AJAX:

Web Service:

Set of instructions gets executed at the server side which are capable of considering a request from client side, process the request accordingly and response back is called a web service.

* In a single server, it can run any no. of web services
* Every web service within the server gets identified with an unique URL
* all the web services within the server runs independent to each other.

Ex: Sample webservice Urls:

http://sample.com/data/info

http://abc.in/test/data/student/data

https://services.com/data etc.

AJAX:

The process of creating a request from client side to a server through a web service URL in order to indirectly process operations on database is called AJAX.

Synchronous & Asynchronous Communication

While interacting to the server from the client, at a time we could able to create any no. of requests, while communicating to the server, the communication type could be synchronous (or) asynchronous comm.

Synchronous Communication:

In this type, once the request been send to the server, from client side it doesn't execute further instructions until there is a response from the server.

Asynchronous Communication:

In this type, once the requests been send to the server, client doesn't waits for its response it continues executing further instructions. Call back function holds the set of instructions and executes automatically once there is a response from the server

Specifying the data type while Communicating

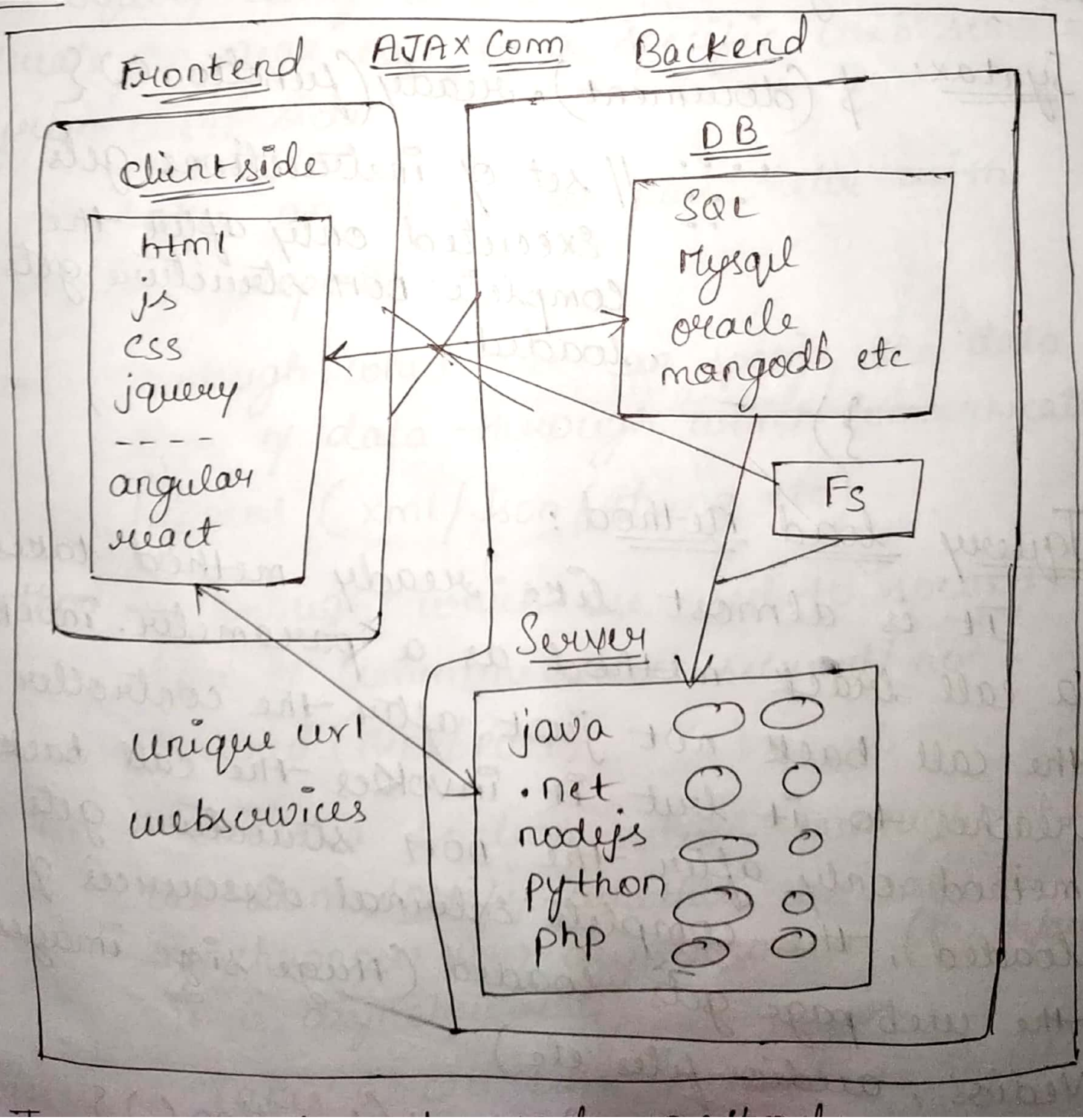
while interacting to the server, through a web server, we need to specify the type of data format in which communication happens.

following are the different data types in which communication can be done.

① JSON ② String ③ XML ④ RSS etc

Among all the data types Json is mostly used and recommended data type.

AJAX:



Creating an AJAX call through Jquery AJAX method:

"$.ajax()" is a pre-defined method supported in Jquery using which we could able to create an ajax call to a specific web service from client side.

* It takes an object as parameter with following properties

url - Through which we can specify the data type of data through which Communication happens (xml/Json/string etc)

method - through which we need to specify type of Communication secured/non-secured (GET/POST)

async - takes a boolean value, through which we can specify whether the call is synchronous (or) asynchronous (by default it is asynchronous).

Success - Takes a call back method as input, and invokes the call back method automatically when there is a response from server.

Error - Takes a call back method as input, and invokes the call back method automatically when there is an error while interacting with the server. etc.

Syntax:

```
$.ajax({
    url: 'web service url',
    method: 'GET/POST',
    data type: 'JSON/XML/string',
    Success: function(<response>){
        :::: // code to handle response
    },
    error: function(err){
        :::: // code to handle error.
    }
});
```

Ex:

```
$.ajax({
    url: 'http://www.test.com/sample',
    method: 'POST'
    datatype: 'Json'
    success: function(response){
        :::: // code to handle response,
             'response' object holds the data
             been returned from server/web
             service.
    },
    error: function(err){
        :::: // code to handle error.
    }
});
```

Converting an object to string and string back to an object

JSON.Stringify() & JSON.Parse() are the two pre-defined methods supported in js through which we could able to convert an object to a string and a string back to a js object.

JSON.stringfy(<Js object>) - Takes an object as input and return a Json object

JSON.Parse(<string object>) - Takes a string object as input and returns a Json object

Limitations of Ajax:

* While interacting with the server through ajax, the main limitation it has is due to security it doesn't allow to communicate b/w cross domains.

* In a case where the client and server from two different domains want to communicate b/w each other. following are the two ways we can establish communication b/w two cross domains.

① using Jsonp (Java script object Notation as parameter

② usings CORS (cross origin resource sharing)

JSONP data format

It is the data format been used while

communicating b/w cross domains.

* It is almost like a JSON object structure where the only difference is it comes as a parameter for a user defined method

Difference b/w JSON and JSONP

JSON:

```
{
  "name": "Aamir",
  "age": 21,
  "location": "Rajmundry"
}
```

JSONP:

```
parseData({"name": "raj", "age": 20, "location":
                                "Hydrebad"})
```

Following are the properties need to be added while creating an ajax call b/w cross domains through JSONP data format

```
datatype : 'JSONP'
crossDomain : true
json call back : '<user defined method name>'
```

CORS setup (cross origin resource sharing)

In this process, we use change the configuration of the server so that it could be able to serve even the request from cross domain

The CORS setup process changes according to the type of server we are using. following

are the set of instructions need to be placed under app.js file if the correct server is nodejs through express

```
app.use(function(req, res, next){
res.header("Access-control-allow-origin:","*");
res.header("Access-control-Allow-Header", origin,
        x_.requested-with, control-type, accept),
    next();
});
```

Jquery promises

promises are different way of creating an ajax call in jquery under which we seperate ajax request object, its corresponding call back method for error & success.

Syntax:

```
var reqobj = $
    url: ____
    datatype: ____
    data: ____
};
reqobj.done(function(response){ // success call back method
    // handle response code.
});
reqobj.for(function(err){ // error call back method
    // handle error.
});
```

Templating:

The concept of creating a static template (piece of Html content without actual data) and injecting data to the template dynamically, now the template will be placed on to the main page is called templating.

* The process of templating is been supported by multiple js external Api's like handlebars, mustache js, JADE/pug, Jquery templating Api etc
* following are the steps to be followed to implement templating feature in an application through Jquery templating API.

Step 1: download and include jquery.load template.js from the URL below

https://plugins.jquery.com/jquery-tmpl/

Step 2: create a static Html template without any dynamic data but with static labels.

Step 3: place the static Html template content within a script tag holding unique id.

Step 4: within the static template content use the following pre-defined attributes through which we can specify what Key value of an object has to be placed at what place of Html element

data-content

data-src

data-title

data - id
data - class
etc..,

Steps:

(i) call the load template method on the container which need to be loaded with template content.

(ii) load template method takes following 3 parameters

① Reference of template container (script tag holding template content)

② data object - which data to be injected to template.

③ append info - Through which we can specify whether data to be replaced (or) append (optional)

Eg: $(".container").load template ($("# ref of container"), data object, {apend : true});

Handling multiple ajax call through jquery promises

"$.when" is a pre-defined method takes multiple promise requests as parameters and involves the provided call back method only after there is a response from all the promises

Syntax:

```
$.when (promise 1, promis 2).done (function()
{
// invokes the call back method only after
   there is a response from all promises
})
```

XML : A predifined markup language used to hold the user defined information under within self descriptive tags. Earlier while transmitting data b/w client and server, XML used to hold the data, XML been transmitted b/w client and server.

Diffurence b/w XML and JSON:

JSON:

```
{
  EmpDetails : [ ]
    { empname : 'Neeha',
      location : 'Vizag',
      id : 'S-002',
      address : {
          street name : 'Test street',
          door no : '123/A',
          pin code : 12345
      }
    },
    {
      empname : 'Teena',
      id : 'S-009',
      location : 'pune',
      address : {
          streetname : 'public street',
          door no : '1/A',
          pincode : 12612
      }
    }
}
```

XML:

```
<employee Details>
    <employee id = "S-002">
    <name> Nisha </name>
    <location> Vizag </location>
    <address>
    <street name> public street </street name>
    <door No> 1/A </door No>
    <pin code> 213423 </pincode>
    </address>
    </employee>
    <employee id = "S-009">
    <name> Teena </name>
    <location> Nellore </location
    <address>
    <street name> ABC street </street name>
    <door no> 12/A </door no>
    <pin code> 123456 </pincode>
    </address>
    </employee>
</employee details>
```

Jquery Selectors

Following are the different ways we could able to refer a DOM element within the page

1. $("#abc") - Returns an element having id as abc.
2. $(".abc") - Returns an element having class as abc

3. $("div") - Returns reference of all elements having tag name as div.

4. $("input") - Returns all input elements.

5. $("input[type=text]") - Returns reference of all input elements of type text.

6. $("p.sample") - Returns all p tags having class as sample

7. $("div:first") - Returns all div tags which are in first child position.

8. $("span:last") - Returns all span tags which are in last child position

9. $("tr:even") - Returns all tr tags which are in even position

10. $("li:odd") - returns all li tags which are in odd position

11. $("p>*") - Returns all the elements which are childerns of p tag.

12. $("div#abc.Sample") - returns div tag with id as 'abc' and holding class as sample.

13. $("div Span") - Returns all span tags which are childrens of div tag.

14. $("div+Span") - Returns a span tag which is immediately followed by a sibling div tag.

15. $("div,.abc,#pqr") - Returns all div tags

and elements with class name abc and elements with id pqr

16. $(":empty") - Returns all elements which not having any children.

17. $("input[@name = test]") - Returns all input elements which has name attribute with value test.

18. $("input[@name ^= test]") - Returns all inputt elements which holds name attribute and it starts with 'test'.

19. $("input[@name $= test]") - Returns all inputt elements which holds name attribute and it ends with 'test'.

20. $("li:hidden") - Returns all li tags which are in hidden state.

21. $(":checked") - Returns all check boxes which are in checked state true.

22. $("li:eq(2)") - Returns all li tags which are in 2nd child position.

23. $("td:lt(2)") - Returns all td tags which are less than in 2nd position.

24. $("span:gt(2)") - Returns all span tags which are greater than in 2nd position

25. $(":parent") - Returns every DOM element which is in parent position and has atleast one child.

26. $('div/p') - Returns all p tags which are only children of div tag.

27. $("div:parent") - Returns all div tags which are in parent position

28. $("a:contains(hello)") - Returns all 'a' tags which contains text context as hello etc.,

Jquery Methods:

following are pre-defined methods been adopted in jquery which can be applied on DOM elements.

1. element.addclass("class name") - used to add single (or) multiple class names to elements.
2. element.removeclass("classname") - used to remove provided class from element.
3. element.hasclass("class") - returns true/false if the provided class name holds by an element.
4. element.remove() - elements get removed from DOM structure.

5. element.empty() - makes the content of element as empty.

6. element.attr() - used to set (or) get an attribute value of element.

7. element.append(<element>) - Appends provided elements as children to an element.

8. element.after(element) - Adds element after an element.

9. element.before(element) - Adds element before an element.

10. element.appendTo(<parent element>) - Adds an element as child to specified element.

11. element.css() - used to add single (or) multiple css properties dynamically to an element.

12. element.html() - used to set (or) get html content of an element.

13. element.text() - used to set (or) get text content of an element.

14. element.removeAttr('attribute name') - To remove specified attribute from an element

15. element.val() - to set or get value from input element.

etc....