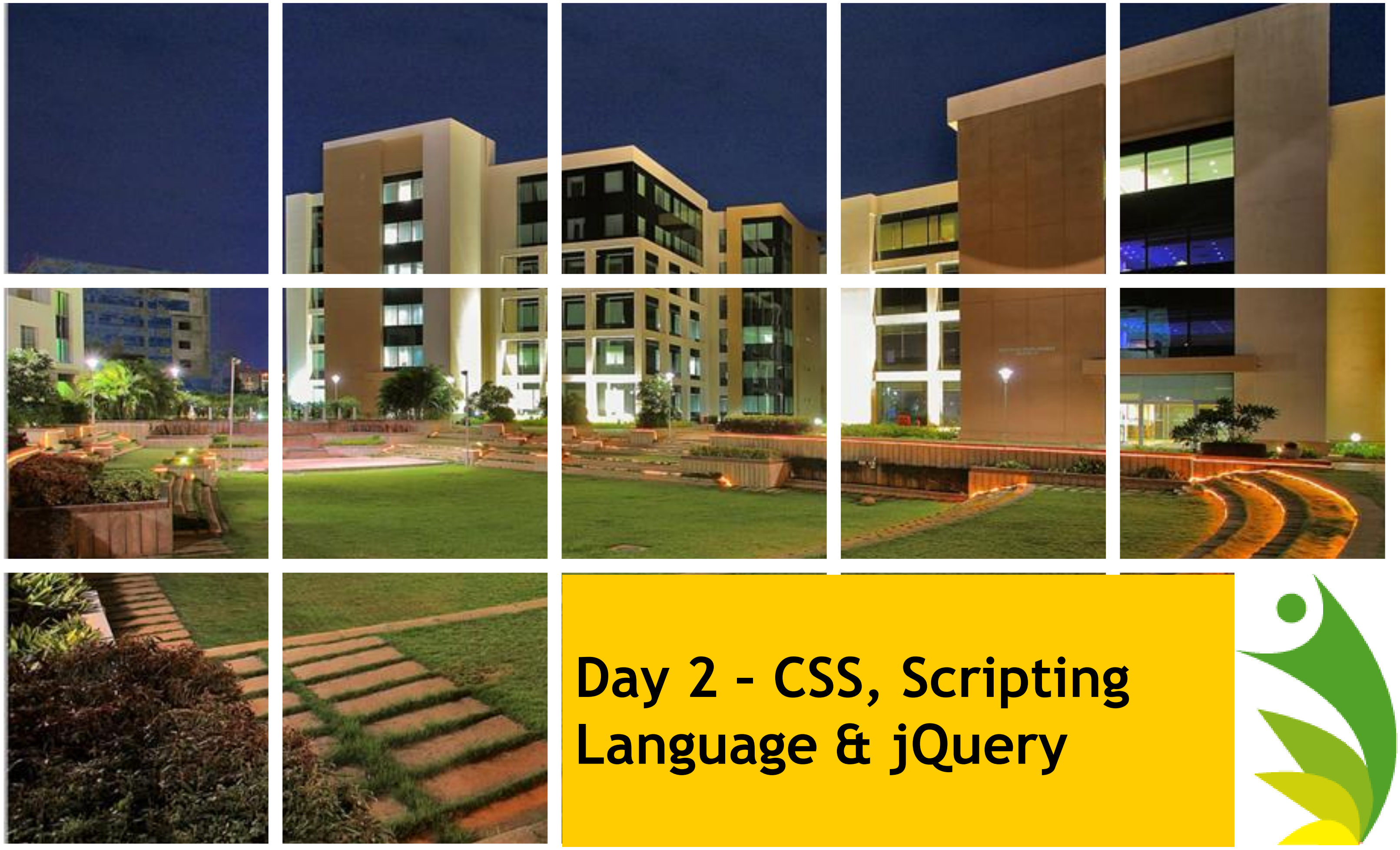


Day 2 - CSS, Scripting Language & jQuery

July 20th 2012

CSS

L&T Infotech

Content – cascading style sheets

- Applying CSS - The different ways you can apply CSS to HTML.
- Selectors, Properties, and Values - The bits that make up CSS.
- Colors - How to use color.
- Text - How to manipulate the size and shape of text.
- Margins and Padding - How to space things out.
- Borders - Borders.

Applying CSS

- 3 ways to apply CSS to HTML
  - In-line
    - With HTML tags using style attribute

      ```
      <p style="color: red">text</p>
      ```
  - Internal

    ```
    <html> <head> <title>CSS Example</title>
       <style type="text/css">
                      p { color: red; } a { color: blue; }
       </style> ...
    ```
  - External

    ```
       <html> <head> <title>CSS Example</title>
           <link rel="stylesheet" type="text/css" href="web.css" />
           ...
    ```

Selector, Property, and Values

- For each selector there are **'properties'** inside **curly brackets**, which simply take the form of words such as **color**, **font-weight** or **background-color.**
  - body {
      font-size: 0.8em;
      color: navy;
    }

- **Length and Percentages**
  - **em** (such as font-size: 2em) is the unit for the **calculated size of a font**. So "2em", for example, is two times the current font size.
  - **px** (such as font-size: 12px) is the unit for **pixels.**
  - **pt** (such as font-size: 12pt) is the unit for **points**.
  - % (such as font-size: 80%) is the unit for **percentages**.
  - Other units include **cm** (centimetres), **mm** (millimetres) and **in** (inches).

color and background color

- A blue background and yellow text could look like this:
  - h1 {
      color: yellow;
      background-color: blue;
    }
- Usage
  - **red**
    is the same as **rgb(255,0,0)** which is the same as **rgb(100%,0%,0%)**
    which is the same as **#ff0000** which is the same as **#f00**
- There are 17 valid predefined color names. They are **aqua**, **black**, **blue**, **fuchsia**, **gray**, **green**, **lime**, **maroon**, **navy**, **olive**, **orange**, **purple**, **red**, **silver**, **teal**, **white**, and **yellow**.
- **transparent** is also a valid value.

Text

- font-family
  - This is the font itself, such as Times New Roman, Arial, or Verdana.
  - a select few '**safe**' fonts (the most commonly used are arial, verdana and “times new roman”)
  - You may specify alternate fonts separated by comma.
- font-size
  - Size of the font.
- font-weight
  - This states whether the text is **bold** or not. In practice this usually only works as **font-weight: bold** or **font-weight: normal**.
    In theory it can also be **bolder**, **lighter**, **100**, **.. 900**
- font-style
  - This states whether the text is **italic** or not. It can be **font-style: italic** or **font-style: normal**.

Text

- text-decoration
  - This states whether the text is underlined or not. This can be:
    text-decoration: overline, which places a line above the text.
    text-decoration: line-through, strike-through
    text-decoration: underline **should only be used for links**
    - This property is usually used to decorate links, such as specifying no underline with **text-decoration: none**.
- text-transform
  - This will change the case of the text.
    text-transform: capitalize turns the first letter of every word into uppercase.
    text-transform: uppercase turns everything into uppercase.
    text-transform: lowercase turns everything into lowercase.
    text-transform: none

Text

- Text spacing
  - The **letter-spacing** and **word-spacing** properties are for spacing between letters or words. The value can be a length or normal.
  - The **line-height** property sets the height of the lines in an element, such as a paragraph, without adjusting the size of the font. It can be a number (which specifies a multiple of the font size, so '2' will be two times the font size, for example), a length, a percentage or normal.
  - The **text-align** property will align the text inside an element to **left, right, center** or **justify.**
  - The **text-indent** property will **indent** the first line of a paragraph.

  -
    ```
    p {
      letter-spacing: 0.5em;
      word-spacing: 2em;
      line-height: 1.5;
      text-align: center;
    }
    ```

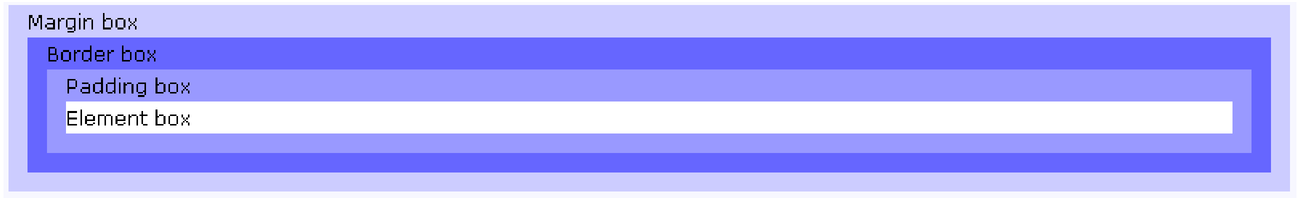
Text

```
body {
        font-family: arial, helvetica, sans-serif;
        font-size: 0.8em;
}

h1 {
        font-size: 2em;
}

h2 {
        font-size: 1.5em;
}

a {
        text-decoration: none;
}

strong {
        font-style: italic;
        text-transform: uppercase;
}
```

L&T Infotech

Margin and Padding - The Box model !

- margin is space **outside** the element, whereas padding is the space **inside** the element.



- **h2** {
  font-size: 1.5em;
  background-color: #ccc;
  margin: 1em;
  padding: 3em;

  }
- The four sides of an element can also be set individually. **margin-top**, **margin-right**, **margin-bottom**, **margin-left**, **padding-top**, **padding-right**, **padding-bottom** and **padding-left**

Border

- To make a border around an element, **border-style** is used. The values can be **solid**, **dotted**, **dashed**, **double**, **groove**, **ridge**, **inset** and **outset**.
- **border-width** sets the **width** of the border, which is usually in pixels. There are also properties for **border-top-width**, **border-right-width**, **border-bottom-width** and **border-left-width**.
- Finally, **border-color** sets the color.

-
  ```
  h2 {
          border-style: dashed;
          border-width: 3px;
          border-left-width: 10px;
          border-right-width: 10px;
          border-color: red;
  }
  ```

  This will make a red dashed border around all **h2** element that is 3 pixels wide on the top and bottom and 10 pixels wide on the left and right (these having over-ridden the 3 pixel wide width of the entire border).

INTERMEDIATE CSS

Contents

- Class and ID Selectors - Make your own selectors
- Grouping and Nesting - Properties assigned to multiple selectors or selectors within selectors.
- Shorthand Properties - Various properties, such as borders and margins that amalgamate other properties into one.

Class and ID Selectors

- You can also define your own selectors in the form of **Class** and **ID** selectors.
- The benefit of this is that you can have the same HTML element, but present it differently depending on its class or ID.
- A class selector is a name preceded by a **full stop** (**.**) and an ID selector is a name preceded by a **hash character** (**#**).
- **CSS:**

```
#top {
        background-color: #ccc;
        padding: 1em
}
.intro {
        color: red;
        font-weight: bold;
}
```

- HTML:

```
<div id="top">
 <h1>Chocolate curry</h1>
 <p class="intro">This is my recipe for
making curry purely with chocolate
</p>
  <p class="intro">Mm mm mm</p>
</div>
```

- The difference between an ID and a class is that an ID can be used to identify one element, whereas a class can be used to identify more than one.
  You can also apply a selector to a specific HTML element, so p.jam { whatever } will only be applied to paragraph elements that have the class 'jam'.

L&T Infotech

Grouping and Nesting

- Grouping
- You can give the same properties to a number of selectors without having to repeat them by separating the selectors by **commas**.
- For example, if you have something like:

```
h2 { color: red; }
.thisOtherClass { color: red; }
.yetAnotherClass { color: red; }
```

You could make it:

```
h2, .thisOtherClass, .yetAnotherClass { color: red; }
```

Grouping and Nesting

- **Nesting**
- For example:
  #top { background-color: #ccc; padding: 1em }
  #top h1 { color: #ff0; }
  #top p { color: red; font-weight: bold; }

```
<div id="top">
        <h1>Chocolate curry</h1>
        <p>This is my recipe for making curry purely with chocolate</p>
        <p>Mmm mm mmmmm</p>
</div>
```

Shorthand Properties

- **margin**, **padding** and **border-width** allow you to amalgamate **margin-top-width**, **margin-right-width**, **margin-bottom-width** etc. in the form of property: **top right bottom left**;
- So:

```
p {
   border-top-width: 1px;
   border-right-width: 5px;
   border-bottom-width: 10px;
   border-left-width: 20px;
}
```

  Can be summed up as:

```
p {          border-width: 1px 5px 10px 20px;          }
```

- **Another example**
  - **p** {

```
  font: 1em/1.5 "Times New Roman", times, serif;
  padding: 3em 1em;
  border: 1px black solid;
  border-width: 1px 5px 5px 1px;
  border-color: red green blue yellow;
  margin: 1em 5em;
}
```

JAVASCRIPT

What is JavaScript?

- Not related to Java
- Designed to add interactivity to web pages
  - It can put text dynamically in HTML page
  - It can react to events
  - It is a case-sensitive language; HTML is not
- Lightweight interpreted programming language
- Can be used to validate user data
- Can be used to detect visitor’s browser
- Can be used to create cookies
- It’s official name is ECMAScript

Client and Server side scripting

- Client-side scripting is about "programming" the behavior of the browser. Generally done using JavaScript.
- Server-side scripting is about "programming" the behavior of the server. Generally done using Perl, PHP or server side VBscript.

- **What can Server Scripts Do?**
- Dynamically edit, change or add any content to a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- Provide security since your server code cannot be viewed from a browser

Put a script into HTML page

```
<html>
    <head>
        <title>Learning JavaScript</title>
        <script type="text/javascript">
                ….
        </script>
    </head>
    <body>
        <p>Hello World!
        <script type="text/javascript">
                ….
        </script>
    </body>
</html>
```

- **Inline Java script**
  ```
  <script type='text/javascript'>
        // Your script goes here.
  </script>
  ```
- **External Java script**
  ```
  <script type='text/javascript' src='common.js'> </script>
  ```

Handling no support

```
<html>
   <body>
        <script type="text/javascript">
        <!--
                document.write("Hello World!");
        //-->
        </script>
   </body>
</html>
```

Output

- Write

```
<html>  <head> </head>              <body>
 <script type='text/javascript'> document.write('Hello World!');</script>
</body> </html>
```

- Alert

```
<html> <head> </head> <body>
      <script type='text/javascript'> alert('Hello World!');  </script>
</body> </html>
```

- getElementById

```
<html>  <head> </head>
      <body>
                  <div id='feedback'></div>
                  <script type='text/javascript'>
                  document.getElementById('feedback').innerHTML='Hello World!';
                  </script>
      </body>
</html>

// <script type='text/javascript'>
//    document.getElementById('feedback').innerHTML='<P><font color=red>Hello World!</font>';
//</script>
```

L&T Infotech

Input

- Events

```
<html>
    <head>
   </head>
      <body>
        <div id='feedback' onClick='goodbye()'>Users without Javascript see          this.</div>

              <script type='text/javascript'>
    document.getElementById('feedback').innerHTML='Hello World!';

                function goodbye() {
                 document.getElementById('feedback').innerHTML='Goodbye World!';
                }
              </script>
         </body>
  </html>
```

Input

- User Input

```
<HTML>
    <HEAD> </HEAD>
    <BODY>
      <input id='userInput' size=60> <button
           onClick='userSubmit()'>Submit</button><BR>
        <P><div id='result'></div>

        <script type='text/javascript'>
               function userSubmit() {
                 var UI=document.getElementById('userInput').value;
                 document.getElementById('result').innerHTML='You typed: '+UI;
                }
             </script>
           </BODY>
    </HTML>
```

Exercise

- Code for alert
- Code for confirm
- Code for prompt

L&T Infotech

JavaScript is Event Driven Language

Event name	Description
▪ onAbort	An image failed to load.
▪ onBeforeUnload	The user is navigating away from a page.
▪ onBlur	A form field lost the focus (User moved to another field)
▪ onChange	The contents of a field has changed.
▪ onClick	User clicked on this item.
▪ onDblClick	User double-clicked on this item.
▪ onError	An error occurred while loading an image.
▪ onFocus	User just moved into this form element.
▪ onKeyDown	A key was pressed.
▪ onKeyPress	A key was pressed OR released.
▪ onKeyUp	A key was released.
▪ onLoad	This object (iframe, image, script) finished loading.
▪ onMouseDown	A mouse button was pressed.
▪ onMouseMove	The mouse moved.
▪ onMouseOut	A mouse moved off of this element.
▪ onMouseOver	The mouse moved over this element.
▪ onMouseUp	The mouse button was released.
▪ onReset	A form reset button was pressed.
▪ onResize	The window or frame was resized.
▪ onSelect	Text has been selected.
▪ onSubmit	A form's Submit button has been pressed.
▪ onUnload	The user is navigating away from a page.

JavaScript Comments

- //

```
var x=5; // Everything from the // to end of line is ignored(*)
var doublePrice=123.45; // 2 times the price of a price.
```

- /*   … */

```
function whirlymajig(jabberwocky) {
    /* Here we take the jabberwocky and insert it in the gire-gimble,
    taking great care to observe the ipsum lorum! For bor-rath-outgrabe!
    We really should patent this! */

    return (jabberwocky*2);
}
```

Variables

```
var thisIsAString = 'This is a string';          var alsoAString = '25';          var isANumber = 25;
var isEqual = (alsoAString==isANumber); // This is true, they are both 25.
var isEqual = (alsoAString===isANumber); // False one is a number, the other a string.
var concat=alsoAString + isANumber; // concat is now 2525
var addition=isANumber + isANumber; // addition is now 50
var alsoANumber=3.05; // is equal to 3.05 (usually).
var floatError=0.06+0.01; // is equal to 0.06999999999999999
var anExponent=1.23e+3; // is equal to 1230
var hexadecimal = 0xff; // is equal to 255.
var octal = 0377; // is equal to 255.
var isTrue = true; // This is a boolean                    var isFalse= false; // This is a boolean
var isArray = [0, 'one', 2, 3, '4', 5]; // This is an array.
var four = isArray[4]; // assign a single array element to a variable. // in this case four = '4'
var isObject = { 'color': 'blue',                    // This is a Javascript object
      'dog': 'bark',
      'array': [0,1,2,3,4,5],
      'myfunc': function () { alert('do something!'); }
}
var dog = isObject.dog; // dog now stores the string 'bark';
isObject.myfunc(); // creates an alert box with the value "do something!"
var someFunction = function() {
      return "I am a function!";
}
var alsoAFunction = someFunction; //No () so alsoAFunction becomes a function
var result = alsoAFunction(); // alsoAFunction is executed here because ()
// executes the function so result stores the return value of the function which is "I am a function!"
```

Variable Scope

```
var global = 'this is global';
function scopeFunction() {
      alsoGlobal = 'This is also global!';
      var notGlobal = 'This is private to scopeFunction!';
      function subFunction() {
                alert(notGlobal); // We can still access notGlobal in this child function.
                stillGlobal = 'No var keyword so this is global!';
                var isPrivate = 'This is private to subFunction!';
      }
      alert(stillGlobal); // This is an error since we haven't executed subfunction
      subFunction(); // execute subfunction
      alert(stillGlobal); // This will output 'No var keyword so this is global!'
      alert(isPrivate); // This generate an error since isPrivate is private to subfunction().
      alert(global); // outputs: 'this is global'
}
alert(global); // outputs: 'this is global'
alert(alsoGlobal); // generates an error since we haven't run scopeFunction yet.
scopeFunction();
alert(alsoGlobal); // outputs: 'This is also global!';
alert(notGlobal); // generates an error.
```

L&T Infotech

Special Keywords

- NaN – Not a Number

  It is never equal to itself, so you can not check to see if 3/’dog’ == 'NaN'. You must use the construct isNaN(3/dog).

  Since NaN is never equal to itself you can use this simple trick as well:

  ```
  if (result != result) { alert('Not a Number!'); }
  ```

- null – means empty - when used in boolean operation it evaluates to false.
- infinity - is returned when an arithmetic operation overflows Javascript's precision which is in the order of 300 digits.
- undefined - Variable hasn’t been defined or assigned yet

```
function doAlert(sayThis) {
 if (sayThis===undefined) { //Check to see if sayThis was passed.
    sayThis='default value'; //It wasn't so give it a default value
   } // End check
   alert(sayThis); // Toss up the alert.
}
```

Arithmetic Operators

Operator	Description
+	Addition (also string concatenation)
-	Subtraction (also unary minus)
*	Multiplication
/	Division
%	Remainder of division (or modulus)
++	pre or post increment
--	pre or post decrement

L&T Infotech

Logical And Comparison Operators

Operator	Description
=	Assignment x=5; // assigns 5 to x
==	Equality, is x==5?
===	Identity, is x 5 and a number as well? (value and type)
!=	Not equal, is x unequal to 5?
!==	Not identical, is x unequal to the Number 5?
<	Less than. is x less than 5?
<=	Less than or equal. is x less than or equal to 5?
>	Greater than. is x greater than 5?
>=	Greater than or qual. is x greater than or equal to 5?
!	Not, if not(false) is true.
||	OR, is (x==5) OR (y==5)
&&	And, is (x==5) AND (y==5)

Insert Special Characters

- var txt=“Top team so far is “Mumbai Indians” in IPL.";
- var txt=“Top team so far is \“Mumbai Indians\” in IPL.";

- Other special characters

Code	Output
\’	Single quote
\”	Double quote
\&	Ampersand
\\	Backslash
\n	new line
\r	carriage return
\t	Tab
\b	Backspace
\f	Form feed

Conditionals - If

The *if statement lets you execute a block of code if some test is passed.*

```
var x=5;
if (x==5) {
        alert('x is equal to 5!');
}
```

You can also use an *else clause to execute code if the test fails.*

```
var x=5;
if (x==5) {
        alert('x is equal to 5!');
} else {
        alert('x is not equal to 5!');
}
```

An *elseif statement also exists which allows for better formatting of long conditional tests.*

```
var x=5;
if (x==1) {
        alert('x is equal to 1!');
} else if (x==2) {
        alert('x is equal to 2!');
} else if (x==5) {
        alert('x is equal to 5!');
} else {
        alert('x isn't 1, 2 or 5!');
}
```

Conditionals - switch

```
var x=5;
switch (x-2) {
    case 1: alert('x is equal to 1!'; break;
    case 2: alert('x is equal to 2!'; break;
    case 5: alert('x is equal to 5!'; break;
    default: alert("x isn't 1, 2 or 5!");
}
```

Allowing evaluations on both the switch and the case.

```
var x=5;
switch (true) {
    case (x==1): alert('x is equal to 1!'; break;
    case (x==2): alert('x is equal to 2!'; break;
    case (x==5): alert('x is equal to 5!'; break;
    default: alert("x isn't 1, 2 or 5!");
}
```

L&T Infotech

Conditionals – Shorthand Assignment

```
function doAddition(firstVar, secondVar)
{
    var first = firstVar || 5;
    var second= secondVar || 10;
    return first+second;
}
doAddition(12);
```

Result:

```
    doAddition()          →        15
    doAddition(7)         →        17
    doAddition(2, 3)      →        5
```

In psuedo code...

```
var someVariable = (assign if this is truthy) || (assign this if first test evaluates false)
```

Conditionals – Ternary Operators

```
var userName = 'Bob';

var hello = (userName=='Bob') ? 'Hello Bob!' : 'Hello Not Bob!';
```

Loops – For

```
for (var i=0; (i<5); i++)
{
    document.writeln('I is equal to '+i+'<br>');
}


// outputs:
    // I is equal to 0
    // I is equal to 1
    // I is equal to 2
    // I is equal to 3
    // I is equal to 4
```

Loops – for/in

Consider the following object…

```
var myObject = { 'animal' : 'dog',
                 'growls' : true,
                 'hasFleas': true,
                 'loyal' : true
}
```

We can loop through these values with the following construct.

```
var myObject = { 'animal' : 'dog',
                 'growls' : true,
                 'hasFleas': true,
                 'loyal' : true }
    for (var property in myObject) {
        document.writeln(property + ' contains ' +
    myObject[property]+'<br>');
    }
// Outputs:
// animal contains dog
// growls contains true
// hasFleas contains true
// loyal contains true
```

Loops – while / do-while

```
var x = 1;
while (x<5)
{
    x = x +1;
}
var x = 1;
while (true)
{
    x = x + 1;
    if (x>=5) {
            break;
    }
}
```

```
var x=1;
do
{
    x = x + 1;
} while (x < 5);
```

Putting it all together

```
<html>
	<head>
		<title>My First Javascript</title>
	</head>
	<body>
		Hello World!
		<p>Say what? <input id="sayThis" size=40>
		<P>How many times? <select id='howMany'>
				<option value=1>1</option>
				<option value=5 selected>5</option>
				<option value=10>10</option>
				<option value=20>20</option>
		</select>
		<p><button onClick='doLoop()'>Do It!</button>
		<p><div id="results"></div>
		<script type='text/html'>
		function doLoop() {
				var sayWhat = document.getElementById('sayThis').value;
				var maxLoop = document.getElementById('howMany').value;
				var str = ''; // where we'll store our output temporarily.
				for (var i=1; (i<=maxLoop); i++) {
						str=str+i+':'+sayWhat+'<br>';
				}
				document.getElementById("results").innerHTML=str;
		}
		</script>
	</body>
</html>
```

Functional JavaScript

Sample Function

```
function hello(who)
{
   alert('Hello '+who); // outputs "Hello world"
}

hello('world'); //This is function call
```

Note: In ‘JavaScript’ function call can happen from same script block or from another script block!

Function as variable

```
var hello = function (who)
{
   alert('Hello '+who); // outputs "Hello world"
}
hello('world');
```

- Function follow the same rules and abilities as any other variable in JavaScript.
- It will have the same scope and it can be passed to other functions just like any other variable.

Calling Functions

```
var test = function ()
{
        return "This is a String";
}

var testCopy = test; //testCopy is a pointer to the test function()
window.alert('testCopy = ' + testCopy );

var testStr = test(); //testStr contains "This is a string“
alert('testStr = ' + testStr );

var testing = testCopy(); //testing contains "This is a string“
alert('testing = ' + testing );
```

Understanding Function Pointers

```
var originalFunction = function () {
    alert('hello world');
}
var copiedFunction = originalFunction;
var originalFunction = function () {
    alert('goodbye world');
}
copiedFunction(); // outputs “Hello World”.
```

Anonymous Functions

- document.onmousedown = function() { alert("You pressed the mouse button"); }

Understanding Function Scope

```
var globalFunction = function () { // global, can be accessed anywhere.
      alert('hello world');
}

var containerFunction = function() { // global, can be accessed anywhere.
      // Test#1
      // var subFunction = function() { // private--global to containerFunction // and its children only.
      // Test#2
      subFunction = function() { // Now it is global.
                alert("I'm Private");
                globalFunction(); //We can access global Function here 2 levels deep.
      }
      globalFunction(); // We can access global Function here 1 level deep.
      subFunction() // We can access subFunction inside containerFunction.
}

containerFunction();
// Test#1   subFunction(); // This produces an error because
subFunction() only exists
                // inside containerFunction
//Test#2
subFunction(); // executes because there is no var in its definition
```

Understanding Function Argument

```
var myFunc = function(a,b,c) { }
```

You can still call the function the following ways…

```
myFunc(1);
myFunc(1,2);
myFunc(1,2,3);
myFunc(1,2,3,4,5,6,7,8,9,10);
```

If myFunc(1) is called then b and c will be of type undefined and you can look for this in several ways.

```
var myFunc = function(a,b,c) {
        if (!a) {a=1};
        b = b || 2;
        if (c==undefined) {c=3;}
                document.writeln(a+'<br>'+b+'<br>'+c+'<BR>');
}
myFunc(1);
```

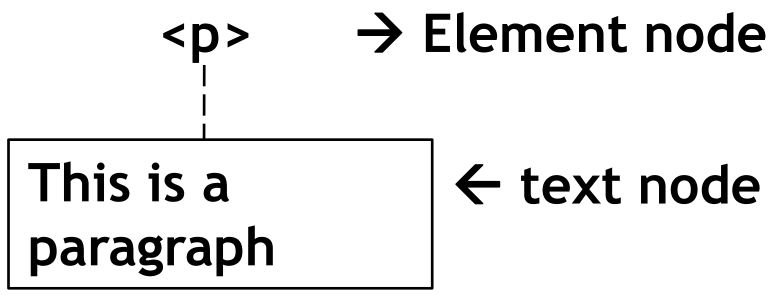
HTML DOM

What is DOM?

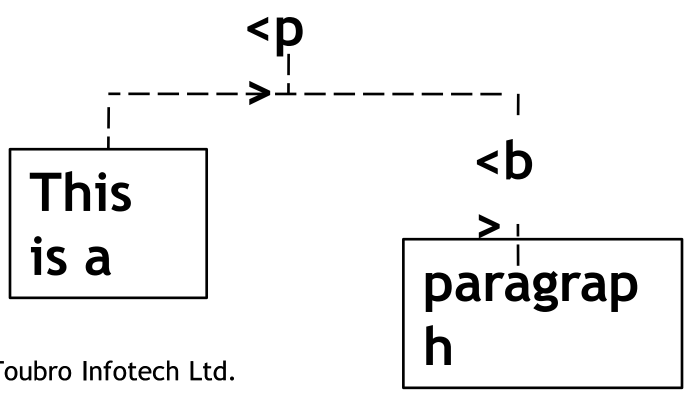
- a W3C (World Wide Web Consortium) standard.
- a object-oriented representation of the web page as a structured group of nodes and objects that have properties and methods.
- It can be modified with a scripting language such as JavaScript.

nodes

- Everything in HTML document is a node.
- <p>This is a paragraph</p>
  - Two nodes are created - an *element* P and a *text node* with content '*This is a paragraph*'.
  - The text node is inside the element, so it is considered a *child node* of the element.



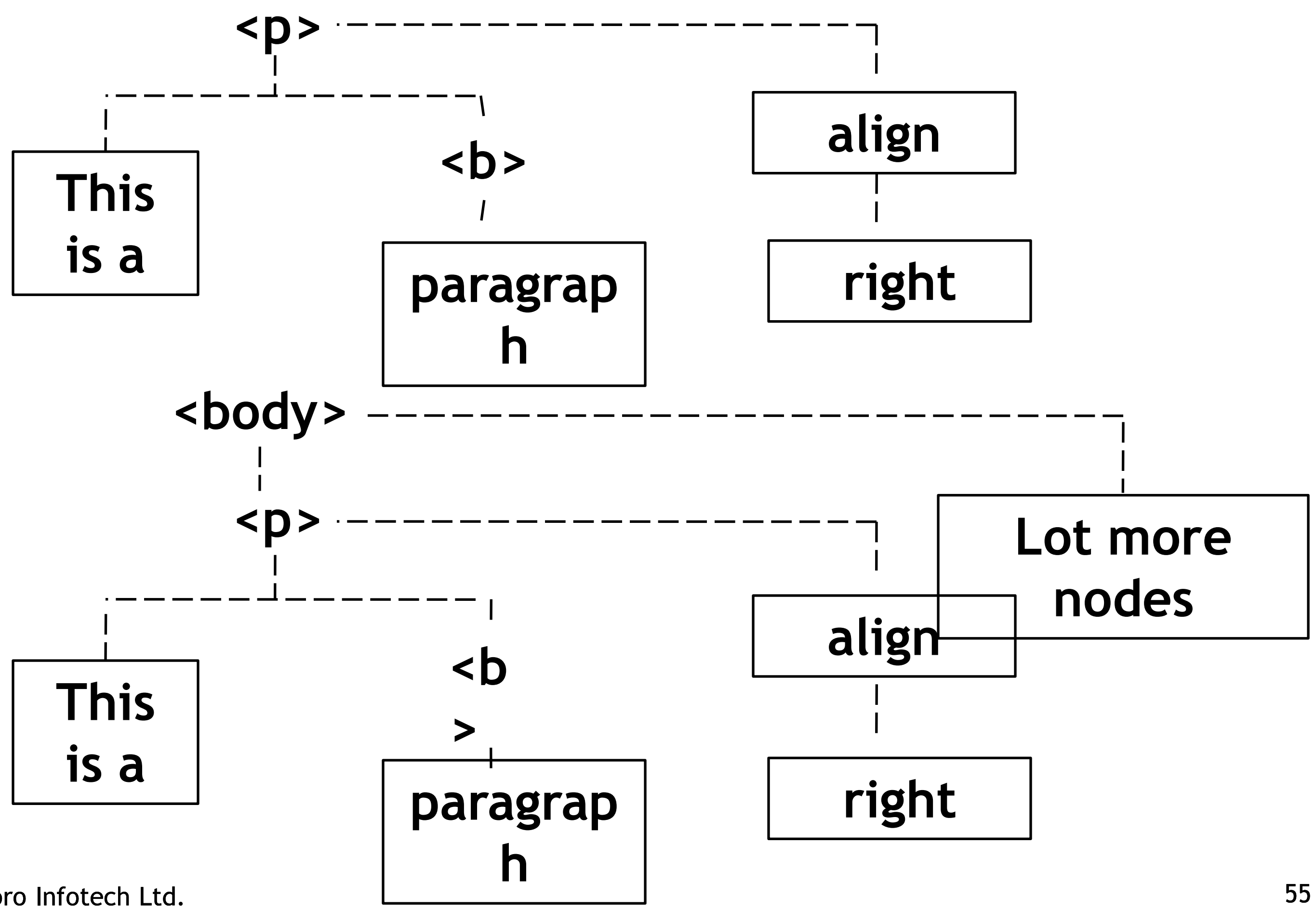
- **If you do - <p>This is a <b>paragraph</b></p>**



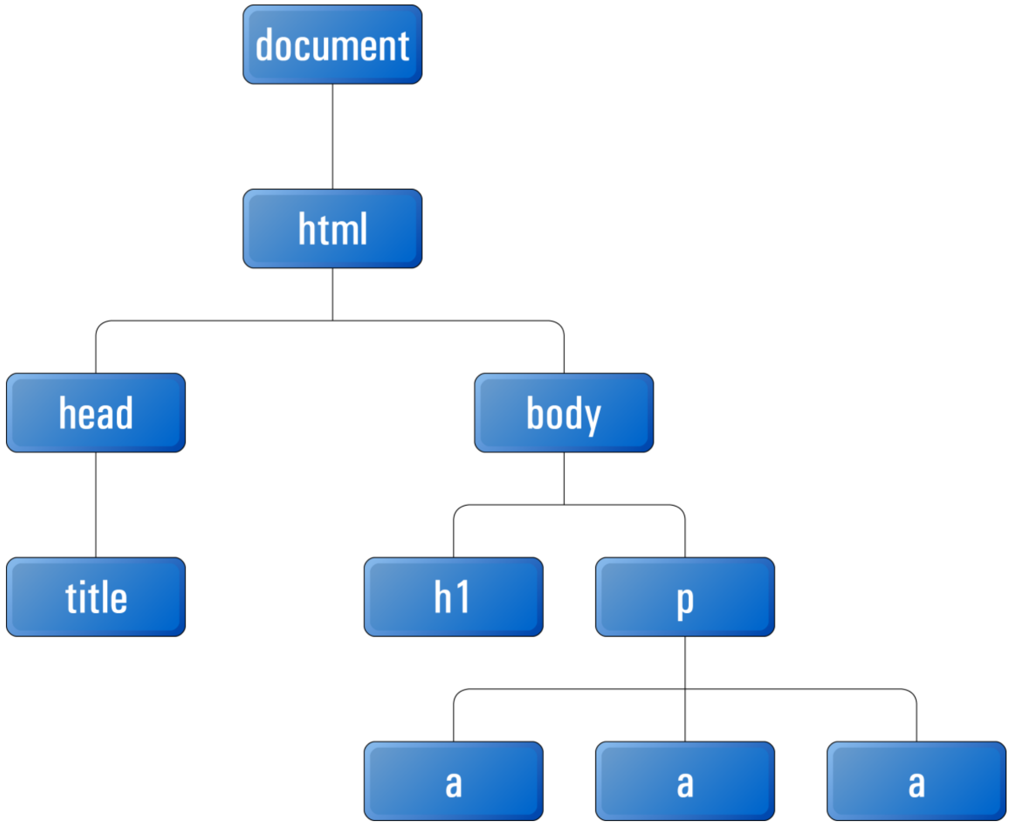
L&T Infotech

Nodes continued

- Finally there are attributes –
- <p align="right">This is a <b>paragraph</b></p>



The picture



jQuery

L&T Infotech

jQuery Introduction

- *jQuery* is a fast and concise JavaScript library created by John Resig in 2006.
- *jQuery* simplifies HTML document traversing, event handling, animating, and Ajax interactions for Rapid Web Development.

jQuery Introduction

What is jQuery?

- jQuery is a library of JavaScript Functions.
- jQuery is a lightweight "write less, do more" JavaScript library.
- jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

L&T Infotech

jQuery Introduction

- jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery:
  - **DOM manipulation:** The jQuery made it easy to select DOM elements, traverse them and modifying their content by using cross-browser open source selector engine called **Sizzle**.
  - **Event handling:** The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
  - **AJAX Support:** The jQuery helps you a lot to develop a responsive and feature-rich site using AJAX technology.
  - **Animations:** The jQuery comes with plenty of built-in animation effects which you can use in your websites.
  - **Lightweight:** The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
  - **Cross Browser Support:** The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
  - **Latest Technology:** The jQuery supports CSS3 selectors and basic XPath syntax.

Adding the jQuery Library to Your Pages

- The jQuery library is stored a single JavaScript file, containing all the jQuery functions.
- It can be added to a web page with the following mark-up:

```
<head>
<script type="text/javascript" src="jquery-1.6.1.min.js"></script>
</head>
```

Please note that the <script> tag should be inside the page's <head> section.

How to use jQuery library?

- You can include *jQuery* library in your HTML file as follows:

```
<html>
<head>
<title>The jQuery Example</title>
<script type="text/javascript" src="jquery-1.6.1.min.js"></script>
<script type="text/javascript">
            // you can add our javascript code here
</script>
</head>
<body>
            ........
</body>
</html>
```

Downloading jQuery

- Two versions of jQuery are available for downloading:
  - one minified and one uncompressed (for debugging or reading).

- Both versions can be downloaded from **www.jQuery.com**

Alternatives to Downloading

- If you don't want to store the jQuery library on your own computer, you can use the hosted jQuery library from Google or Microsoft.

Google

```
<head>
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></sc
ript>
</head>
```

Microsoft

```
<head>
<script type="text/javascript"
src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.min.js"></script>
</head>
```

jQuery Syntax

- **How to call a jQuery library functions?**
  - As almost everything we do when using jQuery reads or manipulates the document object model (DOM), we need to make sure that we start adding events etc. as soon as the DOM is ready.
  - If you want an event to work on your page, you should call it inside the $(document).ready() function. Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.

jQuery Syntax

- To do this, we register a ready event for the document as follows:

```
$(document).ready(function(){

   // jQuery functions go here...

});
```

jQuery Selectors

- To call upon any jQuery library function, use HTML script tags as shown below:

Demo1.html

```
<html>
<head>
<title>The jQuery Example</title>
<script type="text/javascript" src="jquery-1.6.1.min.js"></script>
<script type="text/javascript" language="javascript">
            $(document).ready(function()
            {
                        $("div").click(function()
                        {
                                    alert("Hello world!");
                        });
            });
</script>
</head>
<body>
<div id="newdiv"> Click on this to see a dialogue box. </div>
</body> </html>
```

Custom Scripts

- **How to use Custom Scripts?**
  - It is better to write our custom code in the custom JavaScript file : **custom.js**, as follows:

```
/* Filename: custom.js */
$(document).ready(function()
{
		$("div").click(function()
		{
				alert("Hello world!");
		});
});
```

L&T Infotech

Custom Scripts

- **How to use Custom Scripts?**
  - It is better to write our custom code in the custom JavaScript file : **custom.js**, as follows:

```
<html>
<head>
<title>The jQuery Example</title>
<script type="text/javascript" src=" jquery-1.6.1.min.js "></script>
<script type="text/javascript" src="custom.js"></script>
</head>
<body>
<div id="newdiv"> Click on this to see a dialogue box. </div>
</body>
</html>
```

Using Multiple Libraries

- You can use multiple libraries all together without conflicting each others. For example you can use jQuery and other javascript libraries to
- Many JavaScript libraries use ‘$’ as a function or variable name, just as jQuery does. In jQuery's case, ‘$’ is just an alias for jQuery, so all functionality is available without using ‘$’.
- Run **$.noConflict()** method to give control of the $ variable back to whichever library first implemented it. This helps to make sure that jQuery doesn't conflict with the ‘$’ object of other libraries:

Using Multiple Libraries

- Here is simple way of avoiding any conflict:

```
// Import other library
// Import jQuery
$.noConflict();
// Code that uses other library's $ can follow here.
```

Using Multiple Libraries

- This technique is especially effective in conjunction with the .ready() method's ability to alias the jQuery object, as within the .ready() we can use $ if we wish without fear of conflicts later:

```
// Import other library
// Import jQuery
$.noConflict();
jQuery(document).ready(function($)
{
            // Code that uses jQuery's $ can follow here.
});
// Code that uses other library's $ can follow here.
DOM Element
```

jQuery - Basics

jQuery is a framework built using JavaScript capabilities. So you can use all the functions and other capabilities available in JavaScript.

This chapter would explain most basic concepts but frequently used in jQuery.

- **String:**
  - A string in JavaScript is an immutable object that contains none, one or many characters.
  - Following are the valid examples of a JavaScript String:

```
"This is JavaScript String"
'This is JavaScript String'
‘This is "really" a JavaScript String'
"This is 'really' a JavaScript String"
```

jQuery - Basics

- **Numbers:**
  - Numbers in JavaScript are double-precision 64-bit format IEEE 754 values. They are immutable, just as strings.
  - Following are the valid examples of a JavaScript Numbers:

```
5350
120.27
0.26
```

jQuery - Basics

- **Boolean:**
  - A boolean in JavaScript can be either **true** or **false**. If a number is zero, it defaults to false. If an empty string defaults to false:
  - Following are the valid examples of a JavaScript Boolean:

```
true // true
false // false
0 // false
1 // true
"" // false
"hello" // true
```

jQuery - Basics

- **Objects:**
  - JavaScript supports Object concept very well. You can create an object using the object literal as follows:

```
var emp = { name: "Zara", age: 10 };
```

  - You can write and read properties of an object using the dot notation as follows:

```
// Getting object properties
emp.name // ==> Zara
emp.age // ==> 10

// Setting object properties
emp.name = "Daisy" // <== Daisy
emp.age = 20 // <== 20
```

jQuery - Basics

- **Arrays:**
  - You can define arrays using the array literal as follows:

```
var x = [];
var y = [1, 2, 3, 4, 5];
```

  - An array has a length property that is useful for iteration:

```
var x = [1, 2, 3, 4, 5];
for (var i = 0; i < x.length; i++)
{
            // Do something with x[i]
}
```

jQuery - Basics

- **Functions:**
  - A function in JavaScript can be either named or anonymous. A named function can be defined using *function* keyword as follows:

```
function named()
{
            // do some stuff here
}
```

  - An anonymous function can be defined in similar way as a normal function but it would not have any name.
  - A anonymous function can be assigned to a variable or passed to a method as shown below.

```
var handler = function ()
{
            // do some stuff here
}
```

jQuery - Basics

– JQuery makes a use of anonymous functions very frequently as follows:

```
$(document).ready(function()
{
		// do some stuff here
});
```

- **Arguments:**
  - JavaScript variable *arguments* is a kind of array which has *length* property. Following example explains it very well:

```
function func(x)
{
            console.log(typeof x, arguments.length);
}
func(); //==> "undefined", 0
func(1); //==> "number", 1
func("1", "2", "3"); //==> "string", 3
```

jQuery - Basics

– The arguments object also has a *callee* property, which refers to the function you're inside of. For example:

```
function func()
{
            return arguments.callee;
}
func(); // ==> func
```

- **Context:**
  - JavaScript famous keyword **this** always refers to the current context. Within a function **this** context can change, depending on how the function is called:

```
$(document).ready(function()
{
            // this refers to window.document
});

$("div").click(function()
{
            // this refers to a div DOM element
});
```

  - You can specify the context for a function call using the function-built-in methods **call()** and **apply()** methods.

– The difference between them is how they pass arguments. Call passes all arguments through as arguments to the function, while apply accepts an array as the arguments.

```
function scope()
{
        console.log(this, arguments.length);
}

scope() // window, 0
scope.call("foobar", [1,2]); //==> "foobar", 1
scope.apply("foobar", [1,2]); //==> "foobar", 2
```

jQuery - Basics

- **Callback:**
  - A callback is a plain JavaScript function passed to some method as an argument or option. Some callbacks are just events, called to give the user a chance to react when a certain state is triggered.
  - jQuery's event system uses such callbacks everywhere for example:

```
$("body").click(function(event)
{
        console.log("clicked: " + event.target);
});
```

  - Most callbacks provide arguments and a context. In the event-handler example, the callback is called with one argument, an Event.

jQuery - Basics

– Some callbacks are required to return something, others make that return value optional. To prevent a form submission, a submit event handler can return false as follows:

```
$("#myform").submit(function()
{
            return false;
});
```

jQuery - Selectors

- The jQuery library harnesses the power of Cascading Style Sheets (CSS) selectors to let us quickly and easily access elements or groups of elements in the Document Object Model (DOM).
- A jQuery Selector is a function which makes use of expressions to find out matching elements from a DOM based on the given criteria.

jQuery - Selectors

- **The $() factory function:**
  - All type of selectors available in jQuery, always start with the dollar sign and parentheses: **$()**.
  - The factory function **$()** makes use of following three building blocks while selecting elements in a given document:

jQuery	Description
Tag Name:	Represents a tag name available in the DOM. For example **$('p')** selects all paragraphs in the document.
Tag ID:	Represents a tag available with the given ID in the DOM. For example **$('#some-id')** selects the single element in the document that has an ID of some-id.
Tag Class:	Represents a tag available with the given class in the DOM. For example **$('.some-class')** selects all elements in the document that have a class of some-class.

jQuery - Selectors

- All the items in previous table can be used either on their own or in combination with other selectors. All the jQuery selectors are based on the same principle except some tweaking.

NOTE: The factory function **$()** is a synonym of **jQuery()** function. So in case you are using any other JavaScript library where **$** sign is conflicting with some thing else then you can replace **$** sign by **jQuery** name and you can use function **jQuery()** instead of **$()**.

L&T Infotech

jQuery - Selectors

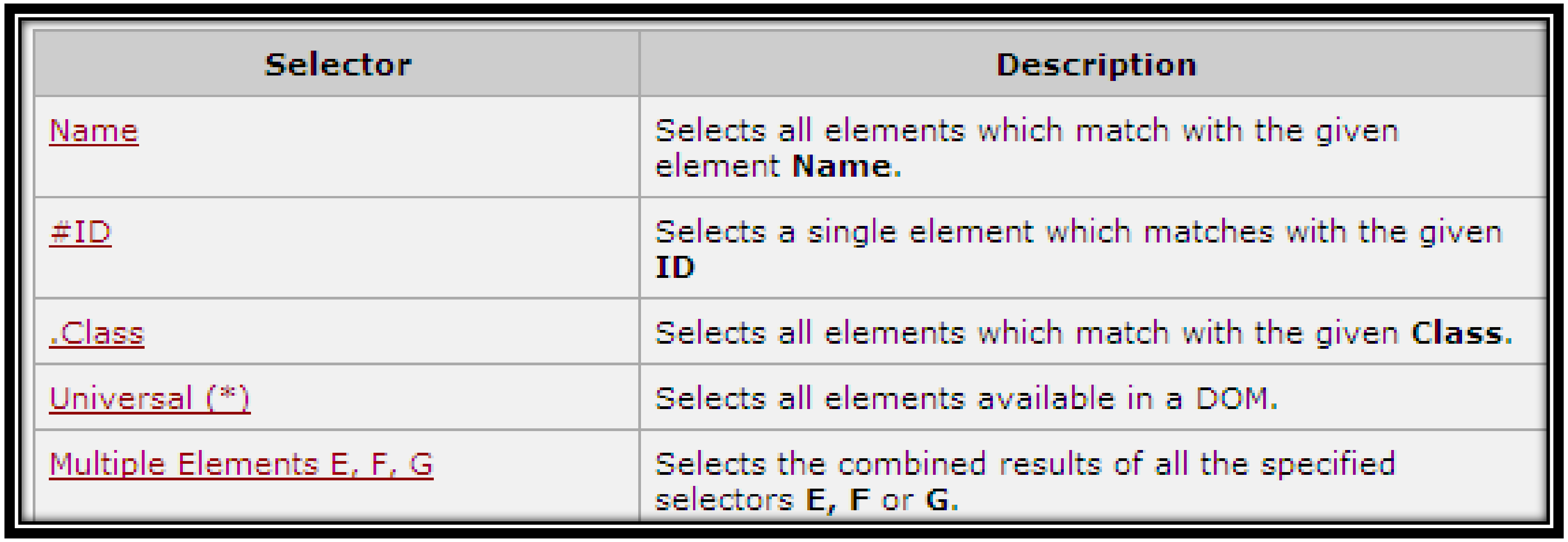
– **Example:**

Demo2.html

```
<html>
<head>
<title>the title</title>
<script type="text/javascript" src=" jquery-1.6.1.min.js"></script>
<script type="text/javascript" language="javascript">
$(document).ready(function()
{
            var pars = $("p");
            for( i=0; i<pars.length; i++ )
            {
                        alert("Found paragraph: " + pars[i].innerHTML);
            }
});
</script>
</head>
<body>
<div>
<p class="myclass">This is a paragraph.</p>
<p id="myid">This is second paragraph.</p> <p>This is third paragraph.</p>
</div> </body> </html>
```

jQuery - Selectors

- **How to use Selectors?**
  - The selectors are very useful and would be required at every step while using jQuery. They get the exact element that you want from your HTML document.
  - Following table lists down few basic selectors and explains them with examples.



Selector	Description
Name	Selects all elements which match with the given element **Name.**
#ID	Selects a single element which matches with the given **ID**
.Class	Selects all elements which match with the given **Class.**
Universal (*)	Selects all elements available in a DOM.
Multiple Elements E, F, G	Selects the combined results of all the specified selectors **E, F** or **G.**

jQuery - Selectors

- Similar to previous syntax and examples, following examples would give you understanding on using different type of other useful selectors:
  - **$('*'):** This selector selects all elements in the document.
  - **$("p > *"):** This selector selects all elements that are children of a paragraph element.
  - **$("#specialID"):** This selector function gets the element with id="specialID".
  - **$(".specialClass"):** This selector gets all the elements that have the class of *specialClass*.
  - **$("li:not(.myclass)"):** Selects all elements matched by <li> that do not have class="myclass".
  - **$("a#specialID.specialClass"):** This selector matches links with an id of *specialID* and a class of *specialClass*.

jQuery - Selectors

- **$("p a.specialClass"):** This selector matches links with a class of *specialClass* declared within <p> elements.
- **$("ul li:first"):** This selector gets only the first <li> element of the <ul>.
- **$("#container p"):** Selects all elements matched by <p> that are descendants of an element that has an id of *container*.
- **$("li > ul"):** Selects all elements matched by <ul> that are children of an element matched by <li>
- **$("strong + em"):** Selects all elements matched by <em> that immediately follow a sibling element matched by <strong>.
- **$("p ~ ul"):** Selects all elements matched by <ul> that follow a sibling element matched by <p>.
- **$("code, em, strong"):** Selects all elements matched by <code> or <em> or <strong>.

jQuery - Selectors

- **$("p strong, .myclass"):** Selects all elements matched by <strong> that are descendants of an element matched by <p> as well as all elements that have a class of *myclass*.
- **$(":empty"):** Selects all elements that have no children.
- **$("p:empty"):** Selects all elements matched by <p> that have no children.
- **$("div[p]"):** Selects all elements matched by <div> that contain an element matched by <p>.
- **$("p[.myclass]"):** Selects all elements matched by <p> that contain an element with a class of *myclass*.
- **$("a[@rel]"):** Selects all elements matched by <a> that have a rel attribute.
- **$("input[@name=myname]"):** Selects all elements matched by <input> that have a name value exactly equal to *myname*.

jQuery - Selectors

- **$("input[@name^=myname]"):** Selects all elements matched by <input> that have a name value beginning with *myname*.
- **$("a[@rel$=self]"):** Selects all elements matched by <p> that have a class value ending with *bar*
- **$("a[@href*=domain.com]"):** Selects all elements matched by <a> that have an href value containing domain.com.
- **$("li:even"):** Selects all elements matched by <li> that have an even index value.
- **$("tr:odd"):** Selects all elements matched by <tr> that have an odd index value.
- **$("li:first"):** Selects the first <li> element.
- **$("li:last"):** Selects the last <li> element.
- **$("li:visible"):** Selects all elements matched by <li> that are visible.
- **$("li:hidden"):** Selects all elements matched by <li> that are hidden.

jQuery - Selectors

- **$(":radio"):** Selects all radio buttons in the form.
- **$(":checked"):** Selects all checked boxex in the form.
- **$(":input"):** Selects only form elements (input, select, textarea, button).
- **$(":text"):** Selects only text elements (input[type=text]).
- **$("li:eq(2)"):** Selects the third <li> element
- **$("li:eq(4)"):** Selects the fifth <li> element
- **$("li:lt(2)"):** Selects all elements matched by <li> element before the third one; in other words, the first two <li> elements.
- **$("p:lt(3)"):** selects all elements matched by <p> elements before the fourth one; in other words the first three <p> elements.
- **$("li:gt(1)"):** Selects all elements matched by <li> after the second one.

L&T Infotech

jQuery - Selectors

- **$("p:gt(2)"):** Selects all elements matched by <p> after the third one.
- **$("div/p"):** Selects all elements matched by <p> that are children of an element matched by <div>.
- **$("div//code"):** Selects all elements matched by <code>that are descendants of an element matched by <div>.
- **$("//p//a"):** Selects all elements matched by <a> that are descendants of an element matched by <p>
- **$("li:first-child"):** Selects all elements matched by <li> that are the first child of their parent.
- **$("li:last-child"):** Selects all elements matched by <li> that are the last child of their parent.
- **$(":parent"):** Selects all elements that are the parent of another element, including text.
- **$("li:contains(second)"):** Selects all elements matched by <li> that contain the text second.

L&T Infotech

jQuery - Selectors

- You can use all the previous selectors with any HTML/XML element in generic way.
- For example if selector **$("li:first")** works for <li> element then **$("p:first")** would also work for <p> element.

jQuery - DOM Attributes

- Some of the most basic components we can manipulate when it comes to DOM elements are the properties and attributes assigned to those elements.
- Most of these attributes are available through JavaScript as DOM node properties. Some of the more common properties are:
  - className
  - tagName
  - id
  - href
  - title
  - rel
  - src

jQuery - DOM Attributes

- Consider the following HTML markup for an image element:

```
<img id="myImage" src="image.gif" alt="An image" class="someClass" title="This
is an image"/>
```

- In this element's markup, the tag name is img, and the markup for id, src, alt, class, and title represents the element's attributes, each of which consists of a name and a value.
- jQuery gives us the means to easily manipulate an element's attributes and gives us access to the element so that we can also change its properties.

jQuery - DOM Attributes

- **Get Attribute Value:**
  - The **attr()** method can be used to either fetch the value of an attribute from the first element in the matched set or set attribute values onto all matched elements.

L&T Infotech

jQuery - DOM Attributes

- **Example:**
  - Following is a simple example which fetches title attribute of <em> tag and set <div id="divid"> value with the same value:

Demo3.html

```
<html>
<head>
<title>the title</title>
<script type="text/javascript" src="jquery-1.6.1.min.js">
</script>
<script type="text/javascript" language="javascript">
$(document).ready(function()
{
            var title = $("em").attr("title");
            $("#divid").text(title);
});
</script>
</head>
<body> <div> <em title="Bold and Brave">This is first paragraph.</em> <p
id="myid">This is second paragraph.</p> <div id="divid"></div> </div>
</body> </html>
```

jQuery - DOM Attributes

- **Set Attribute Value:**
  - The **attr(name, value)** method can be used to set the named attribute onto all elements in the wrapped set using the passed value.

jQuery - DOM Attributes

- **Example:**
  - Following is a simple example which set **src** attribute of an image tag to a correct location:

Demo4.html

```
<html>
<head>
<title>the title</title>
<script type="text/javascript" src="jquery-1.6.1.min.js"></script>
<script type="text/javascript" language="javascript">
$(document).ready(function()
{
            $("#myimg").attr("src", "/images/jquery.jpg");
});
</script>
</head>
<body>
<div>
            <img id="myimg" src="/wongpath.jpg" alt="Sample image" />
</div>
</body> </html>
```

jQuery - DOM Attributes

- **Applying Styles:**
  - The **addClass(classes)** method can be used to apply defined style sheets onto all the matched elements. You can specify multiple classes separated by space.

jQuery - DOM Attributes

- **Example:**
  - Following is a simple example which set **src** attribute of an image tag to a correct location:

Demo5.html

```
<html>
<head>
<title>the title</title>
<script type="text/javascript" src="jquery-1.6.1.min.js"></script>
<script type="text/javascript" language="javascript">
$(document).ready(function()
{
            $("em").addClass("selected");
            $("#myid").addClass("highlight");
});
</script>
<style>
.selected { color:red; }
.highlight { background:yellow; }
</style>
</head> <body> <em title="Bold and Brave">This is first paragraph.</em> <p
id="myid">This is second paragraph.</p> </body> </html>
```

jQuery - DOM Traversing

- jQuery is a very powerful tool which provides a variety of DOM traversal methods to help us select elements in a document randomly as well as in sequential method.
- Most of the DOM Traversal Methods do not modify the jQuery object and they are used to filter out elements from a document based on given conditions.

jQuery - DOM Traversing

- **Find Elements by index:**
  - Consider a simple document with the following HTML content:

```
<html>
<head>
<title>the title</title>
</head>
<body>
<div>
<ul>
		<li>list item 1</li>
		<li>list item 2</li>
		<li>list item 3</li>
		<li>list item 4</li>
		<li>list item 5</li>
		<li>list item 6</li>
</ul>
</div>
</body>
</html>
```

jQuery - DOM Traversing

- In previous example, every list has its own index, and can be located directly by using **eq(index)** method as below example.
- Every child element starts its index from zero, thus, *list item 2* would be accessed by using **$("li").eq(1)** and so on.

Look Demo6.html to get effect

jQuery - CSS Methods

- The jQuery library supports nearly all of the selectors included in Cascading Style Sheet (CSS) specifications 1 through 3, as outlined on the World Wide Web Consortium's site.
- Using JQuery library developers can enhance their websites without worrying about browsers and their versions as long as the browsers have JavaScript enabled.
- Most of the JQuery CSS Methods do not modify the content of the jQuery object and they are used to apply CSS properties on DOM elements.

L&T Infotech

jQuery - CSS Methods

- **Apply CSS Properties:**
  - This is very simple to apply any CSS property using JQuery method **css(PropertyName, PropertyValue)**.
  - Here is the syntax for the method:

```
selector.css( PropertyName, PropertyValue );
```

  - Here you can pass *PropertyName* as a javascript string and based on its value, *PropertyValue* could be string or integer.

jQuery - CSS Methods

- **Example:**
  - Following is an example which adds font color to the second list item.

Demo7.html

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="jquery-1.6.1.min.js"></script>
   <script type="text/javascript" language="javascript">
   $(document).ready(function() {
      $("li").eq(2).css("color", "red");
   });
   </script>
</head>
<body>
   <div><ul>
     <li>list item 1</li><li>list item 2</li><li>list item 3</li>
     <li>list item 4</li><li>list item 5</li><li>list item 6</li></ul>
   </div>
</body>
</html>
```

jQuery - CSS Methods

- **Apply Multiple CSS Properties:**
  - You can apply multiple CSS properties using a single JQuery method **CSS({key1:val1, key2:val2....)**. You can apply as many properties as you like in a single call.
  - Here is the syntax for the method:

```
selector.css( {key1:val1, key2:val2....keyN:valN})
```

  - Here you can pass key as property and val as its value as described above.

jQuery - CSS Methods

- **Example:**
  - Following is an example which adds font color to the second list item.

Demo8.html

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="jquery-1.6.1.min.js"></script>
   <script type="text/javascript" language="javascript">
         $(document).ready(function()
        {
                $("li").eq(2).css({"color":"red", "background-color":"green"});
        });
   </script>
</head>
<body>
   <div><ul>
     <li>list item 1</li><li>list item 2</li><li>list item 3</li>
     <li>list item 4</li><li>list item 5</li><li>list item 6</li></ul>
   </div>
</body>
</html>
```

jQuery - DOM Manipulation Methods

- JQuery provides methods to manipulate DOM in efficient way. You do not need to write big code to modify the value of any element's attribute or to extract HTML code from a paragraph or division.
- JQuery provides methods such as .attr(), .html(), and .val() which act as getters, retrieving information from DOM elements for later use.

jQuery - DOM Manipulation Methods

- **Content Manipulation:**
  - The **html( )** method gets the html contents (innerHTML) of the first matched element.
  - Here is the syntax for the method:

```
selector.html( )
```

jQuery - DOM Manipulation Methods

- **Example:**
  - Following is an example which makes use of .html() and .text(val) methods. Here .html() retrieves HTML content from the object and then .text(val) method sets value of the object using passed parameter:

Refer “Demo9.html” for complete source code

jQuery - DOM Manipulation Methods

- **DOM Element Replacement:**
  - You can replace a complete DOM element with the specified HTML or DOM elements. The **replaceWith(content)** method serves this purpose very well.
  - Here is the syntax for the method:

```
selector.replaceWith( content )
```

  - Here content is what you want to have instead of original element. This could be HTML or simple text.

- **Example:**
  - Following is an example which would replace division element with "<h1>JQuery is Great</h1>": (Refer “Demo10.html”)

jQuery - DOM Manipulation Methods

- **Removing DOM Elements:**
  - There may be a situation when you would like to remove one or more DOM elements from the document. JQuery provides two methods to handle the situation.
  - The **empty( )** method remove all child nodes from the set of matched elements where as the method **remove(expr)** method removes all matched elements from the DOM.
  - Here is the syntax for the method:

```
selector.remove( [ expr ])
or
selector.empty( )
```

  - You can pass optional paramter *expr* to filter the set of elements to be removed.
- **Example:**
  - Following is an example where elements are being removed as soon as they are clicked:(Refer “Demo11.html”)

jQuery - DOM Manipulation Methods

- **Inserting DOM elements:**
  - There may be a situation when you would like to insert new one or more DOM elements in your existing document. JQuery provides various methods to insert elements at various locations.
  - The **after(content)** method insert content after each of the matched elements where as the method **before(content)** method inserts content before each of the matched elements.
  - Here is the syntax for the method:

```
selector.after( content )
or
selector.before( content )
```

  - Here content is what you want to insert. This could be HTML or simple text.

- **Example:**
  - Following is an example where <div> elements are being inserted just before the clicked element: (Refer Demo12.html)

jQuery - Events Handling

- We have the ability to create dynamic web pages by using events. Events are actions that can be detected by your Web Application.
- Following are the examples events:
  - A mouse click
  - A web page loading
  - Taking mouse over an element
  - Submitting an HTML form
  - A keystroke on your keyboard
  - etc.
- When these events are triggered you can then use a custom function to do pretty much whatever you want with the event. These custom functions call Event Handlers.

jQuery - Events Handling

- **Binding event handlers:**
  - Using the jQuery Event Model, we can establish event handlers on DOM elements with the **bind()** method as follows:

Demo13.html

```
$('div').bind('click', function( event )
{
        alert('Hi there!');
});
```

  - This code will cause the division element to respond to the click event; when a user clicks inside this division thereafter, the alert will be shown.

  - The full syntax of the bind() command is as follows:

```
selector.bind( eventType[, eventData], handler)
```

jQuery - Events Handling

- Following is the description of the parameters:
  - **eventType:** A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.
  - **eventData:** This is optional parameter is a map of data that will be passed to the event handler.
  - **handler:** A function to execute each time the event is triggered.

jQuery - Events Handling

- **Removing event handlers:**
  - Typically, once an event handler is established, it remains in effect for the remainder of the life of the page. There may be a need when you would like to remove event handler.
  - jQuery provides the **unbind()** command to remove an exiting event handler. The syntax of unbind() is as follows:

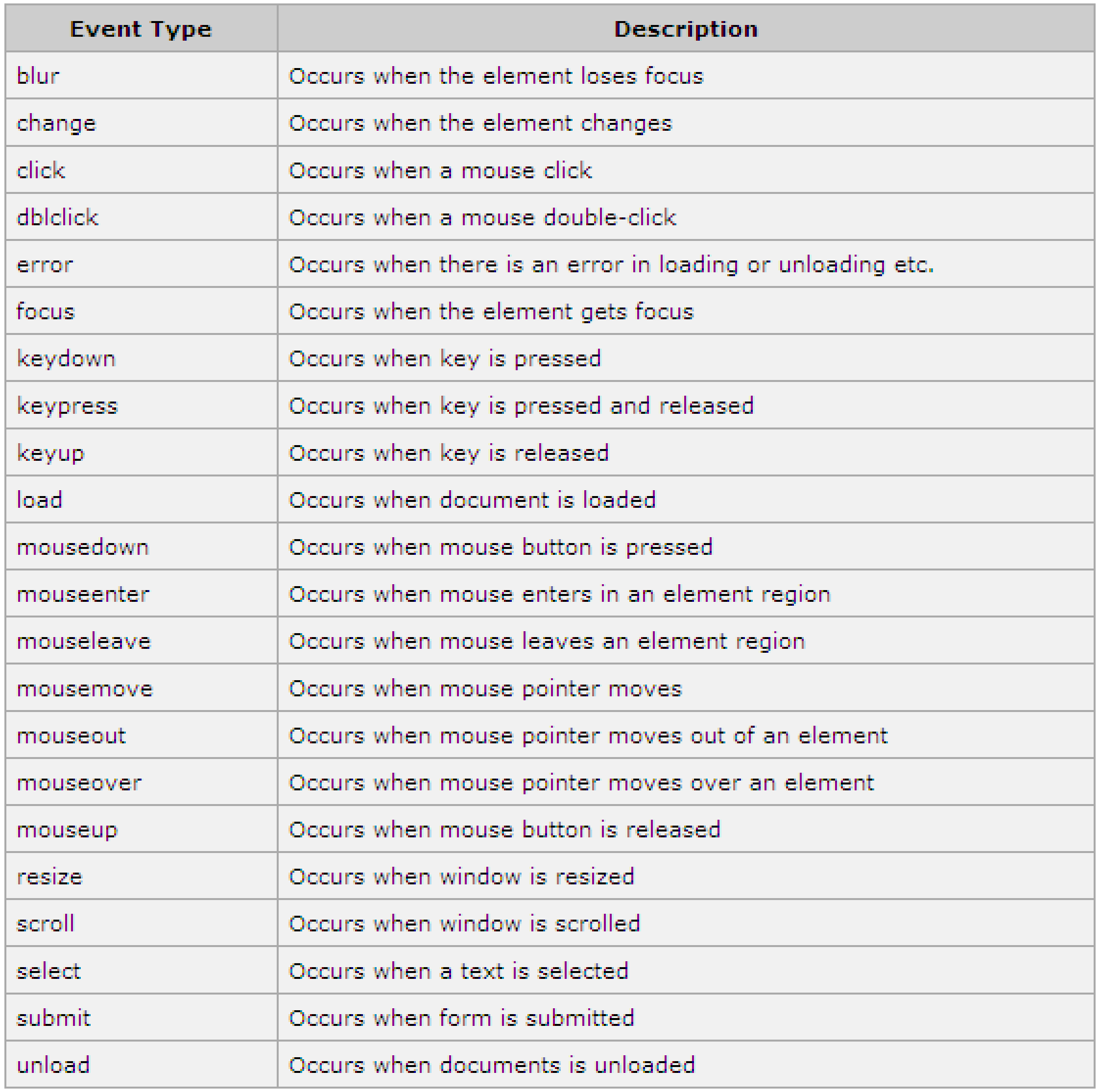
```
selector.unbind(eventType, handler)
or
selector.unbind(eventType)
```

  - Following is the description of the parameters:
    - **eventType:** A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.
    - **handler:** If provided, identifies the specific listener that.s to be removed.

L&T Infotech

jQuery - Events Handling

- **Event Types:**
  - The following are cross platform and recommended event types which you can bind using JQuery



Event Type	Description
blur	Occurs when the element loses focus
change	Occurs when the element changes
click	Occurs when a mouse click
dblclick	Occurs when a mouse double-click
error	Occurs when there is an error in loading or unloading etc.
focus	Occurs when the element gets focus
keydown	Occurs when key is pressed
keypress	Occurs when key is pressed and released
keyup	Occurs when key is released
load	Occurs when document is loaded
mousedown	Occurs when mouse button is pressed
mouseenter	Occurs when mouse enters in an element region
mouseleave	Occurs when mouse leaves an element region
mousemove	Occurs when mouse pointer moves
mouseout	Occurs when mouse pointer moves out of an element
mouseover	Occurs when mouse pointer moves over an element
mouseup	Occurs when mouse button is released
resize	Occurs when window is resized
scroll	Occurs when window is scrolled
select	Occurs when a text is selected
submit	Occurs when form is submitted
unload	Occurs when documents is unloaded

L&T Infotech

jQuery - Effects

- jQuery provides a trivially simple interface for doing various kind of amazing effects. jQuery methods allow us to quickly apply commonly used effects with a minimum configuration.
- This tutorial covers all the important jQuery methods to create visual effects.

- **Showing and Hiding elements:**
  - The commands for showing and hiding elements are pretty much what we would expect: **show()** to show the elements in a wrapped set and **hide()** to hide them.
  - **Syntax:**
  - Here is the simple syntax for **show()** method:

```
[selector].show( speed, [callback] );
```

jQuery - Effects

- Here is the description of all the parameters:
  - **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
  - **callback:** This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.
- Following is the simple syntax for **hide()** method:

```
[selector].hide( speed, [callback] );
```

- Here is the description of all the parameters:
  - **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
  - **callback:** This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

For Working Example refer "Demo14.html"

jQuery - Effects

- **Toggling the elements:**
  - jQuery provides methods to toggle the display state of elements between revealed or hidden. If the element is initially displayed, it will be hidden; if hidden, it will be shown.
  - **Syntax:**
  - Here is the simple syntax for one of the **toggle()** methods:

```
[selector]..toggle([speed][, callback]);
```

  - Here is the description of all the parameters:
    - **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
    - **callback:** This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

For Working Example refer “Demo15.html”

jQuery Event live() Method

- The live() method attaches one or more event handlers for selected elements, and specifies a function to run when the events occur.

Event handlers attached using the live() method will work for both current and FUTURE elements matching the selector (like a new element created by a script).

- **Syntax**

$(*selector*).live(*event*,*data*,*function*)

Parameter	Description
event	Required. Specifies one or more events to attach to the elements. Multiple event values are separated by space. Must be a valid event.
data	Optional. Specifies additional data to pass along to the function
function	Required. Specifies the function to run when the event(s) occur

L&T Infotech

jQuery Event delegate() Method

- The delegate() method attaches one or more event handlers for specified elements that are children of selected elements, and specifies a function to run when the events occur.

Event handlers attached using the delegate() method will work for both current and FUTURE elements (like a new element created by a script).

- **Syntax**

$(*selector*).delegate(*childSelector*,*event*,*data*,*function*)

Parameter	Description
childSelector	Required. Specifies one or more child elements to attach the event handler to
event	Required. Specifies one or more events to attach to the elements. Multiple event values are separated by space. Must be a valid event
data	Optional. Specifies additional data to pass along to the function
function	Required. Specifies the function to run when the event(s) occur

Thank You

Our Business Knowledge,
Your Winning Edge.