Web Technologies
UNIT-III

AJAX (Asynchronous JavaScript and XML)

AJAX is an acronym for **Asynchronous JavaScript and XML**. It is a group of inter-related technologies like javascript, dom, xml, html, css etc. AJAX allows you to send and receive data asynchronously without reloading the entire web page. So it is fast.

AJAX allows you to send only important information to the server not the entire page. So only valuable data from the client side is routed to the server side. It makes your application interactive and faster.

Where it is used?

There are too many web applications running on the web that are using AJAX Technology. Some are:

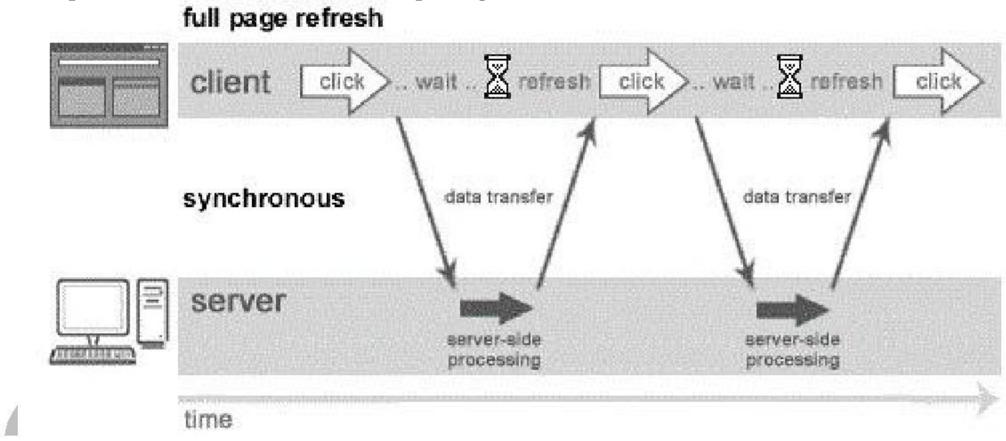
1. Gmail
2. Facebook
3. Twitter
4. Google maps
5. YouTube etc.,

Synchronous Vs. Asynchronous Application

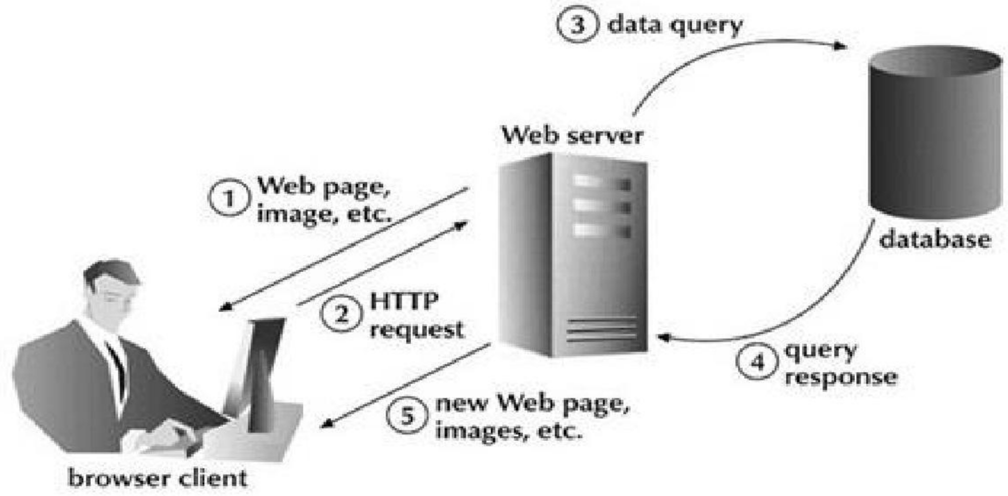
Before understanding AJAX, let's understand classic web application model and AJAX Web application model.

❖ **Synchronous (Classic Web-Application Model)**

A synchronous request blocks the client until operation completes i.e. browser is not unresponsive. In such case, JavaScript Engine of the browser is blocked.

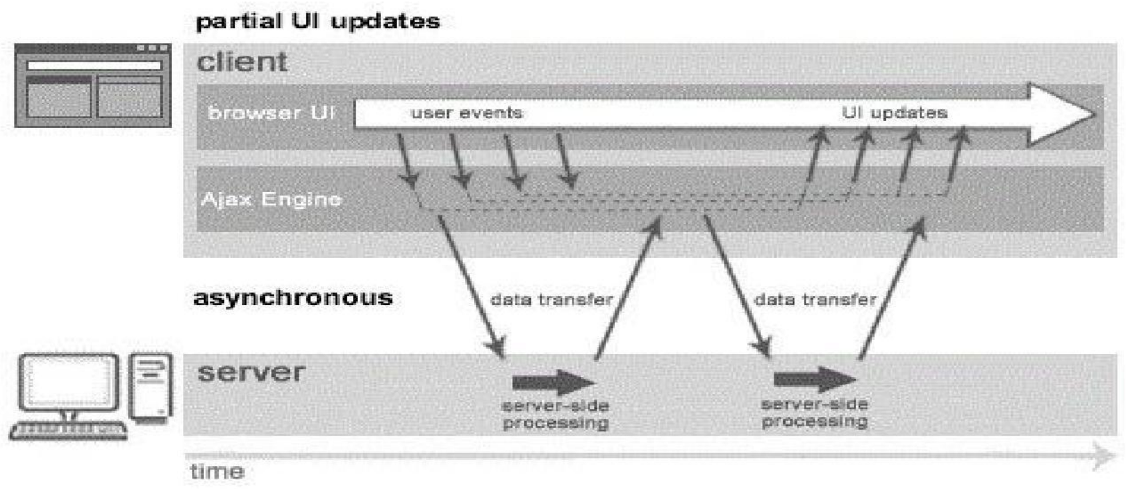


As you can see in the above image, full page is refreshed at request time and user is blocked until request completes. Let's understand it another way.

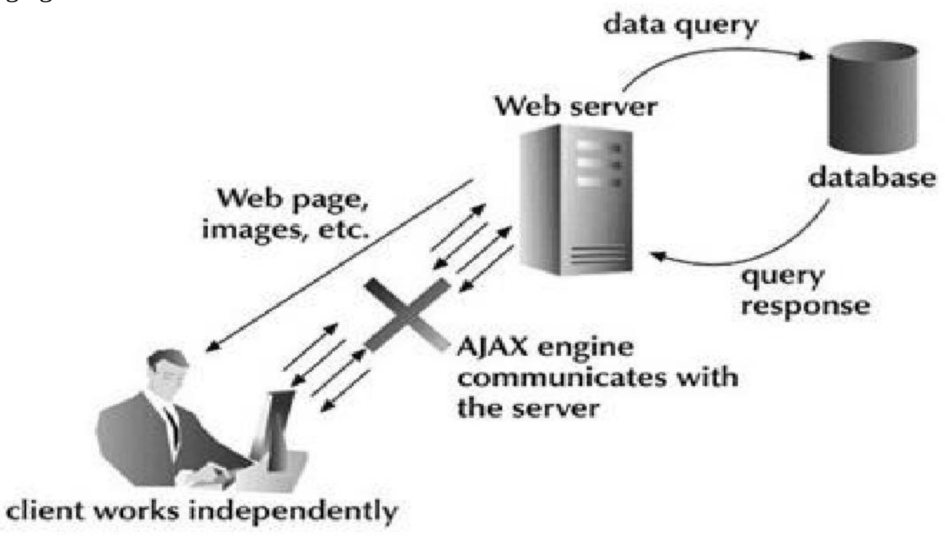


❖ **Asynchronous (AJAX Web-Application Model)**

An asynchronous request doesn't block the client i.e. browser is responsive. At that time, user can perform other operations also. In such case, JavaScript Engine of the browser is not blocked.



As you can see in the above image, full page is not refreshed at request time and user gets response from the AJAX Engine. Let's try to understand asynchronous communication by the image given below.


data query
Web server
database
Web page, images, etc.
query response
AJAX engine communicates with the server
client works independently

AJAX Technologies

AJAX is not a Technology but group of inter-related technologies. AJAX Technologies includes:

- HTML/XHTML and CSS
- DOM
- XML or JSON(JavaScript Object Notation)
- XMLHttpRequest
- JavaScript

- **HTML/XHTML and CSS**
  These technologies are used for displaying content and style. It is mainly used for presentation.

- **DOM**
  It is used for dynamic display and interaction with data.

- **XML or JSON**
  For carrying data to and from server. JSON is like XML but short and faster than XML.

- XMLHttpRequest
  For asynchronous communication between client and server.

- JavaScript
  It is used to bring above technologies together. Independently, it is used mainly for client-side validation.

Understanding XMLHttpRequest

An object of XMLHttpRequest is used for asynchronous communication between client and server. It performs following operations:

1. Sends data from the client in the background
2. Receives the data from the server
3. Updates the webpage without reloading it.

- **Properties of XMLHttpRequest object:**

Property	Description
onReadyStateChange	It is called whenever readystate attribute changes. It must not be used with synchronous requests.
readyState	Represents the state of the request. It ranges from 0 to 4. **0** UNOPENED open() is not called. **1** OPENED open is called but send() is not called. **2** HEADERS_RECEIVED send() is called, and headers and status are available. **3** LOADING Downloading data; responseText holds the data. **4** DONE The operation is completed fully.
reponseText	Returns response as TEXT.
responseXML	Returns response as XML

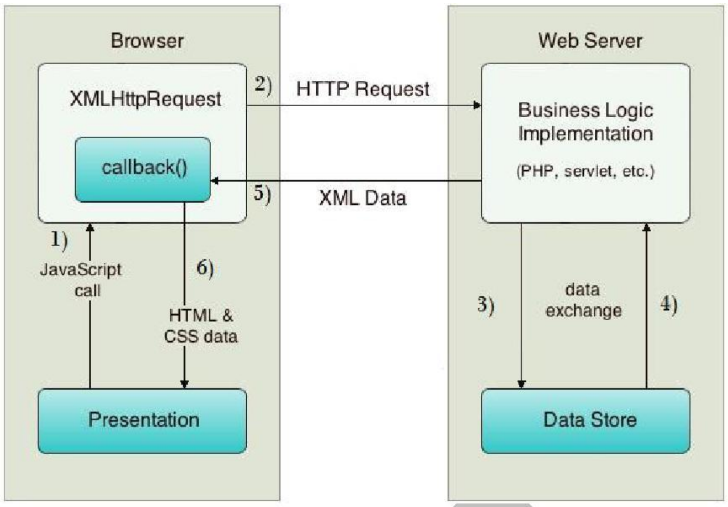
- **Methods of XMLHttpRequest object**

Method	Description
void open(method, URL)	Opens the request specifying get or post method and url.
void open(method, URL, async)	Same as above but specifies asynchronous or not.
void open(method, URL, async, username, password)	Same as above but specifies username and password.
void send()	Sends GET request.
void send(string)	Sends POST request.
setRequestHeader(header,value)	It adds request headers.

How AJAX Works?

AJAX communicates with the server using XMLHttpRequest object. Let's understand the flow of AJAX with the following figure:

1. User sends a request from the UI and a javascript call goes to XMLHttpRequest object.
2. HTTP Request is sent to the server by XMLHttpRequest object.
3. Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.
4. Data is retrieved.
5. Server sends XML data or JSON data to the XMLHttpRequest callback function.
6. HTML and CSS data is displayed on the browser.



Integrating PHP and AJAX:-

The following example will demonstrate how a web page can communicate with a web server while a user type characters in an input field:

Example

Start typing a name in the input field below:

First name:

Suggestions:

Example Explained

In the example above, when a user types a character in the input field, a function called "showHint()" is executed.

The function is triggered by the onkeyup event.

Here is the HTML code:

Example

```
<html>
<head>
<script>
function showHint(str) {
   if (str.length == 0) {
      document.getElementById("txtHint").innerHTML = "";
      return;
   } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
           if (this.readyState == 4 && this.status == 200) {
              document.getElementById("txtHint").innerHTML = this.responseText;
           }
        };
        xmlhttp.open("GET", "gethint.php?q=" + str, true);
        xmlhttp.send();
    }
}
</script>
</head>
<body>

<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

Code explanation:

First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.

However, if the input field is not empty, do the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a PHP file (gethint.php) on the server
- Notice that q parameter is added to the url (gethint.php?q="+str)
- And the str variable holds the content of the input field

The PHP File - "gethint.php"

The PHP file checks an array of names, and returns the corresponding name(s) to the browser:

```
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
$a[] = "Johanna";
```

```
$a[] = "Kitty";
$a[] = "Linda";
$a[] = "Nina";
$a[] = "Ophelia";
$a[] = "Petunia";
$a[] = "Amanda";
$a[] = "Raquel";
$a[] = "Cindy";
$a[] = "Doris";
$a[] = "Eve";
$a[] = "Evita";
$a[] = "Sunniva";
$a[] = "Tove";
$a[] = "Unni";
$a[] = "Violet";
$a[] = "Liza";
$a[] = "Elizabeth";
$a[] = "Ellen";
$a[] = "Wenche";
$a[] = "Vicky";

// get the q parameter from URL
$q = $_REQUEST["q"];

$hint = "";

// lookup all hints from array if $q is different from ""
if ($q !== "") {
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name) {
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint === "") {
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
        }
    }
}

// Output "no suggestion" if no hint was found or output correct values
echo $hint === "" ? "no suggestion" : $hint;
?>
```

Introduction to Web Services

Technology keep on changing, users were forces to learn new application on continuous basis. With internet, focus is shifting to-wards services based software. Users may access these services using wide range of devices such as PDAs, mobile phones, desktop computers etc. Service oriented software development is possible using man known techniques such as COM, CORBA, RMI, JINI, RPC etc. some of them are capable of delivering services over web & some or not. Most of these technologies uses particular protocols for communication & with no standardization. **Web service** is the concept of creating services that can be accessed over web. Most of these

What are Web Services?

A web services may be defines as: An application component accessible via standard web protocols. It is like unit of application logic. It provides services & data to remote clients & other applications. Remote clients & application access web services with internet protocols. They use XML for data transport & SOAP for using services. Accessing service is independent of implementation. With component development model, web service must have following characteristics:

- ❖ Registration with lookup service
- ❖ Public interface for client to invoke service
- ❖ It should use standard web protocols for communication
- ❖ It should be accessible over web
- ❖ It should support loose coupling between uncoupled distributed systems

Web services receive information from clients as messages, containing instructions about what client wants, similar to method calls with parameters. These message delivered by web services are encoded using XML.XML enabled web services are interoperable with other web services.

Web Service Technologies:

Wide variety of technologies supports web services. Following technologies are available for creation of web services. These are vendor neutral technologies. They are:

- ❖ Simple Object Access Protocol(SOAP)
- ❖ Web Services Description Language(WSDL)
- ❖ UDDI(Universal Description Discovery and Integration)

Simple Object Access Protocol (SOAP):

SOAP is a light weight & simple XML based protocol. It enables exchange of structured & typed information on web by describing messaging format for machine to machine communication. It also enables creation of web services based on open infrastructure. SOAP consists of three parts:

- ❖ **SOAP Envelope**: defines what is in message, who is the recipient, whether message is optional or mandatory

- **SOAP Encoding Rules**: defines set of rules for exchanging instances of application defined data types
- **SOAP RPC Representation**: defines convention for representing remote procedure calls & response

SOAP can be used in combination with variety of existing internet protocols & formats including HTTP, SMTP etc. Typical SOAP message is shown below:

```
<IVORY:Envelope xmlns:IVORY=”http://schemas.xmlsoap.org/soap/envelope”
                IVORY:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding”>
<IVORY:Body>
        <m:GetLastTradePrice xmlns:m=”Some-URI”>
        <symbol>DIS</symbol>
        </m:GetLastTradePrice>
</IVORY:Body>
</IVORY:Envelope>
```

The consumer of web service creates SOAP message as above, embeds it in HTTP POST request & sends it to web service for processing:

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset=”utf-8”
Content-Length: nnnn
SOAPAction: “Some-URI”
        ....
SOAP Message
....
```

The message now contains requested stock price. A typical returned SOAP message may look like following:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope”
SOAP-ENV:encodingStyle=” http://schemas.xmlsoap.org/soap/encoding” />
        <SOAP-ENV:Body>
                <m:GetLastTradePrice xmlns:m=”Some-URI”>
                        <Price>34.5</Price>
                </m:GetLastTradePrice>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Interoperability:

The major goal in design of SOAP was to allow for easy creation of interoperable distributed web services. Few details of SOAP specifications are open for interpretation; implementation may differ across different vendors. SOAP message though it is conformant XML message, may not strictly follow SOAP specification.

Implementations:

SOAP technology was developed by DevelopMentor, IBM, Lotus, Microsoft etc. More than 50 vendors have currently implemented SOAP. Most popular implementations are by Apache which is open source java based implementation & by Microsoft in .NET platform. SOAP specification has been submitted to W3C, which is now working on new specifications called XMLP (XML Protocol)

SOAP Messages with Attachments (SwA)

SOAP can send message with an attachment containing of another document or image etc. On Internet, GIF, JPEG data formats are treated as standards for image transmission. Second iteration of SOAP specification allowed for attachments to be combined with SOAP message by using multipart

MIME structure. This multi part structure is called as **SOAP Message Package**. This new specification was developed by HP & Microsoft. Sample SOAP message attachment is shown here:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary;
type=text/xml; start=”<myimagedoc.xml@mystie.com>”
Content-Description: This is the optional message description.
--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <myimagedoc.xml@mysite.com>
<?xmll version= ”1.0 ”?>
<SOAP-ENV: Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope”
<SOAP-ENV:Body>
        …
        <theSignedForm href=”cid:myimage.tiff@mysite.com” />
        …
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
        --MIME_boundary
        Content-Type: image/tiff
        Content-Transfer-Encoding: binary
        Content-ID: <myimagedoc.xml@mysite.com>
        …binary TIFF image…
        --MIME_boundary--
```

Web Services Description Language (WSDL)

WSDL is an XML format for describing web service interface. WSDL file defines set of operations permitted on the server & format that client must follow while requesting service. WSDL file acts like contract between client & service for effective communication between two parties. Client has to request service by sending well formed & conformant SOAP request.

If we are creating web service that offered latest stock quotes, we need to create WSDL file on server that describes service. Client obtains copy of this file, understand contract, create SOAP request based on contract & dispatch request to server using HTTP post. Server validates the request, if found valid executes request. The result which is latest stock price for requested symbol is then returned to client as SOAP response.

WSDL Document:

WSDL document is an XML document that contains of set of definitions. First we declare name spaces required by schema definition:

```
<schema xmlns=”http://www.w3.org/2000/10/XMLSchema”
xmlns:wsdl=”http://schemas.xmlsoap.org/wsdl/”
targetNameSpace=http://schemas.xmlsoap.org/wsdl/ elementFormDefault=”qualified”>
```

The root element is definitions as shown below:

```
<wsdl:defiinitions name=”nmtoken”? targetNameSpace=”uri”?>
        <import namespace=”uri” location=”uri”/>
<wsdl:documentation ….. />?
        …
</wsdl:definitions>
```

The *name* attribute is optional & can serve as light weight form of documentation. The *nmtoken* represents name token that are qualified strings similar to CDATA, but character usage is limited to letters, digits, underscores, colons, periods & dashes. A *targetNamespace* may be specified by providing uri. The *import* tag may be used to associate namespace with document locations. Following code segment shows how declared namespace is associated with document location specified in *import* statement:

```
<definitions name=”StockQuote”
        targetNameSpace=”http://example.com/stockquote/defiinitions”
        xmlns:tns="http://example.com/stockquote/definitions"
        xmlns:xsdl="http://example.com/stockquote/schemas"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
<import namespace="http://example.com/stockquote/schemas"
        Location="http://example.com/stockquote/stockquote.xsd"/>
```

Finally, optional *wsdl:documentation* element is used for declaring human readable documentation. The element may contain any arbitrary text. There are six major elements in document structure that describes service. These are as follows:

❖ **Types Element:** it provides definitions for data types used to describe how messages will exchange data. Syntax for types element is as follows:

```
<wsdl:types> ?
        <wsdl:documentation
        .../> <xsd:schema .../>
        <-- extensibility element -->
</wsdl:types>
```

The *wsdl:documentation* tag is optional as in case of *definitions*. The *xsd* type system may be used to define types in message. WSDL allows type systems to be added via extensibility element.

❖ **Message Element:** It represents abstract definition of data begin transmitted. Syntax for message element:

```
<wsdl:message name="nktoken"> *
        <wsdl;documentation .../>
            <part name="nmtoken" element="qname"? type="qname"? /> *
</wsdl:message>
```

The *message name* attribute is used for defining unique name for message with in document scope. The *wsdl:documentation* is optional & may be used for declaring human readable documentation. The message consists of one or more logical parts. The *part* describes logical abstract content of message. Each part consists of name & optional element & type attributes.\

❖ **Port Type Element:** It defines set of abstract operations. An operation consists of both input & output messages. The *operation* tag defines name of operation, *input* defines input for operation & *output* defines output format for result. The *fault* element is used for describing contents of SOAP fault details element. It specifies abstract message format for error messages that may be output as result of operation:

```
<wsdl:portType name="nmtoken">*
                <wsdl:documentation ..../>?
        <wsdl:operation name="nmtoken">*
                <wsdl:documentation ..../>?
                        <wsdl:input name="nmtoken"? message="qname">?
                <wsdl:documentation ..../>?
                </wsdl:input>
                        <wsdl:output name="nmtoken"? message="qname">?
                <wsdl:documentation ..../>?
                </wsdl:output>
                        <wsdl:fault name="nmtoken"? message="qname">?
                        <wsdl:documentation ..../>?
                </wsdl:fault>
        </wsdl:operation>
</wsdl:portType>
```

❖ **Binding Element:** It defines protocol to be used & specifies data format for operations & messages defined by particular *portType*. The full syntax for binding is given below:

```
<wsdl:binding name=”nmtoken” type=”qname”> *
                <wsdl:documentation ..../>?
                <--Extensibility element -->*
        <wsdl:operation name=”nmtoken”>*
                <wsdl:documentation ..../>?
                <--Extensibility element -->*
        <wsdl:input> ?
                <wsdl:documentation ..../>?
                <--Extensibility element -->*
        </wsdl:input>
        <wsdl:output> ?
                <wsdl:documentation ..../>?
                <--Extensibility element -->*
        </wsdl:output>
        <wsdl:fault name=”nmtoken”> *
                <wsdl:documentation ..../>?
                <--Extensibility element -->*
        </wsdl:fault>
        </wsdl:operation>
</wsdl:binding>
```

The operation in WSDL file can be document oriented or remote procedure call (RPC) oriented. The style attribute of *<soap:binding>* element defines type of operation. If operation is document oriented, input & output messages will consist of XML documents. If operation is RPC oriented, input message contains operations input parameters & output message contains result of operation.

❖ **Port Element:** It defines individual end point by specifying single address for binding:

```
<wsdl:port name=”nmtoken” binding=”qname”> *
        <--Extensibility element (1) -->
</wsdl:port>
```

The *name* attribute defines unique name for port with current WSDL document. The *binding* attribute refers to binding & extensibility element is used to specify address information for port.

❖ **Service Element:** it aggregates set of related ports. Each port specifies address for binding:

```
<wsdl:service name=”nmtoken”> *
                <wsdl:documentation ..../>?
        <wsdl:port name=”nktoken” binding=”qname”> *
                <wsdl:documentation .../> ?
                <--Extensibility element -->
        </wsdl:port>
                <--Extensibility element -->
        </wsdl:service>
```

Universal Description, Discovery & Integration (UDDI)

We need to publish web services so that customers & business partners can use the services. It requires common registry to register web service for clients to find it. For this several vendors including IBM, HP, Oracle, Sun Microsystem etc. formed an industry consortium known as UDDI. Today more than 250 companies have joined UDDI project. The main task of this project is to develop specifications for web based business registry. The registry should be able to describe web service & allow others to discover registered web services.

UDDI allows any organization to publish information about its web services. The framework defines standard for businesses to share information, describe their services & their business & to decide what information is made public & what information is kept private. The interface is based on XML & SOAP, uses HTTP to interact with registry.

Registry itself holds information about business such as company name, contact etc. it holds both descriptive & technical information about web service. It provides search facilities that allow to search specific industry segment or geographic location.

Implementation:

This is global, public registry called UDDI business registry. It is possible for individuals to set up private UDDI registries. The implementations for creating private registries are available from IBM, Idoox etc. Microsoft has developed UDDI SDK that allows visual basic programmer to write program code to interact with UDDI registry. The use of SDK greatly simplifies interaction with registry & shields programmer from local level details of XML & SOAP.

Electronic Business XML (ebXML):

ebXML is set of specifications that allows businesses to collaborate. It enables global electronic market place where business can meet & tranasact with help of XML based messages. Business may be geographically located anywhere in world & could be of any size to participate in global marketplace. The framework defines specifications for sharing of web based business services. It includes specifications for message service, collaborative partner agreements, core components, business process methodology, registry & repository.

It defines registry & repository where business can register themselves by providing their contact information, address & so on. Such information is called Core Component. After business has registered with ebXML registry, other partners can look up registry to locate that business. Once business partner is located, the core components of located business are downloaded. Once buyer is satisfied with fact that seller service can meet its requirements, it negotiates contract with seller. Such collaborative partner agreements are defined in ebXML. Once both parties agree on contract terms, sign agreements & collaborative business transaction by exchanging their private documents. ebXML provides marketplace & defines several XML based documents for business to join & transact in such marketplace.