Introduction to JavaScript

- Today's Websites need to go much beyond HTML
- Today's web site must be intelligent enough to accept users input and dynamically structure web page content, tailor made - to a user's requirements.
- This requires a web development environment that will allow the creation of **Interactive Web Pages**.
- Also it should provide the facility for **validating user input.**
  - Conditional Checking Construct – Case Checking Constructs
  - Well Controlled Loop Constructs.

Introduction to JavaScript

- The development environment should support the syntax for creation and handling the memory variables and objects.
- JavaScript is a scripting language (related to Web Site Development Environment) which supports above (But not limited) features to enhance the process of Web Development.
- JavaScript is typically embedded into HTML Program

HTML & JavaScript

- HTML, itself, allows a very low level of **Dynamicity** and **Interactivity**.
- Truly interactive pages can not be created using standard **HTML Tags** alone.
- Embedding JavaScript in HTML does this.
- For instance…… a Web Site can be created and hosted on a web server to take orders for product.
- At the same time, **Form Data Validation** must be done

- Validations Expected
  - Orders should be accepted only for products which are available
  - Any quantity ordered should not exceed the quantity currently available
  - The order date should be set to current date
  - The quantity required can not be left blank.
  - The place of delivery (address) can not be left blank.
- HTML alone can not do any of this.
- At the Max., HTML can provide an elegant interface for capturing Order Information.
- HTML do not provide any technique for **Validating User Input**
- **This makes it necessary to introduce client side script in HTML Program which extends basic functionality.**

HTML & JavaScript

- JavaScript is an O.B. Language that allows creation of Interactive Web Pages
- JavaScript allows user entries, which are loaded into HTML form, to be processed as required.
- This empowers a web site to return site information according to user's request.
- The perfect blend of **JavaScript** along with Inherent **HTML Capabilities** will offer more appreciable way of Web Page Development.

Strengths of JavaScript

- **An Interpreted Language**
  - Which requires no compilation steps.
  - The syntax is completely recognized by Browser directly.
- **Obvious compatibility with HTML**
  - HTML files containing JavaScript Instructions can easily be executed by Browser. No different platform is required (Editor or Compiler or IDE).
- **Minimal Syntax – Easy to Learn & Use – Rapid Development**
- **Minimum Storage Requirement at Web Server and Downloadable Time.**
- **Procedural Capabilities**
  - Conditional Checking & Branching – Looping
- **Supports Event Driven Programming efficiently**
  - Ability to recognize the Button Press

Writing JavaScript Into HTML

- JavaScript Syntax is embedded into HTML File.
- A Browser reads HTML files and interprets HTML Tags.
- Since entire JavaScript need to be included as an integral part of HTML document, when required, the browser needs to be informed that specific section of **HTML** Code is **JavaScript**.
- The browser will then use its built-in JavaScript Engine to interpret this code.
- For this purpose, <SCRIPT> ..... </SCRIPT> Tag is used

```
<SCRIPT type="text/ javascript ”>
    // JavaScript Code Snippet is written here
< /SCRIPT>
```

Basic Programming Techniques

- A JavaScript program can be placed anywhere within the HTML file
- Typically developers favor placing their programs between **<head>** tags in order to separate the programming code from the Web page content and layout.

Basic Programming Techniques

- JavaScript offers the very same programming capabilities found in most programming languages.
  - Creating variables, constants, programming constructs, user defined functions and so on.
- The <HEAD> … </HEAD> tag makes an ideal place to create JavaScript Variables
- This is because Head of HTML Document is always processed before the body.
- Placing JavaScript variables and UDFs in this section will cause them to be defined before being used.

- **Comments**
- The syntax for a single-line comment is
  - *// comment text*
- The syntax of a multi-line comment is
  - */* comment text covering several lines */*

Basic Programming Techniques

- **Writing Output to a Web Page**
- JavaScript provides two methods to write text to a Web page:
  - **document.write("text");**
  - **document.writeln("text");**
- The document.writeln() method differs from document.write() in that it attaches a carriage return to the end of each text string sent to the Web page.

JavaScript Variables

- **Declaring (Creating) JavaScript Variables**
- Creating a variable in JavaScript is most often referred to as "declaring" a variable.
- You declare JavaScript variables with the **var** keyword
- Variable names are **Case Sensitive**.
- Variables can be of type
  - Number
  - Boolean
  - String

Type Casting & Creating Variables

- In JavaScript, variables are loosely cast.
  - The type of variable **is implicitly defined** based on values that are assigned to it from time to time.
- Creating variables
- var <variable name> = value;
- Var firstname;
- Var lastname = “Mathew”
- Var phone = 24225575;

First JavaScript Code


```
<HEAD>
        <TITLE>
                Variables in a Script
        </TITLE>

        <script type="text/ javascript">

                var name = prompt("Enter your Name...");
        </script>
    </HEAD>

    <BODY>
        <script type="text/ javascript">

                document.write("<H2> Hello  "  + name + " </H2>");
        </script>

    </BODY>
```

JavaScript Arrays

- Arrays are objects that are capable of storing sequence of values with indexed locations
- Declaration of Array
- arrayName = new Array(length);
- arrayName = new Array( );
- Cust_Orders = new Array();
- Cust_Orders[50] = “Pepsi”
- When JavaScript accesses this Cust_Orders[50], it will extend the size of that array to 51.

Operators in JavaScript

- JavaScript supports the operator types as that of other languages
  - Arithmetic +, -, *, /, %, ++, --
  - Logical &&, ||, !
  - Relational ==, !=, <, >, <=, >=,
  - String Operators + as String Concatenation
  - Ternary Operator condition ? Value1 : Value2

Control Structures in JavaScript

- JavaScript provides complete range of basic programming constructs
- If…….Else Construct

```
var day = prompt(" What's today…?");
if( day == "Saturday")
{
  document.write("This is WEEKEND");
}
else{
 document.write("This is WEEKDAY"); }
```

Iterative Constructs


```
For(var num = 10; num>=1; num--)
    {
        document.writeln( num);
    }

var num = 20
while( num<=30)
{
    document.writeln( num);
    num++;
}
```

UDFs in JavaScript


```
function function_name(param1, param2,-------)
{

}
```

Place of Declaration : Preferably within <Head> Section

Declaration and Definition both will stand together.

Dialog Boxes in JavaScript

- The Alert Dialog Box
  - The simplest way to direct the text to browser window.
  - alert() method takes the text as argument and displays the Alert Dialog Box
  - The program execution will not continue until the OK button pressed on dialog box
  - alert("Click OK to Continue");

Dialog Boxes in JavaScript

- The Prompt Dialog Box
  - Alert Dialog Box simply displays the information and does not allow any other interaction rather than OK
  - prompt() method instantiates the Prompt Dialog Box
  - This displays the specified message
  - In addition, it also provides a single data entry field which accepts the user input

JavaScript Code using Function and Events


```
<HTML>
    <HEAD> <TITLE>  UDFs called on Events </TITLE>
        <script type="text/ javascript">
            var name = "";
            function hello()
             {
                    name = prompt("Enter Your Name","Name");
                    alert('Greetings  ' + name + ' !!! ' + '  Welcome to My Page  ');
              }
        </script>
        </HEAD>
        <BODY onLoad = “hello( );” onUnload = “goodbye();”>
        </BODY>     </HTML>
```

Named JavaScript Event Handlers

- onChange
  - Text Field, Text area is modified
- onClick
  - A link or document is clicked
- onDragDrop
  - A dragged object is dropped in a window or frame
- onError
  - An error occures during loading of image, window or frame
- onKeyPress
  - User presses a key

Named JavaScript Event Handlers

- onLoad
  - When web page is loaded
- onMouseOver
  - The mouse is moved over a link
- onSubmit
  - The user presses a form's submit button
- onUnload
  - The user exits a document

Some more handlings for JavaScript

- ** Element.**
  - div is a block level container for other elements which divides the HTML Document into presentable sections.
  - Prominent attribute of this element is ID with which we can refer the actual presentable division
  - Which otherwise is not done when we use the JavaScript's output function document.write().
  - When document.write() displays the contents, it overwrites the earlier presented material.

Element

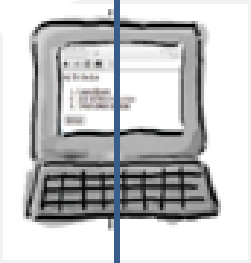

```
<HEAD>
    <SCRIPT LANGUAGE="JavaScript">
        function custInfo()
        {
          outputText=prompt("Enter the Name of Customer  ");
          document.write(outputText);      messageDIV.innerHTML=outputText;
        }
    </SCRIPT>
</HEAD>
<BODY>
    <INPUT TYPE="BUTTON" VALUE="Get the Name of Customer"
        OnClick="custInfo()">
    <BR><BR>
                                <DIV ID="messageDIV">        </DIV>
</BODY>
```

Element

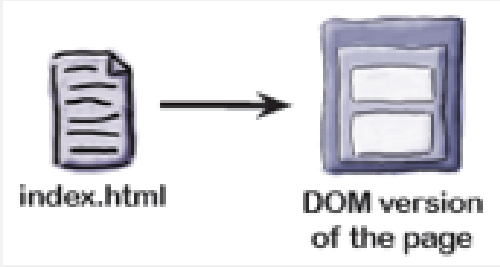
- ** Element.**
  - When works, it first divides the page into presentable divisions.
  - So earlier contents on screen are kept intact and new contents are displayed into different division
  - For this, we can use <DIV> Element's property innerHTML
  - This property actually holds the text content of <DIV> Element.
  - messageDIV.innerHTML = outputText
  - Here messageDIV is the ID for our element

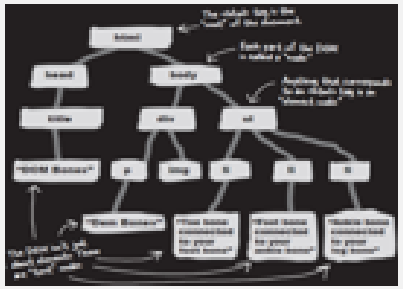
Look into the Browser

1. The Browser loads the HTML file, it gets from server



2. Browser builds a render able document using Document Object Model





3. Browser renders DOM Objects in browser window according to the syntax and symatics

These are the stuffs happening inside the browser

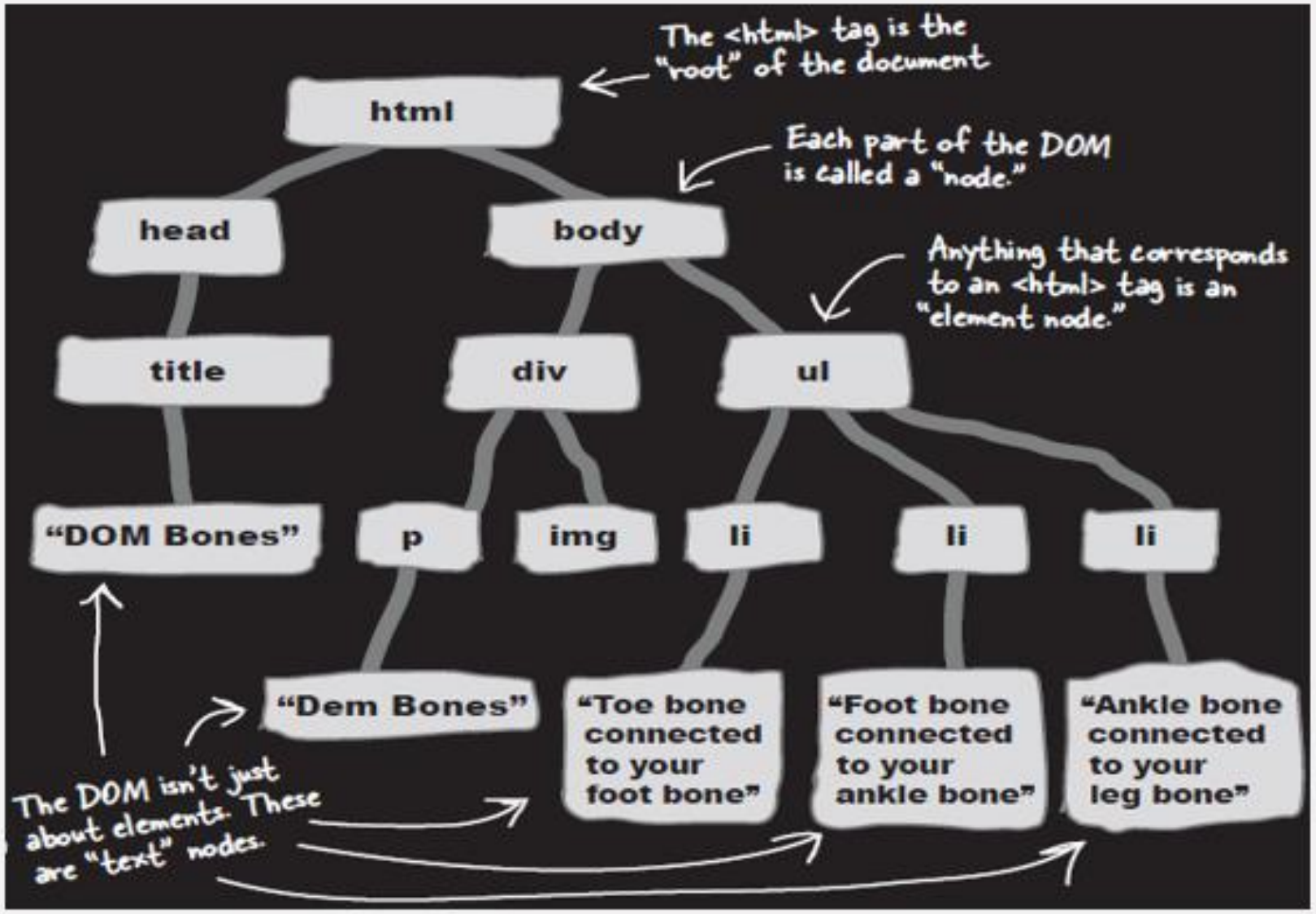
4. The JavaScript Interpreter references the DOM objects collectively and the JavaScript or jQuery functions operate on the objects individually / as a group.

YASH Technologies
More than what you think.

According to Document Object Model

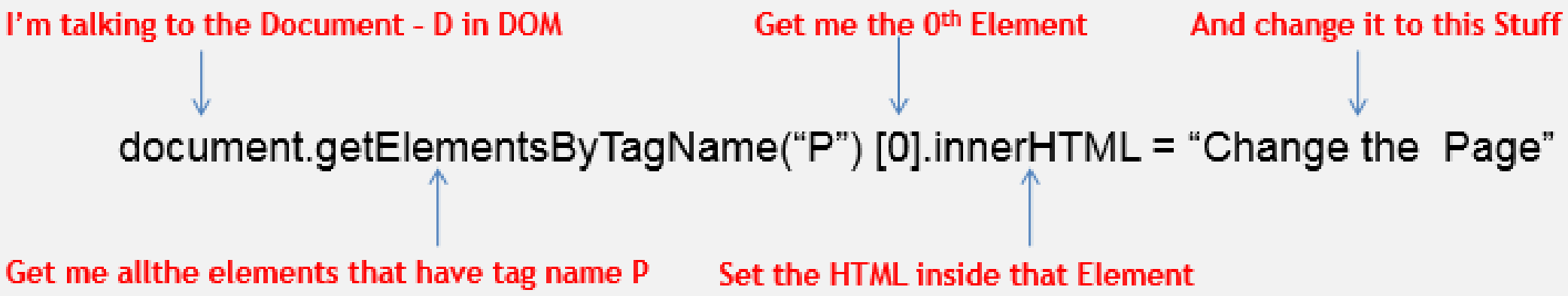
- The top most object in DOM is the **Browser** itself.
- Next level in the model is browser's **window**.
- Then comes the **document** displayed in window.
- The DOM hierarchy continues downward to encompass Form and its individual elements like Text Boxes, Buttons & Boxes and so on.
- Browser
  - Anchor
  - Link
  - Form
    - Textbox – Radio Button – checkbox – push button

The hidden structure of a web page – with DOM



The way we can handle complexities of DOM

- For instance, let's say we want to change the HTML inside of the **paragraph** element on our page.
- The JavaScript way :



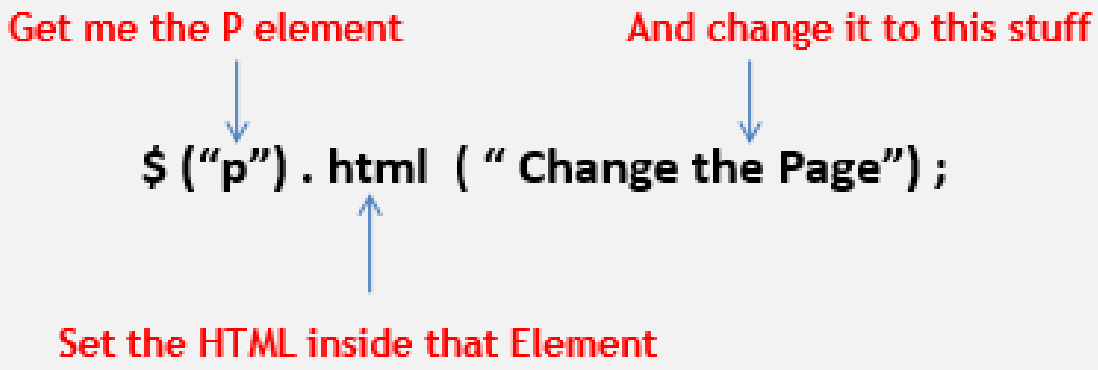
The way we can handle complexities of DOM

- Now suppose we want to change the HTML inside all 5 paragraph elements on our page

```
for ( i = 0 ; I <= 4 ; I ++ )
  {
     document.getElementsByTagName("P") [i].innerHTML = "Change the  Page" ;

  }
```

- The jQuery way :



Javascript methods(Array)

toString()→array to String(Comma Separated values)
Join() →you can specify the separator
Pop()
Push()
Shift()→pop →removes the first element and rest elements to lower index

```
const days=[“mon”,”Tue”,”Wed”];
let d=days.shift()
```

Mon
Days→tue,wed
Unshift()→add an element in the beginning of the array

```
Days.unshift(“sun”);
```

Concat()
Splice()→add new elements to an array
Slice()→slice out a element /divide the an array

const days=[“mon”,”Tue”,”Wed”,”Thursday”,”Fri”];

Days.splice(1,0,”Sat”);
1→ represents the position where new element should be added
0→0 elements should be removed
“sat”→new element to be added
Splice→returns an array with deleted elements

Days.Splice(0,1)→0→added position
1→removed(how many elements)

Slice()
Let d=Days.slice(2)
(d)
Slice(1,4)→1 start and 4 upto(but not including)

Javascript methods(Array)

const days=[“mon”,”Tue”,”Wed”,”Thursday”,”Fri”];
Days.sort()
Days.reverse()
String
Slice(start,end)
Substring(start,end)
Substr(start,length)
Replace()
Let a=“Yash Technologies”
Let b=a.replace(“Technologies”,”Tech”);
toUpperCase()
toLowerCase()
b=a.toUpperCase()
Concat()
Trim()→Removes whitespace from both sides
charAt()
Split()

Javascript methods(Array)

Let a="Welcome to Yash";
a.indexOf("to");→first occurrence
a.lastIndexOf("to")→last occurrence
Return -1; if the string is not found