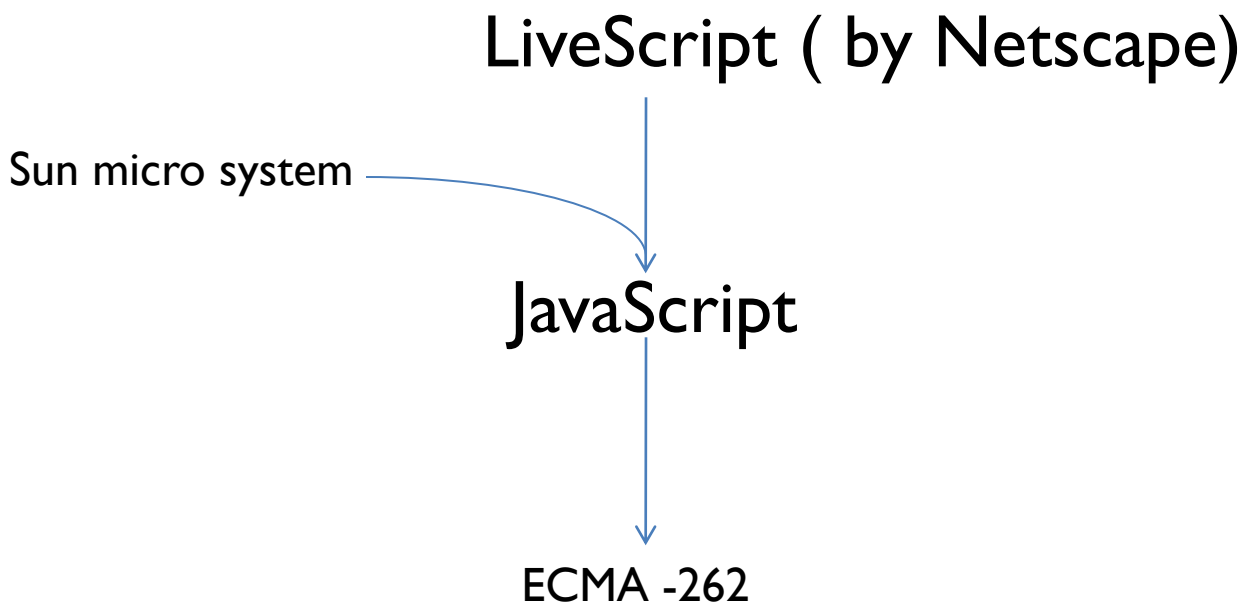
JAVASCRIPT & JQUERY

Module 4

APPLICATIONS

- Validation of user input
- Pattern matching
- Alert generation
- Mismatch finding
- Load balancing with server
- Alternative to java applet
- Event detection & Interaction
- Create new content in browser display dynamically
- DOM- to change CSS
- Etc……

ORIGIN



- MICROSOFT – Jscript
- XHTML, CSS, JavaScript are scripting language- not compiled
- C,C++, are programming language- complied

- Script: collection of JavaScript code
- XHTML document can include any no of embedded scripts

- JavaScript parts

Core:

- heart of language
- include operators
- expressions
- statements
- sub-programs

Client:

- collection of objects
- control of browser
- interaction with user e.g mouse click event

Server:

- collection of objects
- support server side functions
- support database management like function

JavaScript vs. Java

*syntax of both are same for expression, control & assignment statement

JavaScript

a. Dynamically typed
b. Types are known at compile time
c. Objects are dynamic
d. Data& methods will change(dynamic)
e. Object oriented model is different from java

Java

a. Strongly typed
b. Known at runtime
c. Static
d. fixed

- Much of JavaScript is event driven

  - Mouse click & Form submission

- Validation at client side save

  - Server time
  - Internet time

- JavaScript in head /body

  head: produce content on user interaction

  body: interpreted once

- Script at head don't interpret while processing head

OBJECT ORIENTATION

- JavaScript is object based language ,
  - -No classes
  - -No class based inheritance → No polymorphism
- But support "*prototype based inheritance*"
- Object serve as both as object & model of object.
- Many features related to oop concept.

object (collection of property)

- data property
  - primitive value
  - reference (other obj)
- method property
  - method (called by obj)
  - function (not by obj)

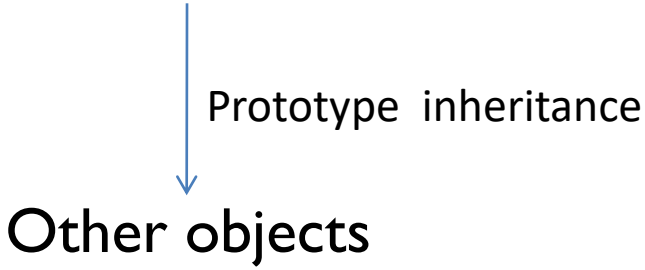
- Primitive property:
  - Non-object type
  - Implemented in hardware
  - So faster operation
  - Are accessed directly
  - Also called as *value types*
- reference property:
  - Used to refer another object
  - In software
  - Like reference type in java
  - Property is accessed by attaching property name

e.g myCar . engine

variable referencing object

object property

- Root object in Js is *Object*



- With in a page *Js* should in

```
<script type="text/javascript">
        Js code……
</script>
```

- If external(indirect method) js,

```
<script type="text/javascript"  src="url.js">
</script>
```

- Any no of script tag can be include in a xhtml page

- Advantage of indirect method of embedding
  - -Hiding script from user
  - - Hiding script from old browser
  - -Good to separate computation from layout & presentation

- Identifier: same rule as other language (a ,_, $), no length limit.
- Case sensitive
- Keyword (case , break, if , var ,while……..)
- Comments:
  - //
  - /* */

- Js script always embed in

```
<!--
        -- JavaScript ---
 //     -->
```

Because: 1. old browser does recognize <script> tag , but no JavaScript interpreter : Not a problem

2. old browser doesn't recognize <script> tag, display script as test to user: Is a problem

3. XHTML validator has problem if xhtml element in script e.g

4. it is better to place Js code in separate file

- Using semicolon is optional as statement terminator.

- E.g.

```
<?xml version="1.0" encoding ="utf-8" ?>
<!Doctype html public "        "  >
<html xmlns=" ">
<head>
    <title> First Program </title>
</head>
<body>

<script type="text/javascript">

        document . write("hello");
</script>
                        object
</body>                                  method
<html>
```

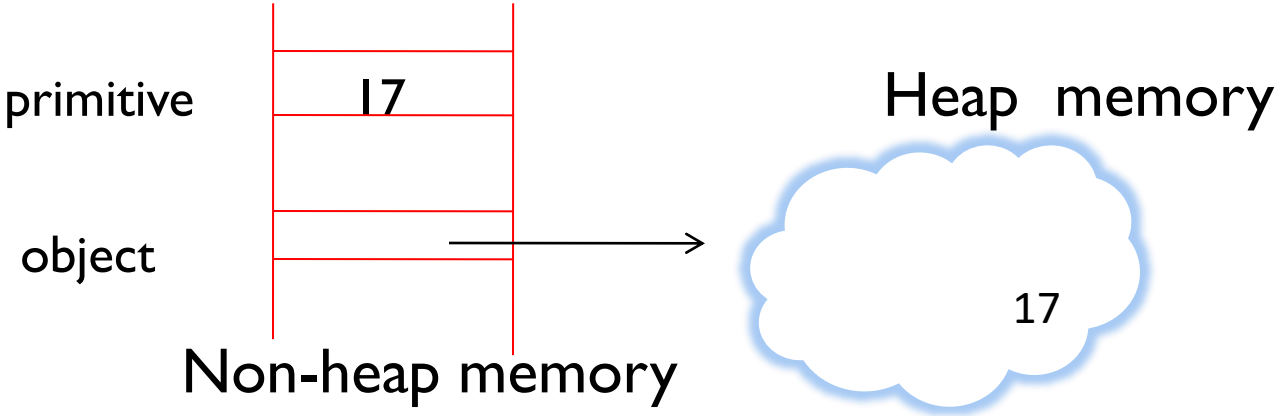
Primitives, operation & Expression

- 5 primitive types (slide 7):

  1. number → Number
  2. string → String
  3. boolean → Boolean
  4. null
  5. undefined

  wrapper objects

- Wrapper provide useful property & method for manipulation.
- Primitive value is stored in non-heap memory, object in Heap



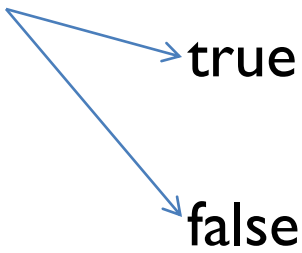
number :

* All numeric types are Number (No : long , double, int)
* Numbers are Double precision floating point
* Also called as *numbers*
* Either integer , floating point forms , exponents ,hex (0x)

string:

* use ‘ abc’ or “ abc ”
* use escape sequence

Boolean



Null:

* Only possible value is **null** (no value)
* Undeclared variable result in null
* Using null value cause runtime exception

Undefined:

* only declared not defined variable return *undefined*

- Variable is dynamically typed (contain any value)
  - Variables are not typed but values
  - Variable contain either primitive or reference value
  - Using *var* keyword

  E.g. var counter, index,pi=3.14, name="bob" flag= true;

- Variable that has been declared, but not assigned a value , has the value *undefined*

Numeric operator :

binary operators: +, -, *, /, %

unary operators: +,-, ++,-- (pre & post)

If a=7,

(++a)*3 = 24

(a++)*3=21

- If a * b + c (precedence rule)
- If a / b /c (associativity rule)

Right associative: ++ , --

Left associative : * / % + -

- Few predefined objects:

1. Math : contain methods for numeric object
   :used to perform mathematical operation
   E.g. : Math.sin(60), Math. ceil(2.5)

2. Number : collection of constant value properties
   : NaN (not a number)
   1. Arithmetic op result in error (5/ 0)
   2. Results can't represent in double precision
   3. Nan ≠ Nan , false always

   E.g : MAX_VALUE, POSITIVE-INFINITY

3. Date : Used to manipulate date
   : var today= **new** Date();
   E.g : today. getday() return value in (0-6)
   { 0- Sunday, 1-Monday……..}

- String is a unit scalar value
- String concatenation operator +

Type conversion (casting)

Implicit (coercions) Explicit

<u>Implicit</u>

- Js try to convert variable based on situation

E.g 1. " Augest " + 2018 → Augest 2018

(converted to string automatically)

E.g 2. "5 " * 3 → 15

(converted to number automatically)

E.g 3. "Augest " * 3 → NaN

(converted to NaN)

- null /false → 0
  undefined → NaN
  true → 1
  when used as number

- 0 / empty string /Nan
  null / undefined → false
  All other number/ string → true
  when used as Boolean

- Relational operator also cause implicit type conversion

Explicit conversion

- Force type conversion
- *Number to string*

  * Using String constructor

    var str-value=String(5); → " 5 "

  * Using toString() method

    var num=6;

    var str-value=num.toString(); → "6"

    var str-value1=num.toString(2); → 110

- *String to number*

  *Concatenating with empty string .i.e. 7 + " " → "7"

  *Number constructor var num =Number("7");

  *Subtracting 0 from string i.e var num= astring-0;

*No alphabetic character in string (space ok)

- parseInt () & parseFloat()

  * Search for number at start of string
  * String can contain other non digit character

E.g var nu=parseInt("567ab"); → 567

<u>string methods & properties</u>

* length property e.g var str="George";
  var len=str.length → 6

- For string method, character position start at Zero
  - -charAt(2)
  - -indexOf("e")
  - -subString(2,4)
  - -toLowerCase();

- typeof operator return type of its single operand.

```
var a=5,  b='g' ,c=true, d;
var str=String('df');
var num=Number(5);
var bol=Boolean(true);
```

typeof(a); → number

typeof(b); → string

typeof(c); → boolean

typeof(d); → undefined (dynamic type)

typeof(null); →

typeof(str); →

typeof(num); →

} object (have no type)

- Assignment operator: simple (=) & compound (+=)

Output & Input

- Output screen for Js is browser screen .

  java: console, file…

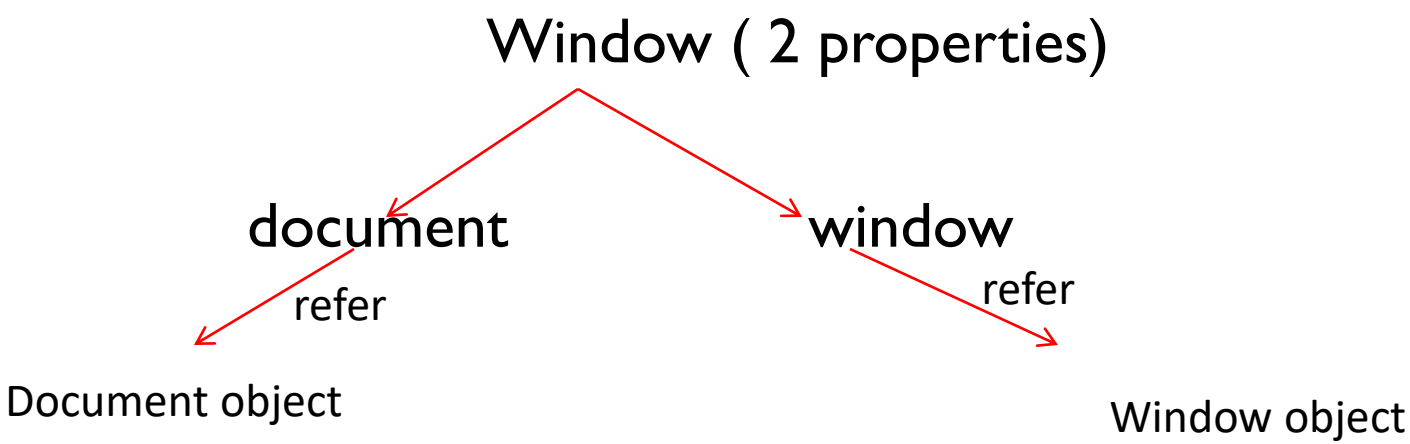
- Document object

  -Represent XHTML document (e.g index.html)

  -many methods & properties

  -write() : craete dynamic xhtml content

- Window object models the window (tab) in which the browser displays an XHTML document.



document . write(“result is”, res, “<br />”,”D”);

op: *result is 42*

D

- Contain any XHTML tags
- Create XHTML content
- Can take any number of parameters

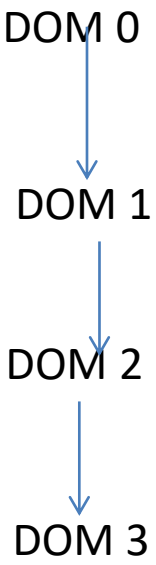
DOM(Document object model)

- Js provide collection of objects, methods & properties to interact with XHTML document

- We can manipulate XHTML element using DOM

- DOM is an API / Abstract representation

- Js implement DOM as collection of object



DOM :

1. Used to respond to document related events
2. Dynamically modifying the content
3. Provide portability

- DOM tree- Entire XHTML document
- DOM node- Each element –as object

-DOM 0 ✔

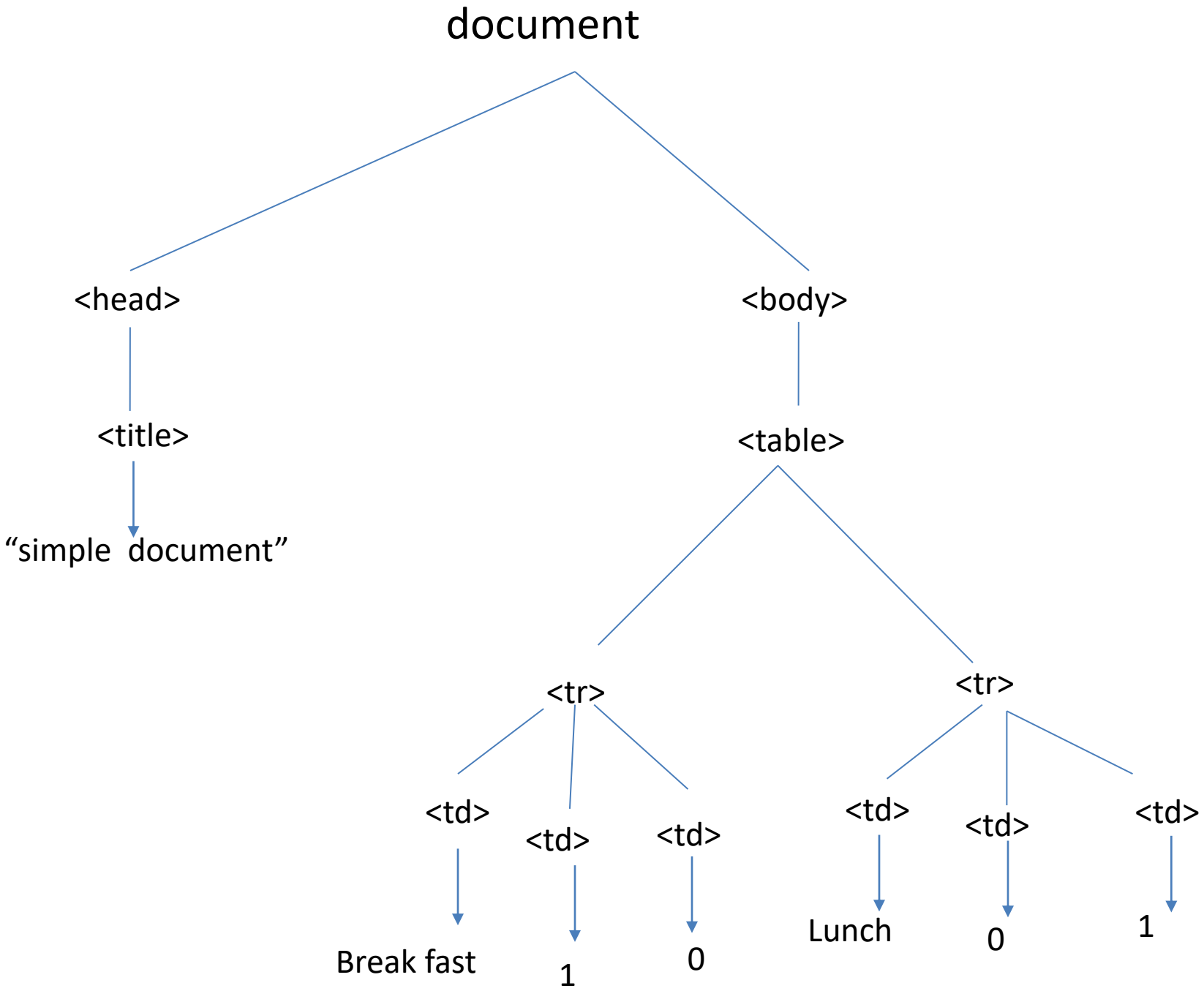
-DOM 1 (1998) – XHTML & XML

-DOM 2 (2000) -- Style sheet based object model

-DOM 3(Developing) – Content model

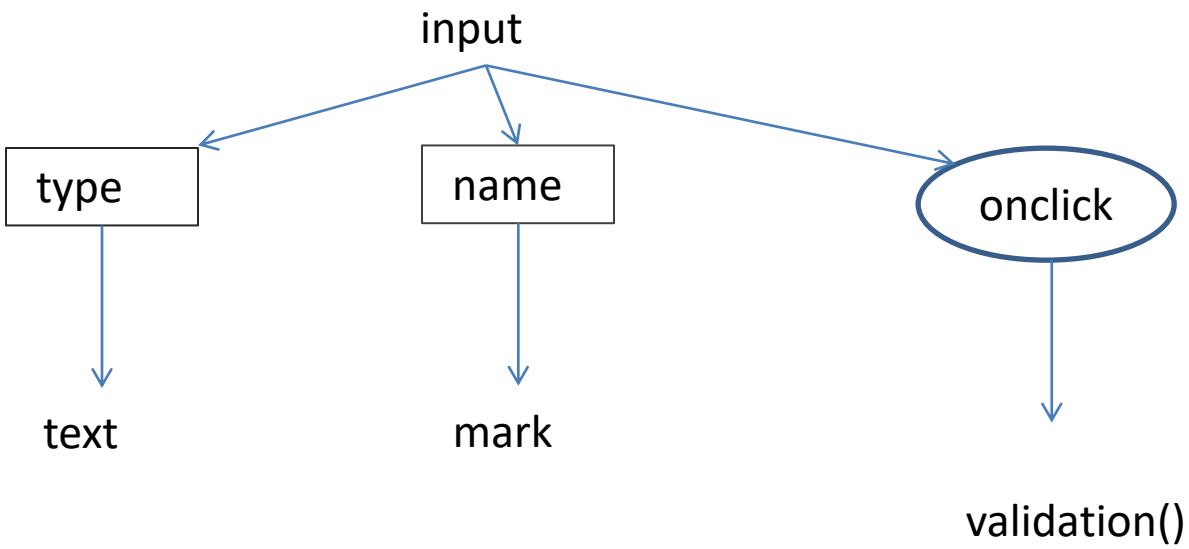
```
• <html>
      <head>
          <title>simple document </title>
      <head>
       <body>
           <table>
                   <tr>
                           <td>breakfast</td>
                           <td>1</td>
                           <td>0</td>
                   </tr>
                           <tr><td>lunch</td>
                           <td>0</td>
                           <td>1</td>
                   </tr>
          </table>

      </body>
</html>
```


document
<head>
<body>
<title>
<table>
“simple document”
<tr>
<tr>
<td>
<td>
<td>
<td>
<td>
<td>
Break fast
1
0
Lunch
0
1

- DOM node is an object represent xhtml element ,with data and operation.

E.g <input type="text" name="mark" onclick="validation();" />



- Dom object address is obtained in many ways

Method1.

```
<form   action=“”   method=“ ”>
        <input type=“ text” name=“turnon”>
</form>
```

```
var dom1= document . forms[0].elements[0] ;
                                # dom1 is object
```

- Manipulate data before going to server (form)

- Disadvantage : If new element is inserted

Method2.

```
<form action=“ ” method=“ ” name =“ myform”>
        <input type=“ text” name=“turnon”>
</form>
```

```
var dom2= document. myform.turnon;
                    #dom2 is object
```

- Disadvantage : XHTML 1.1 not support name attribute

- How to uniquely identify radio & checkbox button?

  - same name

Method3.

```
<form   action=“”   method=“ ”>
        <input type=“ text” name=“turnon”  Id=“turn”>
    </form>
var dom3= document. getElementById(“ok”);
                    #dom3 is an object
```

- Very element has id(unique)
- Get element from any deeply nesting (e.g mahatma ghandi)

Disadvantage: Not convenient to search a group of radio/ checkbox to determine which is checked

- Event model → DOM 0 ✔
  → DOM 2

- Event is a notification that something is happen(browser & user) like button click, window closing.

- Event handler - code executed in response to the appearance of event (similar to exception)

- Js support event driven approach

- Not sequential execution - code run while event occur

- Events are objects, name is case sensitive (lower case)

  e.g. click

- Events associated with xhtml element

  e.g. click event is generated while user clicking link or button

- Registration: process of connecting event handler to event

Registration by :

1. using tag attribute
2. assign handler address to object property

EVENT	TAG ATTRIBUTE
blur	onblur
change	onchange
click	onclick
dblclick	ondblclick
focus	onfocus
Keydown	onkeydown
keypress	onkeypress
load	onload
select	onselect
submit	onsubmit
unload	onunload
reset	onreset

- These events are associated with XHTML tag attributes, which can be used to connect the event to handler.

Event → Tag attribute → Event handler

e.g. <input type="text" name="tuenon" onblur="val();" />

- Same attribute can appear in different kinds of tag.

- Focus of an element is forced with focus().
- Only one element has focus at a time
- When focus moved , then blur occur

- onblur : <a> , <button>,< input>,<teaxtarea>,<select>
- onchange : < input>,<teaxtarea>,<select>
- onclick :< input>, <a>
- onload : <body>
- onunload : <body>
- onsubmit, onreset: <form>

- Event registration: 2 ways

Way1: Tag attribute

```
<input type="button" id=" 2a" onclick="alert('submit form data?')"/>
```

(or)

```
<input type="button" id=" 2a" onclick="val();"/>
```

Way2: Assignment to the associated property

- E.g document.getElementById("2a").onclick=val;

Example 1

- <html>

```
<html>

    <head>
        <title>First Example </title>
        <script type="text/javascript" src="load.js" ></script>
    </head>


    <body onload="welcome();">
    </body>

<html>

<!– index.xhtml-->
```

```
function  welcome(){
                alert("you are welcome");
                        }
```

// load.js

//don't use write() in event handler. why?

- We can manipulate style sheet using DOM & Event handling

Example 2

```
<html>
<head>
<script type="text/javascript" src="load2.js"></script>
</head>
<body>
< form action="" method ="post" name="myform">
<input type="text" name="usname" id="usname"/>
<input type="password" name="pwd" id="pwd"/>
<input type="button" value="Login" onclick="val();"/>
</form>
</body>
</html>
```

<!- - index.html- ->

```
function val(){
        var user= document.myform.usname;
        var pawd=documet.getElementById(“pwd”).value;
        if((user .value== null) ||(pawd==null))
            {
                alert(“Enter username/passward”);
                user.focus();
                return false;
            }
           else
                {
                   alert(“thank you”);
                   return true;
                }
```