is a function passed as a parameter to another function.

```
xhttp.onreadystatechange = function() {
  // What to do when the response is ready
}
```

Open the XMLHttpRequest object

```
xhttp.open("GET", "filename");
```

Send a Request to a server

```
xhttp.send();
```

How to check the current status of the request?

Use **readyState** property to check the current status of the request

0: request not initialized

1: server connection established

2: request received

3: processing request

4: request finished and response is ready

The **onreadystatechange** event is used to define a function to perform some action when a ready state changes

The **status** property is used to check the request status

200: "OK"
403: "Forbidden"
404: "Not Found"

Use **responseText** property to get the response

Example

Create a web page to read the contents of a file books.txt and show the contents on the page when we click on a button.

Records in books.txt file

101, Java Programming, 405
102, C Programming, 304

Contents of ajaxdemo.html file

```
<!DOCTYPE html>
<html>
<body>
<h2>AJAX DEMO</h2>
<div id="demo">
</div>
<button type="button" onclick="loadDoc()">Show Books</button>
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "books.txt");
  xhttp.send();
}
</script>
</body>
</html>
```

What is Instance method?

The methods that always need an instance to call them are called as instance methods.

Example

Create a class **Number** having a data member as **num**. Create a constructor to initialize data in num. Create an instance method factorial() to return factorial of num.

```
public class Number {
    private int num;
    public Number(int num){
        this.num=num;
    }
    public int factorial(){ //instance method
        int f=1;
        for(int i=1;i<=num;i++)
            f=f*i;
        return f;
    }
}
class InstanceMethodDemo{
    public static void main(String[] args) {
        Number x=new Number(7);
        System.out.println(x.factorial());
    }
}
```

What is Class method?

The methods that do not require an instance are called as class methods. Such methods are called using the class name but can also be called using instance.

Use **static** keyword to define such methods.

Example

Create a class **Number** having a class method factorial() that takes a number as argument and return factorial of it.

```
public class Number {
    public static int factorial(int num){ //class method
        int f=1;
        for(int i=1;i<=num;i++)
            f=f*i;
        return f;
    }
}
class InstanceMethodDemo{
    public static void main(String[] args)  {
        System.out.println(Number.factorial(6));
        Number x=new Number();
        System.out.println(x.factorial(8));
    }
}
```

What is factory method?

Special class method that returns instance of its own class and allows to create only one instance of that class is called as factory method.

To define such methods, we need to mark the constructor as private.

```
public class Number {
    private int num;
    private static Number x=new Number(); // singleton object
    private Number(){ // private constructor
    }
    public static Number createNumber(int num){ // factory
method
        x.num=num;
        return x;
    }
    public int factorial(){ //instance method
        int f=1;
        for(int i=1;i<=num;i++)
            f=f*i;
        return f;
    }
}
class InstanceMethodDemo{
    public static void main(String[] args)  {
        Number x=Number.createNumber(7);
        System.out.println(x.factorial());
    }
}
```

What is singleton object?

When a class allows to create to only one instance or object then that object is known as singleton object.

Can we have private constructor? If yes, what is the use of private constructor?

Yes, Used to create singleton object and factory methods

What is in, out and err?

The in is static reference variable of InputStream class of java.io package

The out and err are reference variables of PrintStream class of java.io package

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Scanner;
public class Sys {
    public static Scanner sc=new
Scanner(System.in);
    public static BufferedReader br=new
BufferedReader(new
InputStreamReader(System.in));
}
```

```
import java.io.IOException;
public class Voter {
    public static void main(String[]
args) throws IOException {
        System.out.println("Enter your
name :");
        String name=Sys.br.readLine();
        System.out.println("Age :  ");
        int age=Sys.sc.nextInt();
        if(age>=18)
            System.out.println("Dear
"+name+" you can vote");
        else
            System.out.println("Dear
"+name+" you cannot vote");
    }
}
```

Java Networking or Socket Programming

A system to provide the communication between client and server applications. Such applications can be on same computer or other computers.

The Java Networking can work in two modes

- Connected mode
- Disconnected mode

In connected mode, the client and server both must be connected together. If server is not connected then data will not be transferred at all.

In disconnected mode, the client and server and not connected directly. The client sends the request to the server, if the server is available it will handle the request but no confirmation is returned. If server is not available, then data will be lost.

To transfer the data in connected mode Java uses TCP/IP protocol while in disconnected mode Java uses UDP (User Datagram Protocol).

All required classes for networking are provided in **java.net** package

- Socket class
- ServerSocket class
- DatagramPacket class
- DatagramSocket class
- InetAddress class

To send the data to some other machine we must have two things

- IP Address or DNS (Domain Name Server) of other machine
- Port Number

One machine can have 65535 ports. Some of the ports are used by default by some applications

- Port 80 is used for HTTP
- Port 8080 is used by Tomcat and Oracle servers
- Port 21 is used for FTP (File Transfer Protocol)
- Port 25 is used for SMTP (Simple Mail Transfer Protocol)
- Port 3306 is taken by MySQL by default
- Port 1521 is taken by Oracle Database by default

Working with Socket class

- A class used to lookup a server and connect with it
- To lookup a server we need two things
    - IP Address or DNS
    - Port Number

Syntax

Socket(String ipaddress or dsn, int port number) throws IOException

If connection is not established then IOException is thrown

Once the connection is established, now we can transfer the data to other machine and can also read the data from other machine.

Use getInputStream() method to open the keyboard of remote machine.

InputStream getInputStream()

Use getOutputStream() method to open the monitor of remove machine

OutputStream getOutputStream()

To place the server on a port inside a machine, we can check the free port.

Example

Write a program to input the port number from user. Check the availability of that port on local machine.

```
import java.io.IOException;
import java.net.*;
import java.util.Scanner;
public class SocketTest {
    public static void main(String[] args){
        int port=0;
        try{
            Scanner sc=new Scanner(System.in);
            System.out.print("Enter the port number to check :
");
            port=sc.nextInt();
            Socket s=new Socket("localhost",port);
            System.out.println("Port "+port+" is busy");
        }catch (IOException ex){
            System.out.println("Port "+port+" is free");
        }
    }
}
```

Creating a server socket

- A server socket is used to server multiple clients
- A server socket requires the port number
- ServerSocket(int portnumber) throws IOException
- Use **accept()** method to wait for a client and return Socket information of client machine
    - Socket accept()

Client request → Server Socket

Example

Write a program having a server socket that can serve multiple clients. When a client get connected then show client information with a message “Client connected”

```
import java.net.*;
import java.io.*;
public class DemoServer {
    public static void main(String[] args) {
        try{
            ServerSocket server=new ServerSocket(8080);
            System.out.println("Server is ready....");
            for(;;){
                Socket client=server.accept();
                System.out.println("Client connected
"+client);
            }
        }catch (Exception ex){
            System.err.println("Error : "+ex.getMessage());
        }
    }
}
```

```
import java.net.*;
import java.io.*;
public class DemoClient {
    public static void main(String[] args) {
        try{
            Socket server=new Socket("localhost",8080);
            System.out.println("Server found : "+server);
        }catch (IOException ex){
            System.out.println("Error : "+ex.getMessage());
        }
    }
}
```

Example

Write a program having a server socket that can serve multiple clients. When a client get connected then show client information with a message “Client connected”

Send a message from the server to the client “Connected on <current date and time>”

```
import java.net.*;
import java.io.*;
public class DemoServer {
    public static void main(String[] args) {
        try{
            ServerSocket server=new ServerSocket(8080);
            System.out.println("Server is ready....");
            for(;;){
                Socket client=server.accept();
                System.out.println("Client connected
"+client);
                String message="Connected on "+new
java.util.Date();
                OutputStream os=client.getOutputStream();
                PrintWriter pw=new PrintWriter(os,true);
                pw.println(message);
            }
        }catch (Exception ex){
            System.err.println("Error : "+ex.getMessage());
        }
    }
}
```

```
import java.net.*;
import java.io.*;
public class DemoClient {
    public static void main(String[] args) {
        try{
            Socket server=new Socket("localhost",8080);
            System.out.println("Server found : "+server);
            InputStream is=server.getInputStream();
            BufferedReader br=new BufferedReader(new
InputStreamReader(is));
            String message=br.readLine();
            System.out.println(message);
        }catch (IOException ex){
            System.out.println("Error : "+ex.getMessage());
        }
    }
}
```

Using User Datagram Protocol (UDP)

To send the data in disconnected mode we three classes

- DatagramSocket
- DatagramPacket
- InetAddress

DatagramSocket class is used to a socket between client and server using a port number

- DatagramSocket()
- DatagramSocket(int portnumber)

Once the socket is created we can send the data among the client and server. To send the data, every time we need to create a DatagramPacket with the data in binary and to receive the data we must a blank data packet

- DatagramPacket(byte []data, int length)
- DatagramPacket(byte []data, int length, InetAddress ia, int portno)
- byte []getData()
- int getLength()

To convert the data from string to byte array use **getBytes()** method of string class

byte[] getBytes()

To convert the byte data again to string data, use the constructor of String class

String(byte []data, int offset, int length)

The InetAddress class is used to get the address of the machine on which we need to send the data using factory methods

- InetAddress getByName()
- InetAddress.getLocalHost()

```java
import java.net.*;
public class Sender{
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();
        String str = "Welcome to UDP System";
        InetAddress ip = InetAddress.getByName("localhost");
        DatagramPacket dp = new DatagramPacket(str.getBytes(),
str.length(), ip, 3000);
        ds.send(dp);
        ds.close();
    }
}
```

```java
import java.net.*;
public class Receiver{
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(3000);
        byte[] buf = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf, 1024);
        ds.receive(dp);
        String str = new String(dp.getData(), 0,
dp.getLength());
        System.out.println(str);
        ds.close();
    }
}
```

What is URL?

URL stands for Uniform Resource Locator. It is actual path to reach to some resource on the Internet.

URL has two parts

- Protocol to access the resource
- The location of the resource

Examples of URL

http://www.yahoo.com

https://www.yahoo.com

ftp://www.yahoo.com

Java provides URL class under java.net package to work with some URL for reading and writing the data on that URL

It is used to establish the connection with the given URL and return the reference of another class URLConnection for further processing.

Constructor

URL(String url)

Methods

URLConnection openConnection()

What is URLConnection?

The Java URLConnection class represents a communication link between the URL and the application. It can be used to read and write data to the specified resource referred by the URL.

Once a connection is established and the Java program has an URLConnection object, we can use it to read or write or get further information like content length, etc.

Constructor

URLConnection(String url)

Methods

Object getContent()

int getContentLength()

String getContentType()

InputStream getInputStream()

```
import java.io.*;
import java.net.*;
public class URLConnectionDemo {
    public static void main(String[] args){
        try{
            URL url=new URL("https://abes.ac.in");
            URLConnection urlcon=url.openConnection();
            InputStream stream=urlcon.getInputStream();
            int i;
            while((i=stream.read())!=-1){
                System.out.print((char)i);
            }
        }catch(Exception e){System.out.println(e);}
    }
}
```