MODULE - III

JAVA SCRIPT

What is JavaScript and What can it do?

JavaScript => It is an object-oriented, dynamically typed, scripting language.

→ It is primarily a client-side scripting language.

→ Javascript and Java are vastly different programming language.

→ Java is a full-fledged compiled, object-oriented language run on any platform with JVM.

→ Javascript has fewer object-oriented features of Java and runs directly inside the browser without the need for the JVM.

→ Javascript is object oriented in that everything is an object. eg: variables, objects, properties, methods.

→ Javascript approaches objects differently than other languages, does not have a formal class mechanism nor inheritance.

→ It is a strange object-oriented language.

JS → It is dynamically typed, in that variables can be easily converted from one datatype to another.

→ Java is statically typed eg int abc;

→ In Javascript, the type of data a variable can hold is assigned at runtime and can change during run time as well.

Client-Side Scripting :-

→ Client-side refers to the client machine ie) browser, running code locally rather than relying on the server to execute code and return the result

eg: flash, VBscript, Java and Javascript.

→ Some technologies work in certain browsers, others require plug-ins to function.

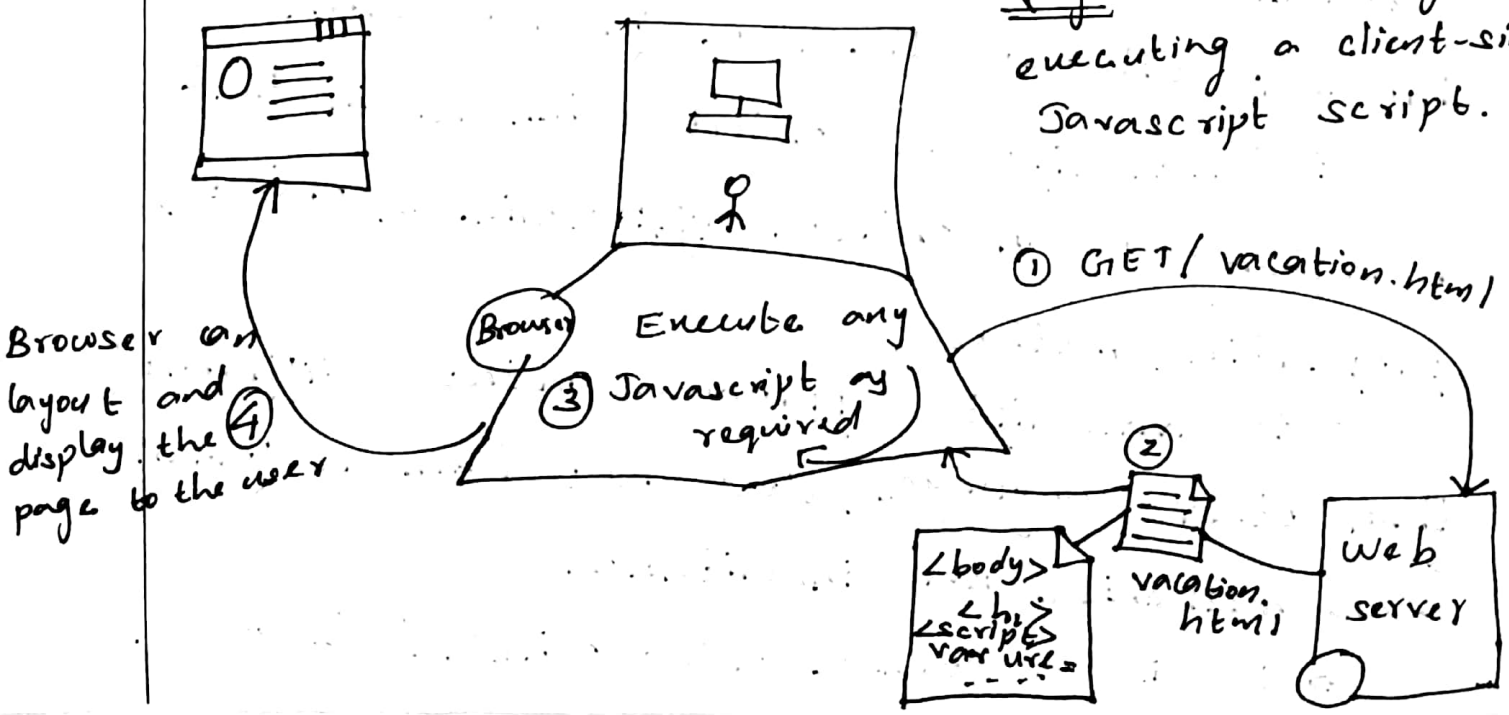


Fig: Downloading and executing a client-side Javascript script.

Study Notes
CMR

Advantages of client-side scripting:

1) processing can be offloaded from the server to client machines, thereby reducing the load on the server.

2) The browser can respond more rapidly to user events than a request to a remote server ever could, which improves the user experience.

3) JS can interact with the downloaded HTML in a way that the server cannot; creating a user experience more like desktop software than simple HTML ever could.

DisAdvantages of client-side scripting:

1) There is no guarantee that the client has JS enabled, meaning any required functionality must be housed on the server, despite the possibility that it could be offloaded.

2) The idiosyncrasis between various browsers and operating systems make it difficult to test for all potential client configurations. What works in one browser, may generate an error in another.

3) Javascript heavy web applications can be complicated to debug and maintain, which decreases code readability, and increases the difficulty of web development.

→ Js is not the only type of client-side scripting other 2 noteworthy client-side approaches to web are,

1) Adobe flash
2) Java Applets.

1) Adobe flash

→ It is a vector based drawing and animation program, a video file format, and a software platform that has its own Javascript like programming Language called ActionScript.

→ Flash is often used for animated advertisements and online games and also be used to construct web interfaces.

→ Flash objects are in a format called swf (ShockWave Flash).

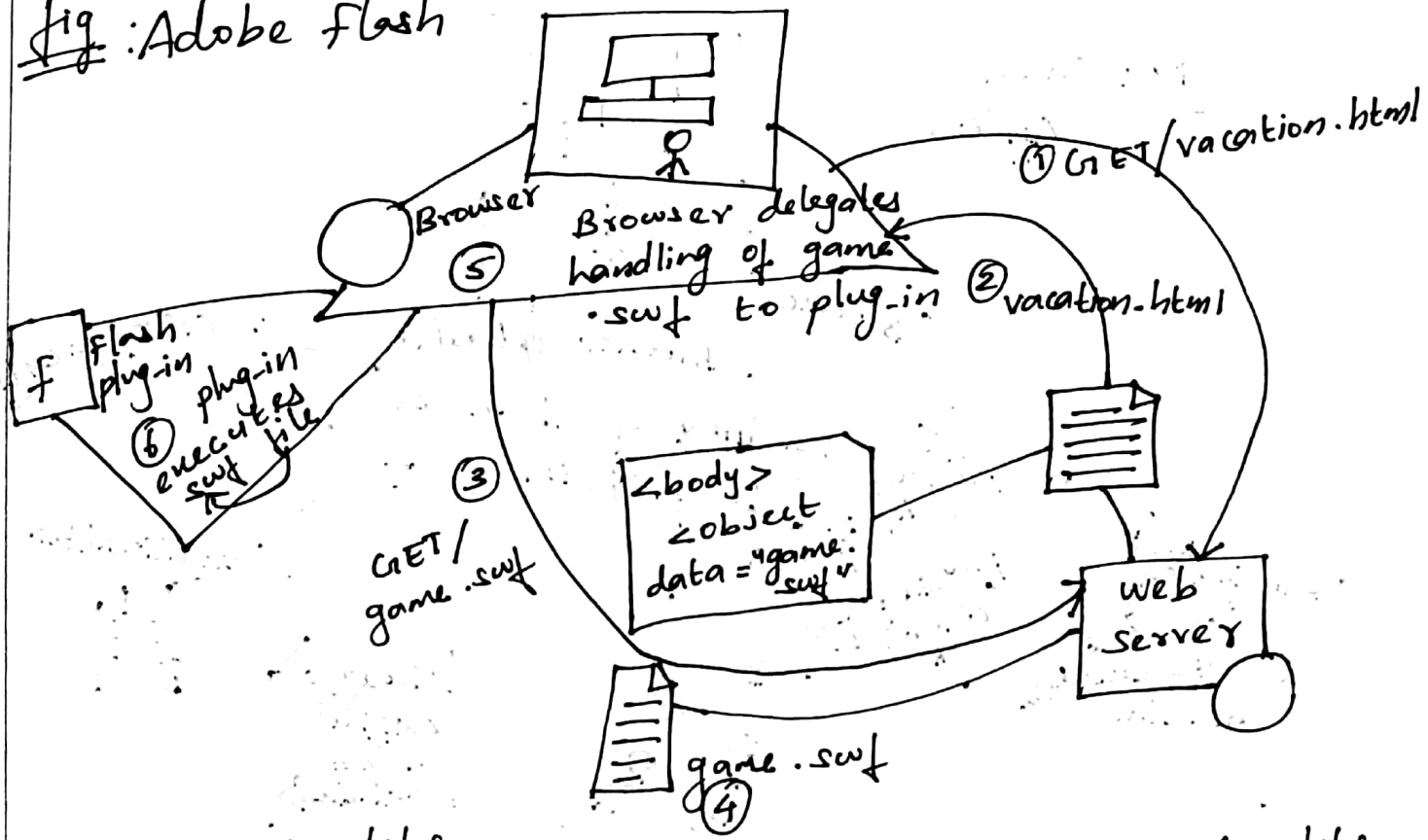
→ It is included in html document via the <object> tag.

→ The swf file is then downloaded by the browser, then the browser delegates control to a plug-in to execute the Flash file.

→ A browser plug-in is a software add-on that extends the functionality and capabilities of the browser by allowing it to view and process different types of web content.

eg: Fire Bug in Firefox browser provides a wide range of tools that help the developer understand

fig: Adobe flash



2) Java Applets

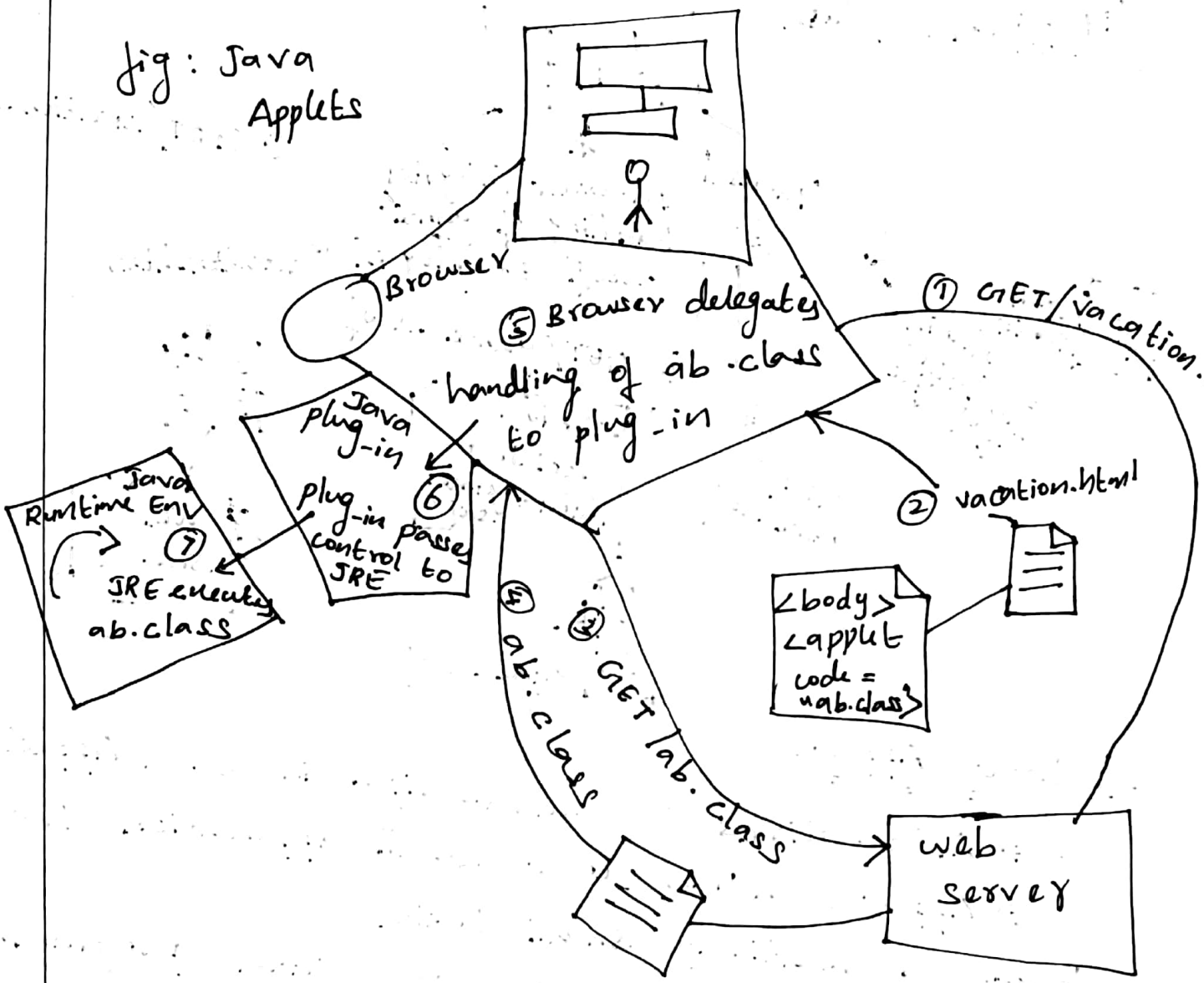
→ Alternatives to Javascript is Java Applets.

→ Applets refers to a small application that performs a relatively small task.

→ It is written using the Java programming Language and are seperate objects included within an HTML document via the applet tag, downloaded and then passed on to a Java plug-in.

→ The plug-in then passes on the execution of the applet outside the browser to the Java Runtime Environment (JRE) that is installed on the client's machine.

fig: Java Applets



→ Both Flash plug-ins and Java applets are loosing support by major players for number of reasons.

1) Java applets require JVM to be installed, which some players are not allowing for security reasons (iPhone etc

2) Flash & Java Applets require frequent updates, which can annoy the user and present security risks.

Ashrith Noel
CMR

JavaScript's History and Uses :-

→ JavaScript was introduced by Netscape in their Navigator browser in 1996.

→ It is originally called LiveScript.

→ It is an implementation of a standadized scripting Language called ECMAScript.

→ Internet Explorer first did not support JavaScript, but it has its own browser based scripting Language (VBscript).

→ IE supports now. microsoft refer it as JScript

→ Oracle owns the trademark for JavaScript.

→ The current version is 1.8.5

AJAX (Asynchronous JavaScript and XML)

↳ It refers to a style of website development that makes use of Javascript to create more responsive user experiences.

→ The responsiveness is created is via asynchrono data requests via JavaScript and the XMLHttp Request object.

→ Active X control → microsoft's IE in 1999.

→ The most important feature of AJAX sites is the asynchronous data requests.

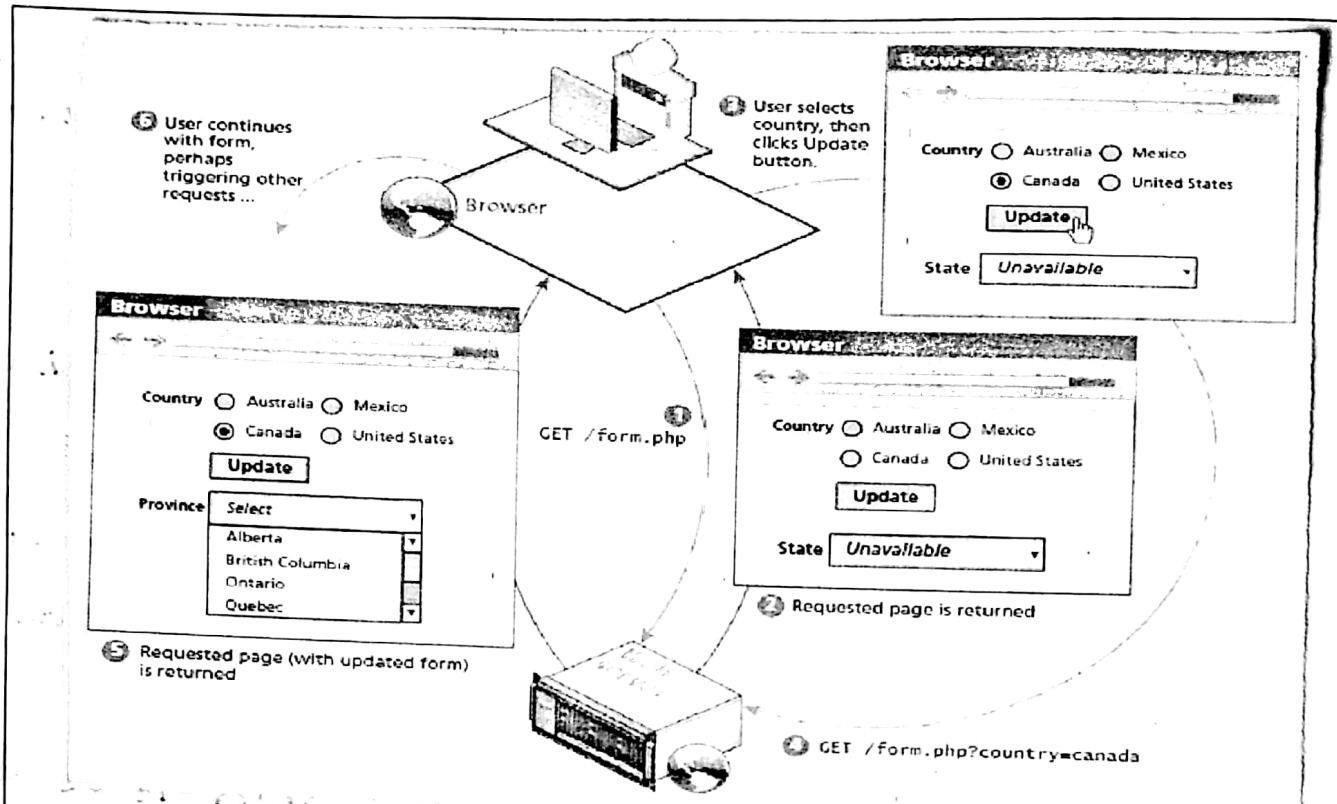


fig: Normal HTTP request-response Loop.

→ The above figure illustrates the processing flow for a page that requires updates based on user input using the normal synchronous non-AJAX page request-response loop.

→ The interaction requires multiple requests to the server which not only slows the user experience, it puts the server under extra load.

→ This results leads to performance issues.

Ashwini Noel
CMR

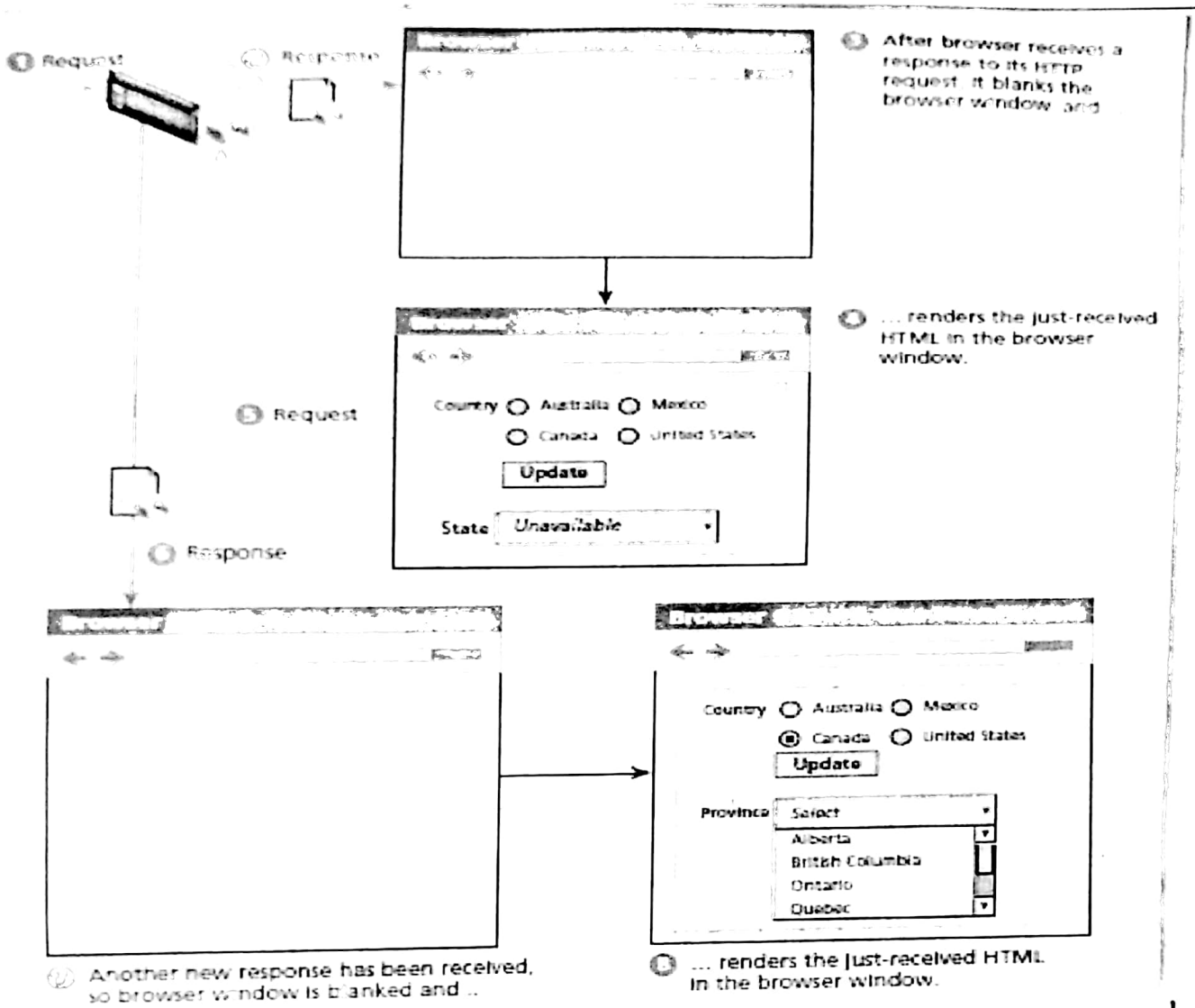


fig: Normal HTTP request-response loop, take two

→ The above figure illustrates when these multiple requests are being made across the Internet to a busy server, then the time costs of the normal HTTP request-response loop will be more visually noticeable to the user.

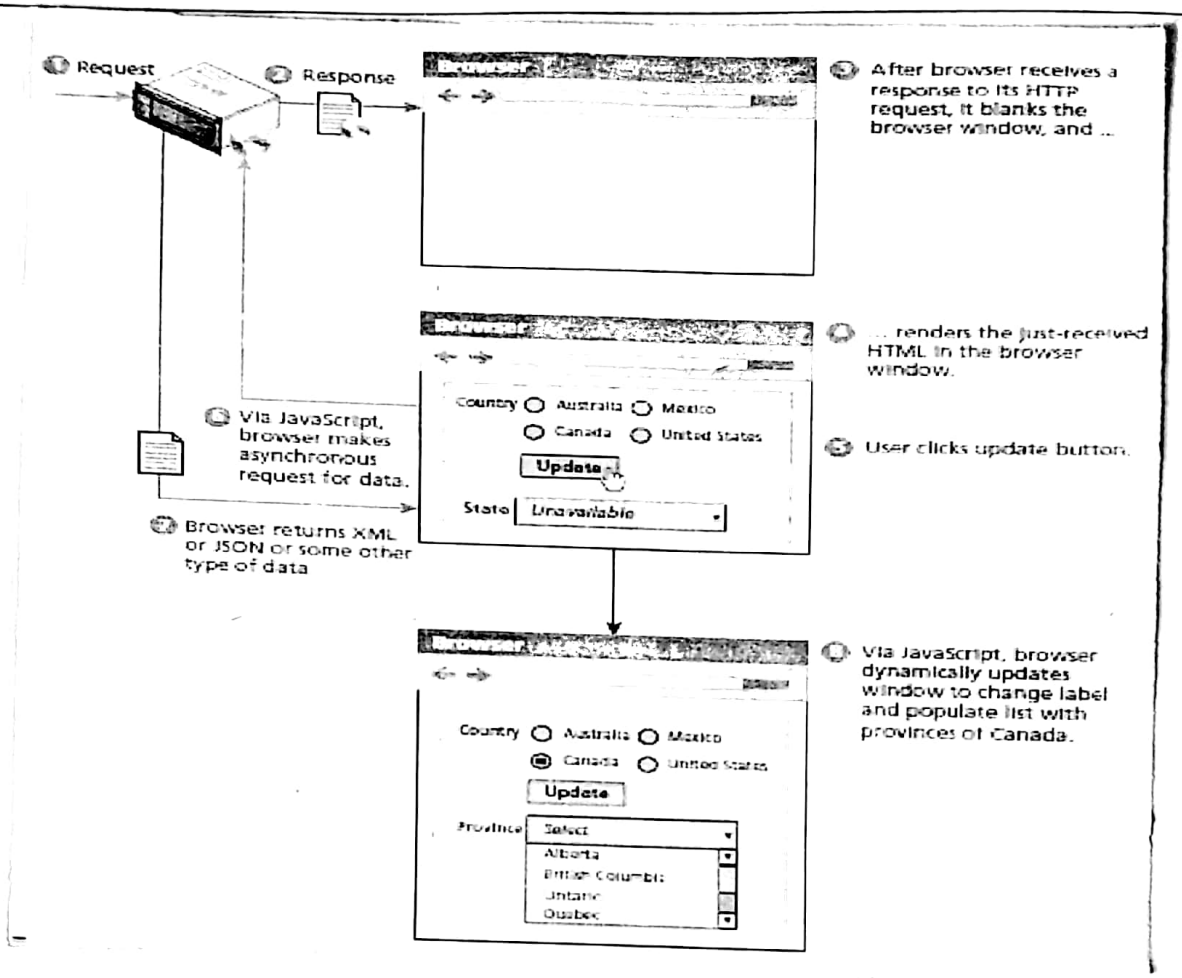


fig: Asynchronous Data Requests.

→ The above figure illustrates the http request response in AJAX.

→ AJAX provides web authors with a way to avoid the visual and temporal deficiencies of normal HTTP interactions.

→ with AJAX, web pages it is possible to update sections of a page by making special requests of the server in the background, creating the illusion of continuity.

→ JavaScript frameworks such as jQuery, prototype, ASP.NET AJAX and MooTools.

→ These frameworks reduce the amount of JavaScript code required to perform typical AJAX tasks.

→ MVC frameworks such as AngularJS, Backbone and Knockout have gained a lot of interest from developers.

Java Design Principles:-

Layers:-

→ In object-oriented programming, a software layer is a way of conceptually grouping programming classes that have similar functionality and dependencies.

→ common software design layer names include:

* presentation layer:- classes focussed on the user interface.
* Business layer:- classes that model real-world entities such as customers, sales..
* Data layer:- classes that handle the interaction with the data sources

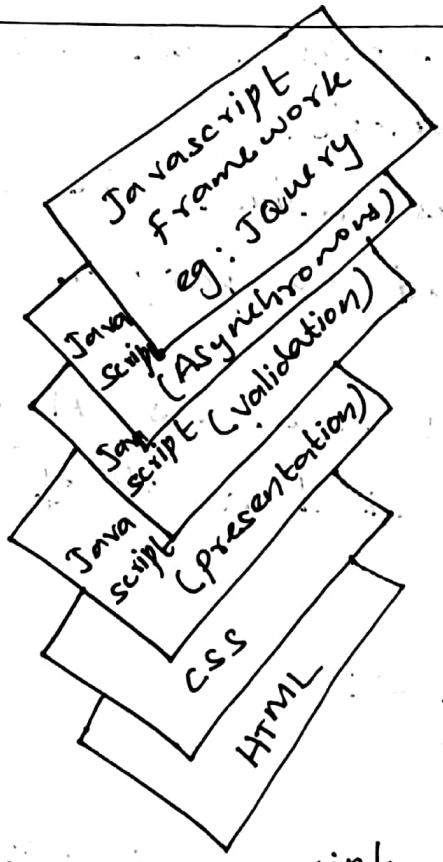


fig: Javascript Layers.

*Presentation Layer :-

→ This focus on the display of information.

→ Javascript can alter the HTML of a page, which results in a change, visible to the user.

→ These presentation layer applications include common things like creating, hiding, showing divs using tabs to show multiple views or having arrows to page through result sets.

*Validation Layer :-

→ JS can also be used to validate logical aspects of the users experience.

eg : validating a form to make sure the email entered is valid before sending it along.

→ It is often used in conjunction with the presentation layer.

* Asynchronous Layer :-

→ JS operates in a synchronous manner where a request sent to the server requires a response before the next lines of code can be executed.

→ During the wait b/w request and response the browser sits in a loading state and only updates upon receiving the response.

→ Asynchronous layer can route requests to the server in the background.

→ If certain events are triggered, the JS sends the HTTP requests to the server, but while waiting for the response, the rest of the application functions normally and the browser isn't in a loading state

→ when the response arrives, JS will update a portion of the page.

Users without Javascript :-

Users have a myraid of reasons for not using JS, and that includes:

1. web crawler :-
   → These automated software agents do not interpret JS. the crawler cannot see the enhanced look anyway.
2. Browser plug-in :-
   → It is a piece of software, which motivates some users to install plug-ins that stop JS execution.
3. Text-based clients :-
   → some clients are using a text-based browser
   eg: webpage has only links and text.
4. Visually disabled client :-
   → This client will use special web browsing software to read the contents of a web page out loud to them.
   → these specialized browsers donot interpret JS and some JS on sites is not accessible to these users.

Sherly Noel
CMR

Graceful Degradation and Progressive Enhancement

Graceful Degradation :-

→ with this strategy you develop your site for the abilities of current browsers.

→ For those users who are not using current browsers; we need to provide an alternative site or pages for those using older browsers that lack the JS used on the main site.

→ The site is "degraded" (ie) loses capability "gracefully" (ie) without pop-up Javascript error codes or without condescending messages telling users to upgrade their browsers.

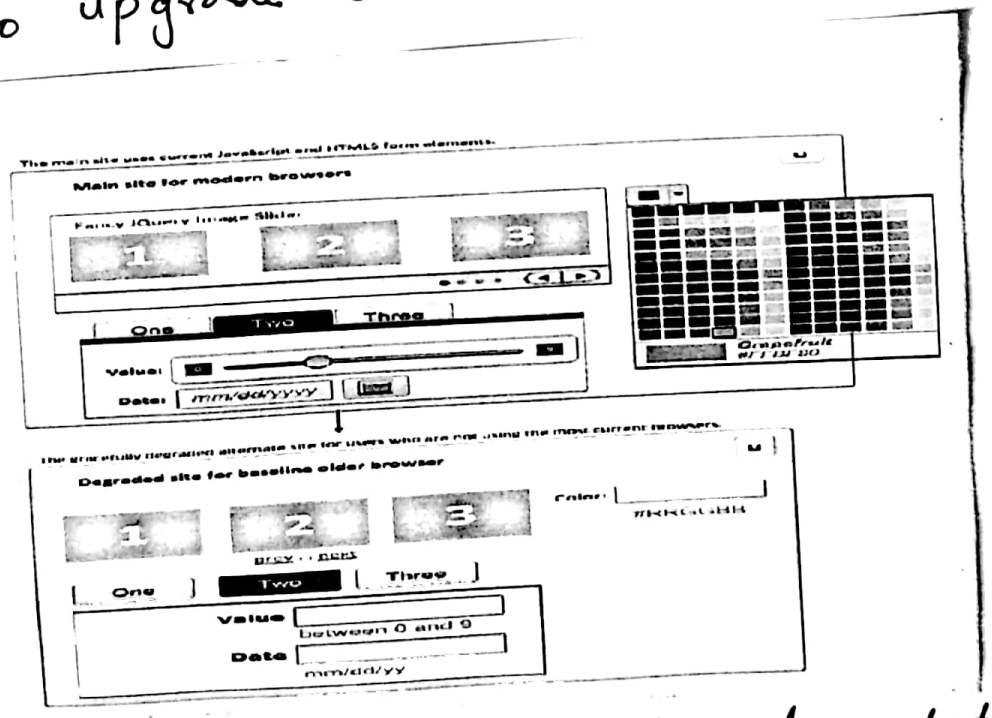


fig: Example for graceful degradation

CMR

progressive Enhancement :-

→ The developer creates the site using CSS, and HTML features that are supported by all browsers of a certain age.

→ The developers can now "progressively" i.e.) for each browser "enhance" i.e.) add functionality to their site based on the capabilities of the users browsers.

eg :: current version of chrome and opera might see a fancy HTML5 color input form elements while users of IE7 might see a simple text box.

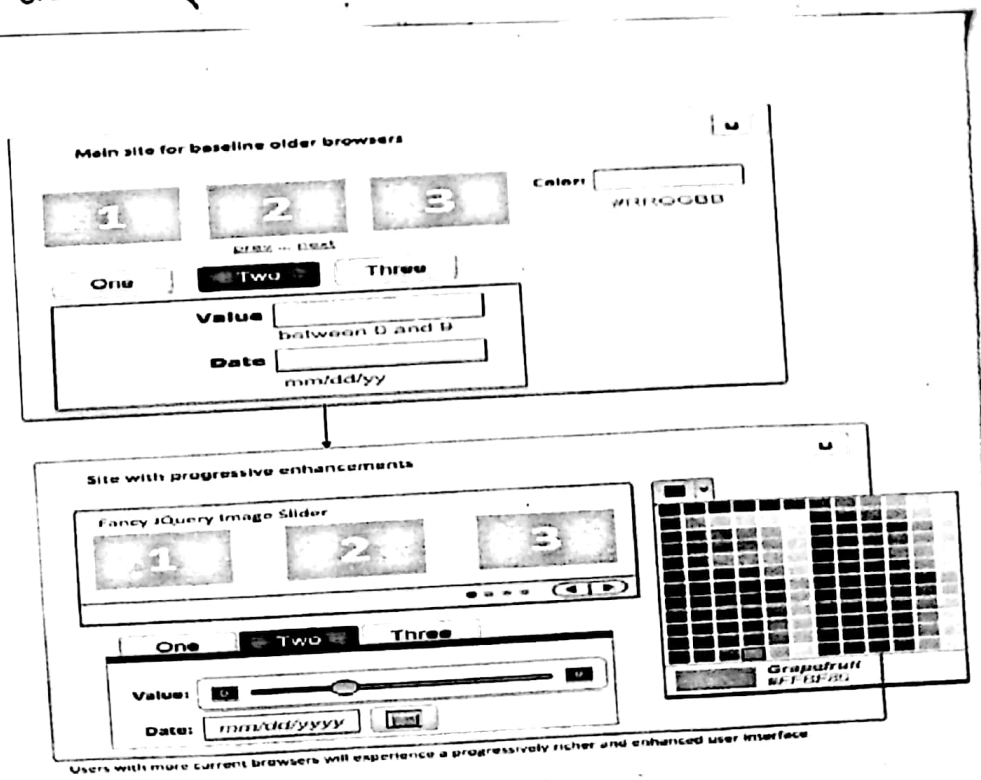


fig: site with progressive enhancements.

Sherly Noel
CMR

The <noscript> tag:

→ Any text between the opening and closing tags will only be displayed to users without the ability to load JS.

→ This tag defines an alternate content for users that have disabled scripts in their browser or have a browser that doesn't support script.

→ The content inside the <noscript> element will be displayed if scripts are not supported.

Where does Javascript Go?

→ JS can be linked to HTML page in a number of ways.

1. Inline Javascript:
2. Embedded Javascript
3. External Javascript.

1) Inline JS :-

→ It refers to the practice of including JS code directly within certain HTML attributes.

eg `<a href = "JavaScript: OpenWindow();" > More info </a>`

CMR

<input type = "button" onclick = "alert ('Are You sure')

→ This requires maintainers to scan through almost every line of HTML looking for your inline JS.

2) Embedded JS:-

→ It refers to the practise of placing JS code within a <script> element.

eg: <script type = "text / javascript">
alert ("Hello world");
</script>

3) External Javascript:-

→ This refers to the practice of placing JS code into a seperate JS file (fn.js).

eg: <head>
<script type = "text/javascript" src = "fn.js
</script>
</head>

→ The external files typically contain functio definitions, data definitions and other block of JS code.

→ script can be placed either in head or body part of html.

Shirly Noel
CMR

Advanced Inclusion of Javascript:-

→ Browsers that do not have JS, loads the page slower,

→ Two approaches are used to solve this pbm.

① Load one or more scripts into an <iframe>

② Load a JS file from within another JS file, so this JS file will be loaded upon demand.

Syntax

→ JS developer will encounter the following,

* Everything is type sensitive, including function, class and variable names.
* The scope of variables in blocks is not supported
* There is a === operator, which tests not only for equality but type equivalence.
* Null and undefined are two distinctly different states for a variable.
* semicolons are not required.
* There is no integer type, only number.

eg:

```
Var x ;          // a variable x is defined.
Var y = 0 ;      // y is defined & initialized to 0.
y = 4            // y is assigned the value of 4.
```

Variable declaration and assignment.

Variables:

→ variables in JS are dynamically typed.

→ Assignment can happen at declaration time by appending the value to the declaration, or at run time with a simple right-to-left assignment.

→ Conditional assignment operator can also be used to assign based on condition.

eg: `x = (y <= 4) ? "y is 4" : "y is not 4";`

(y <= 4) — condition; "y is 4" — value if true; "y is not 4" — value if false.

Comparison Operators:-

→ JS is equiped with number of operators to compare two values.

eg Operator : ==, === (exactly equals, including type), <, >, <=, >=, !=, !==

Logical operators:-

→ The boolean operators are and, or, not.

→ Their truth table is listed below,

AND

A	B	A && B
T	T	T
T	F	F
F	T	F
F	f	f

OR

A	B	A || B
T	T	T
T	f	T
F	T	T
F	f	F

NOT

A	!A
T	F
f	T

Conditionals:

→ JS syntax is identical to C, PHP, JAVA.

→ In this syntax, the condition to test is contained within () brackets with the body contained in { } blocks.

eg:
```
var hoursofday;
var greeting;
if (hoursofday > 4 && hoursofday < 12)
{ greeting = "Good Morning";
}
else if (hoursofday >= 12 && hoursofday < 20)
{ greeting = "Good Afternoon";
}
else
{ greeting = "Good Evening";
}
```

Loops:

Loops use the () and { } blocks to define the condition and the body of the loop.

* while loop
* for loop.

While loops:

→ Loops normally initialize a loop control variable, use it in the condition and modify it within the loop.

eg:
```
Var i=10;
while (i<10)
{
    i++;
}
```

For loops:

→ For loop combines initialization, condition, and post-loop operation into one statement.

eg:
```
for (var i=0; i<10; i++
{
    alert(i);
}
```

Functions:

→ Functions are the building block for modular code in Javascript.

→ They are defined by using the reserved word function and then the function name and (optional) parameters.

→ Since JS is dynamically typed, functions do not require a return type nor do the parameters require type.

Shruthy Noel
CMR

```
function power (x, y)
{ var pow = 1;
  for (var i=0; i<y; i++)
  { pow = pow * x;
  }
  return pow;
}
```

→ call the function without using the keyword function.

Alert :

→ The alert() function makes the browser show a pop-up to the user with whatever is passed being the message displayed.

```
alert ("Good Morning");
```

Errors using Try and catch :

→ When the browsers Javascript engine encounters an error, it will throw an exception.

→ These exceptions interrupt the regular, sequential execution of the program and can stop the Javascript engine altogether.

→ You can optionally catch these errors preventing disruption of the program using the try-catch block,

```
try {
    nonexistantfunction ("hello");
}
catch(err)
{ alert ("An exception was catch: " + err);
}
```

```
try {
    var x = -1;
    if (x < 0)
        throw "smallerthan0Error";
    }
    catch(err)
    { alert (err + "was thrown");
    }
```

eg: Throwing a user-defined exception.

Javascript Objects :-

→ Objects can have constructors, properties and methods associated with them.

Constructors :-

→ To create a new object, use the keyword new, the class name and brackets () with n optional parameters inside, comma delimited.

Var someObject = new ObjectName (parameter 1, param 2, ..., parameter n);

var greeting = "Good Morning";

Instead of the formal definition

var greeting = new String ("Good Morning");

properties :-

→ Each object might have properties that can be accessed, depending on its definition.

→ when a property exists, it can be accessed using dot notation, where a dot between the instance name and the property references that property.

eg: alert (someObject.property);

methods :-

→ objects can also have methods, which are functions associated with an instance of an object.

→ These methods are called using the same dot notation as for properties, but instead of accessing a variable, we are calling a method,

someObject.doSomething ( );

Objects included in Javascript :-

→ A number of useful objects are included with JS. eg : Array, Boolean, Date, Math, String etc.

⇒ Arrays :-

Arrays in JS can be created in 3 ways;

① var greet = new Array();

② var greet = new Array ("good morn", "good Aft");
↳ initialization of array with values.

③ var greet = ["good morn", "good afternoon"];

Accessing and traversing an array :-

→ To access array values, use index.

eg : var b = greet[0];
↳ index ⇒ good morn.

→ To determine the maximum index or length, length property is used.

```
for (var i=0 ; i< greet .length; i++)
    document.write (greet[i]);
```

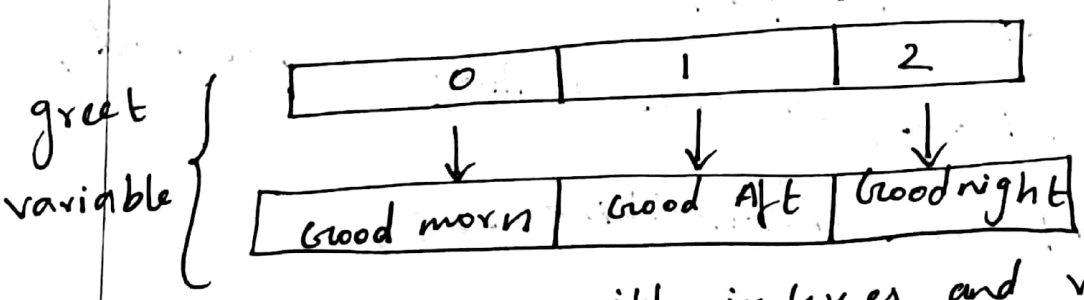


fig : Js array with indexes and values illustrated.

modifying an Array

→ To add an item to an existing array, use the push method.

greet.push("good night");

→ pop method can be used to remove an item from the back of an array.

→ Additional array methods includes concat(), slice() join(), reverse(), shift() and sort().

⇒ Math :- class

→ This allows to use mathematical functio

→ The static class contains methods such as max(), min(), pow(), sqrt(), exp(), sin(), cos() etc.

eg: Math.PI(); // 3.141592
Math.sqrt(4); // 2
Math.random(); // blw 0 to 1.

⇒ String :-

→ string can be created using,

var greet = new String("good"); // long constructor
var greet = "good"; //short constructor

→ string methods includes length, charAt(), match(), search(), indexOf(), split(), etc.

eg. var g = greet.length;

⇒ Date :

→ Date class can be created by,

```
var d = new Date();
```

→ To display date as string,

```
document.write("Today is : " + d.toString());
```

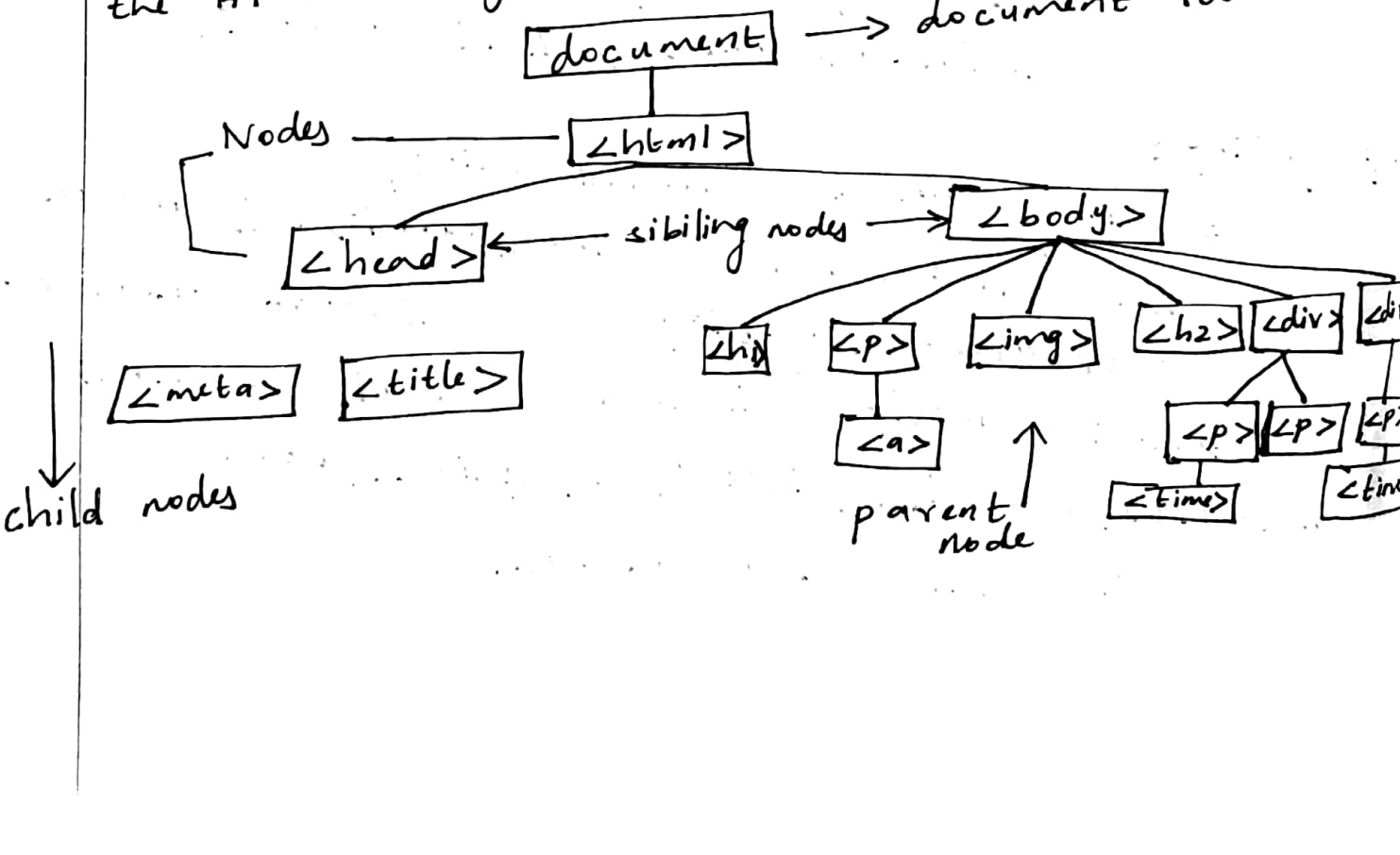
Window Object :-

→ It corresponds to the browser itself.

→ object of window is alert();

The Document Object Model (DOM):

→ DOM is a way of programmatically accessing the elements and attributes within the HTML through API.



Nodes :-

In the Dom, each element within the HTML document is called a node.

→ If Dom is a tree, then each node is an individual branch.

eg: <p> photo of conservatory pond in

```
<a href = "http://www.centralpark.com/">
    central park
</a>
</p>
```

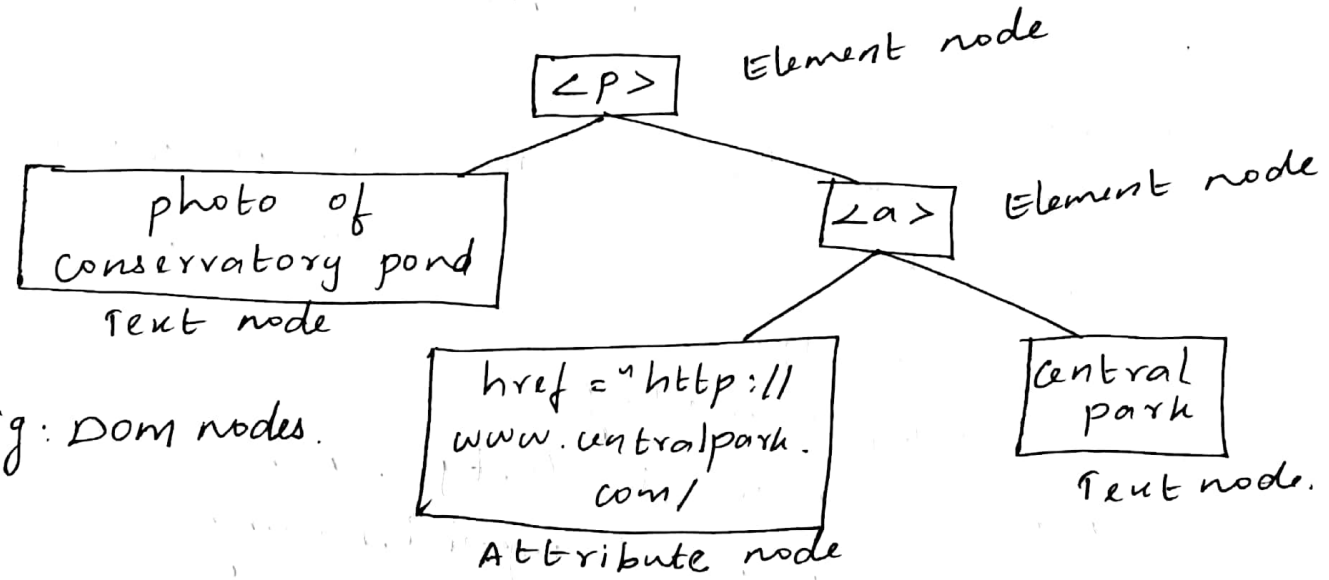


fig: Dom nodes.

→ Js involve finding a node and then accessing or modifying it via properties & methods.

	property	Description
1.	Attributes	collection of node attributes
2.	childNodes	A nodelist of child nodes for this node
3.	firstChild	First child of this node.

4.	lastChild	Last child of this node.
5.	nextSibiling	Next sibiling node for this node.
6.	nodeName	Name of the node.
7.	nodeType	Type of the node.
8.	nodeValue	Value of the node.
9.	parentNode	parent node for this node.
10.	parentSibiling	previous sibiling node for this node.

Document Object :-

→ The Dom document object is the root Javascript object representing the entire HTML document.

→ It is globally accessible as document.

→ The attributes of this object are,

* doctype.
* inputEncoding.

eg: Var a = document.doctype.name
Var b = document.inputEncoding.

other methods include,

	Method	Description
1.	createAttribute()	- Create an attribute node
2.	createElement()	- Create an element node
3.	createTextNode()	- Create a text node
4.	getElementById()	- Returns the element node whose id attribute matches the passed id att
5.	getElementsByTagName()	- Returns a nodelist of elements whos tagname matches the passed name parameter.

eg
```
<div id = "latest">
    <p> CMRIT </p>
</div>
```

```
var abc = document.getElementById("latest");
var list = document.getElementsByTagName("div");
```

Element Node Object :-

→ The type of object returned by the method document.getElementById() is an element node object.

→ Since ID's must be unique in an HTML document, getElementById() returns a single node.

Element node properties

	Property	Description
1.	className	- The current value for the class attribute of this HTML element.
2.	id	- The current value for the id of this element.
3.	innerHTML	- Represent all the things inside of the tags.
4.	style	- The style attribute of an element.
5.	tagName	- The tag name for the element.

HTML Dom Element properties for certain Tag

Property	Description	Tags
href	The href attribute used in a tag to specify a URL to link to.	a
name	This is a bookmark to identify this tag.	a, input, for textarea.
src	Links to an external URL that should be loaded into the page.	img, input, ifra script.
value	This is value attribute of input tags.	input, textare submit.

Modifying a Dom element:

→ document.write() method is used to create output to the HTML page from JS.

eg: changing the HTML using innerHTML.

```
var lat = document.getElementById("latest");
var old = lat.innerHTML;
lat.innerHTML = old + " <p> updated with JS </p>";
```

↳ the text message will be added to the html page

O/P

cmrit
updated with JS.

Sheeba Noel
CMR

A more verbose technique :

→ Dom functions createTextNode(), removeChild() and appendChild() allow us to modify an element in a rigorous way.

eg:
```
<body>
    <ul id = "list">
        <li> coffee </li>
        <li> tea </li>
    </ul>

    <button onclick = "func()" > click </button>
    <script>
        function func()
        {
            var node = document.CreateElement("LI");
            var textnode = document.createText
                        Node ("water");
            node.appendChild(textnode);
            document.getElementById("list").
                        appendChild(node);
        }
    </script>
```

appendChild()

removeChild()

```
<script>
    function func()
    {
        var list1 = document.getElementById
                        ("list");
        list1.removeChild(list.childNodes[0]);
    }
</script>
```

Changing the Element's style :-

→ We can add or remove any style using the style or classname property of the element node.

① eg : var mylist = document.getElementById("list");
mylist.style.background-color = "red";
mylist.style.color = "green";

② mylist.className = "c2";

Additional properties :-

→ few methods to deal with certain tags.

< input type = "password" name = "pw" id = "pw" />

var pass = document.getElementById("pw");

alert (pass.value);

JavaScript Events:

→ A JS event is an action that can be detected by Javascript.

→ An event is triggered and then it can be caught by JS functions, which then do something in response.

→ A visual comparison of old and new technique is given below.,

old, Inline technique

```
<form name = 'mainform' onsubmit = validate(this)>
    <input type = "text" name = "name" onhover =
        onhover(this) onfocus = focus(this)/>
    <input type = "submit" onclick = validate(this)>
</form>
```

new, Layered listener technique

```
<script type = "text/javascript" src = "listerner.js">
</script>
<form name = 'mainform'>
    ...
</form>
```

Listener Approach :-

Two approaches for registering an event. old style and new Dom 2 approach to registering listeners.

```
var greetingBox = document.getElementById
                    ("ex1");
greetingBox.onclick = alert ("Good Morning");
```

fig : old style of registering a listener

```
Var greetingBox = document.getElementById('ele
greetingBox.addEventListener('click', alert
                    ('Good Morning')).
greetingBox.addEventListener('mouseOut', alert
                    ('GoodBye'));
```

List : The new Dom2 approach to registering Listeners.

An alternative to this approach is to use an anonymous function.

eg:
```
Var element = document.getElementById('el
element.onclick = function() {
        var d = new Date();
        alert("you clicked this on" +
                    d.toString());
};
```

Event Object :-

Events passed to the function handler as a parameter named e.

```
function someHandler(e)
{
    // e is the event that triggerd this
                    handler.
}
```

→ These objects have many properties and methods. few of them are,

1. Bubbles:

This property is a boolean value. If an events bubbles property is set to true, then there must be an event handler in place to handle the event or it will bubble up to its parent and trigger an event handle.

→ If the parent has no handler, it continues to bubble up until it hits the document root and it goes away unhandled.

2. Cancelable:

→ It is also a boolean value that indicates whether or not the event can be cancelled.

→ If an event is cancelable, then the default action associated with it can be canceled.

3. preventDefault:

→ A cancelable default action for an event can be stopped using the preventDefault() method.

```
eg: function submitButtonClicked(e)
    {   if(e.cancelable)
        {   e.preventDefault();
        }
    }
```

eg: A sample event handler function that prevents the default event.

Event Types:

→ There are several classes of event with several types of event.

→ The classes are,

* mouse event.
* keyboard events.
* form events.
* frame events.

* Mouse Events :-

→ This event is defined to capture a range of interactions driven by the mouse.

→ These can be further categorized as mouse click and mouse move events.

	Event	Description
1.	onclick	– The mouse was clicked on an elem
2.	ondblclick	– The mouse was double clicked on an element.
3.	onmousedown	– The mouse was pressed down over an element.
4.	onmouseup	– The mouse was released over an elem
5.	onmouseover	– The mouse was moved over an element
6.	onmouseout	– The mouse was moved off an element.
7.	onmousemove	– The mouse was moved while over an element.

* Keyboard Events :-

→ These events are most useful within input fields.

	Event	Description
1.	onkeydown	– The user is pressing a key.
2.	onkeypress	– The user presses a key & releases a key
3.	onkeyup	– The user releases a key

eg: <input type = "text" id = "keyEx" >

```
document.getElementBy("keyEx").onkeydown = function
myfunction(e)
{
    var keyPressed = e.keyCode;
    var character = String.fromCharCode(
                    keyPressed);
```

alert ("key" + character + "was pressed");
}

*Form Events :-

→ Through form the user input is collected and transmitted to the server.

Event	Description
1. onblur	A form element has lost focus.
2. onchange	some <input>, <textarea>, <select> field had their value change.
3. onfocus	This is triggered when an element gets focus.
4. onreset	HTML forms have the ability to be rese This event is triggered when that happens
5. onselect	When the user selects some text.
6. onsubmit	When the form is submitted this event is triggered.

eg: document.getElementById("loginform").onsubm
= function(e) {
var pass=document.getElementById("pw").
value;
if (pass=="")
{

```
        alert ("enter a password");
        e.preventDefault();
    }
}
```

* Frame Events :-

→ These events are related to the browser frame that contains your web page.

	Event	Description
1.	onabort	- An object was stopped from loading.
2.	onerror	- An object or image did not properly load.
3.	onload	- when a document or object has been loaded.
4.	onresize	- The document view was resized.
5.	onscroll	- The document view was scrolled.
6.	onunload	- The document has unloaded.

eg:
```
window.onload = function()
{
    // all Javascript initialization here.
}
```

Forms :

→ The user form input should be validated on both the client side and the server side.

eg: <form action = 'login.php' method = 'post'
id = 'loginform' >
<input type = 'text' name = 'username'
id = 'username' />
<input type = 'password' name = "password"
id = 'password' />
<input type = 'submit' />
</form>

Validating Forms:-

→ Form validation is one of the most common applications of Javascript.

→ Javascript does prevalidation, which includes email validation, number validation and data validation.

Empty field validation:

→ There is no point sending a request to login if the username was left blank.

→ The way to check for an empty field in JS is to compare a value to both null and the empty string ("") to ensure it is not empty.

eg:

```
document.getElementById("loginform").onsubmit
                                = function(e)
  {
    var fieldValue = document.getElementById
                ("username").value;
    if (fieldvalue == null || fieldvalue == "")
    { // the field was empty. stop form submission
      e.preventDefault();
      alert("you must enter a username");
    }
  }
```

→ To check for checkbox, to ensure checkbox is ticked,

```
var inputfield = document.getElementById("license");
if (inputfield.type == "checkbox")
{ if (inputfield.checked)
    // now we know the box is checked.
}
```

Number validation :-

→ Number validation can take many forms.

eg : parseInt(), isNaN(), isFinite()

eg:

```
function isNumeric(n)
  { return !isNaN(parseFloat(n)) && isFinite(n);
  }
```

Submitting forms :-

→ Submitting a form using Javascript requires having a node variable for the form element.

eg. var formExample = document.getElementById("loginform");
formExample.submit();

→ This is done in conjuction with calling preventDefault() on the submit event.

→ This can be used to submit a form when the user did not click the submit button