Background Properties:

NN: Netscape, **IE**: Internet Explorer, **W3C**: Web Standard

Property	Description	Values	NN	IE	W3C
background	A shorthand property for setting all background properties in one declaration	*background-color* *background-image* *background-repeat* *background-attachment* *background-position*	6.0	4.0	CSS1
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	scroll fixed	6.0	4.0	CSS1
background-color	Sets the background color of an element	*color-rgb* *color-hex* *color-name* transparent	4.0	4.0	CSS1
background-image	Sets an image as the background	*url* none	4.0	4.0	CSS1
background-position	Sets the starting position of a background image	top left top center top right center left center center center right bottom left bottom center bottom right *x-% y-%* *x-pos y-pos*	6.0	4.0	CSS1
background-repeat	Sets if/how a background image will be repeated	repeat repeat-x repeat-y no-repeat	4.0	4.0	CSS1

Text Properties:

NN: Netscape, **IE**: Internet Explorer, **W3C**: Web Standard

Property	Description	Possible Values	NN	IE	W3C
color	Sets the color of a text	*color*	4.0	3.0	CSS1
direction	Sets the text direction	ltr rtl			CSS2
letter-spacing	Increase or decrease the space between characters	normal *length*	6.0	4.0	CSS1
text-align	Aligns the text in an element	left right center justify	4.0	4.0	CSS1
text-decoration	Adds decoration to text	none underline overline line-through blink	4.0	4.0	CSS1
text-indent	Indents the first line of text in an element	*length* *%*	4.0	4.0	CSS1
text-shadow		none *color* *length*			
text-transform	Controls the letters in an element	none capitalize uppercase lowercase	4.0	4.0	CSS1
white-space	Sets how white space inside an element is handled	normal pre nowrap	4.0	5.5	CSS1
word-spacing	Increase or decrease the space between words	normal *length*	6.0	6.0	CSS1

Font Properties:

NN: Netscape, **IE**: Internet Explorer, **W3C**: Web Standard

Property	Description	Values	NN	IE	W3C
font	A shorthand property for setting all of the properties for a font in one declaration	*font-style* *font-variant* *font-weight* *font-size/line-height* *font-family* caption icon menu message-box small-caption status-bar	4.0	4.0	CSS1
font-family	A prioritized list of font family names and/or generic family names for an element	*family-name* *generic-family*	4.0	3.0	CSS1
font-size	Sets the size of a font	xx-small x-small small medium large x-large xx-large smaller larger *length* *%*	4.0	3.0	CSS1
font-size-adjust	Specifies an aspect value for an element that will preserve the x-height of the first-choice font	none *number*			CSS2
font-stretch	Condenses or expands the current font-family	normal wider narrower ultra-condensed extra-condensed condensed semi-condensed semi-expanded			CSS2

		expanded extra-expanded ultra-expanded			
font-style	Sets the style of the font	normal italic oblique	4.0	4.0	CSS1
font-variant	Displays text in a small-caps font or a normal font	normal small-caps	6.0	4.0	CSS1
font-weight	Sets the weight of a font	normal bold bolder lighter 100 200 300 400 500 600 700 800 900			

Border Properties:

NN: Netscape, **IE**: Internet Explorer, **W3C**: Web Standard

Property	Description	Values	NN	IE	W3C
border	A shorthand property for setting all of the properties for the four borders in one declaration	*border-width* *border-style* *border-color*	4.0	4.0	CSS1
border-bottom	A shorthand property for setting all of the properties for the bottom border in one declaration	*border-bottom-width* *border-style* *border-color*	6.0	4.0	CSS1
border-bottom-color	Sets the color of the bottom border	*border-color*	6.0	4.0	CSS2
border-bottom-style	Sets the style of the bottom border	*border-style*	6.0	4.0	CSS2
border-bottom-width	Sets the width of the	thin	4.0	4.0	CSS1

	bottom border	medium thick *length*			
border-color	Sets the color of the four borders, can have from one to four colors	*color*	6.0	4.0	CSS1
border-left	A shorthand property for setting all of the properties for the left border in one declaration	*border-left-width* *border-style* *border-color*	6.0	4.0	CSS1
border-left-color	Sets the color of the left border	*border-color*	6.0	4.0	CSS2
border-left-style	Sets the style of the left border	*border-style*	6.0	4.0	CSS2
border-left-width	Sets the width of the left border	thin medium thick *length*	4.0	4.0	CSS1
border-right	A shorthand property for setting all of the properties for the right border in one declaration	*border-right-width* *border-style* *border-color*	6.0	4.0	CSS1
border-right-color	Sets the color of the right border	*border-color*	6.0	4.0	CSS2
border-right-style	Sets the style of the right border	*border-style*	6.0	4.0	CSS2
border-right-width	Sets the width of the right border	thin medium thick *length*	4.0	4.0	CSS1
border-style	Sets the style of the four borders, can have from one to four styles	none hidden dotted dashed solid double groove ridge inset outset	6.0	4.0	CSS1
border-top	A shorthand property for	*border-top-width*	6.0	4.0	CSS1

	setting all of the properties for the top border in one declaration	*border-style* *border-color*			
border-top-color	Sets the color of the top border	*border-color*	6.0	4.0	CSS2
border-top-style	Sets the style of the top border	*border-style*	6.0	4.0	CSS2
border-top-width	Sets the width of the top border	thin medium thick *length*	4.0	4.0	CSS1
border-width	A shorthand property for setting the width of the four borders in one declaration, can have from one to four values				

Margin Properties:

NN: Netscape, **IE**: Internet Explorer, **W3C**: Web Standard

Property	Description	Values	NN	IE	W3C
margin	A shorthand property for setting the margin properties in one declaration	*margin-top* *margin-right* *margin-bottom* *margin-left*	4.0	4.0	CSS1
margin-bottom	Sets the bottom margin of an element	auto *length* *%*	4.0	4.0	CSS1
margin-left	Sets the left margin of an element	auto *length* *%*	4.0	3.0	CSS1
margin-right	Sets the right margin of an element	auto *length* *%*	4.0	3.0	CSS1
margin-top	Sets the top margin of an element	auto *length* *%*	4.0	3.0	CSS1

Padding Properties:

NN: Netscape, **IE**: Internet Explorer, **W3C**: Web Standard

Property	Description	Values	NN	IE	W3C
padding	A shorthand property for setting all of the padding properties in one declaration	*padding-top* *padding-right* *padding-bottom* *padding-left*	4.0	4.0	CSS1
padding-bottom	Sets the bottom padding of an element	*length* *%*	4.0	4.0	CSS1
padding-left	Sets the left padding of an element	*length* *%*	4.0	4.0	CSS1
padding-right	Sets the right padding of an element	*length* *%*	4.0	4.0	CSS1
padding-top	Sets the top padding of an element	*length* *%*	4.0	4.0	CSS1

Lecture 2

What we will Learn

- **Heading tag:** <h1>, <h2>, <h3>, <h4>, <h5>, <h6>
- **
 tag**
- **Comment Tag:** <!-- -- >

In the last lecture, we learned that the first tag on the web page we made is the HTML tag which tells the browser that the document is HTML;

<html>

The last tag on the page is the end HTML tag, which tells the browser that the HTML bit has finished;

</html>

Did you notice the '/' bit which tells the browser that this is the end tag, not the start?

Heading

Today we will learn about heading tag, which looks like <h1> or <h2> or <h3> or <h4> or <h5> or <h6>.
Let us take an example to understand this tag.

```
<html>
<head>
<title >My Home Page with a heading</title>
</head>
<body>
<h1>My Heading</h1>
You write things here...
</body>
</html>
```

Write the code above and save as an .html file.
And the output looks something like this

My Heading

You write things here….

Take another example

```
<html>
<head>
<title >My Home Page with a heading</title>
</head>
<body>
<h1>Heading 1</h1>
```

```
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
<p>This is normal text!</p>
</body>
</html>
```

And the output is now,

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

This is normal text!

So, that was different ways of representing text.

Let us study about
 tag. Please write the code below and save as an .html file

```
<html>
<head>
<title >My Home Page with a heading</title>
</head>
<body>
   My name is Dileep Kumar.
   I am studying at University of Suwon        .
</body>
</html>
```

The output will be

My name is Dileep Kumar I am studying at University of Suwon

Q) Can any one tell me what is the problem with the above output?
A) When I wrote the code, I made two lines, one line contains my name and the other line contains the name of University. But the output shows two lines one after the other.
So that means, HTML does not interpret the <return> key.

The solution to the above problem is
 take.
 tag is also called as blank rule. It inserts a line.

```
<html>
<head>
<title >My Home Page with a heading</title>
</head>
<body>
   My name is Dileep Kumar. <br>
   I am studying at University of Suwon        .
</body>
</html>
```

Now the output will be

My name is Dileep Kumar.
I am studying at University of Suwon.

Q What is the output for the code below?

```
<html>
<head>
<title >My Home Page with a heading</title>
</head>
<body>
   My name is Dileep Kumar. <br> I am studying at University of Suwon.
</body>
</html>
```

Answer)

Comments

Q) Why do we need comments?, infact every language supports comments, be it C, C++ or Java?

Answer) When we write a page with a lot of code, and we come back to it later to wonder what we did, it can be made easier if we have comments in the document which don't show up in the browser.

For example

```
<html>
<head>
<!-- This is the title of my page→
```

```
<title>My Page</title>
</head>
<body>
<!--This is the start of my page! As if I didn't know
that - I did say it was a trivial example!-->


<!--Start of my heading-->
<h1>My Heading - What my page is about.</h1>
<!--End of my heading-->

</body>
```

So the output is

My Heading - What my page is about.

Everthing contained in <!-- -- > is not visible on the browser.

Lecture3

Today we will discuss about formatting the text on the browser. That is

- **How to make our text appear as bold, italic or underlined**
- **To set the page color, i.e. background color and Text color**

Formatting the text

The following are the tags for bold, underline, italic

  for bold
<u> </u> for underline
  for italic

Let us look at an example

```
<html>
<head>
        <title>Welcome to my Home Page</title>
        </head>
<body>
        My name is <b>Dileep Kumar</b>.<br>
        I am </i> at <u>University of Suwon</u>.<br>
        My major is <i>computer science</i>
</body>
</html>
```

The output will be

My name is **Dileep Kumar**.
I am at University of Suwon.
My major is *computer science*

Q) What is the output of the following code?

```
html>
<head>
        <title>Welcome to my Home Page</title>
        </head>
<body>
        My name is <b><u><i>Dileep Kumar</b></u></i>.
</body>
</html>
```

A)

Q) What is the default background color of webpage?

A) White.

Q) How can I change the background color of a web page?

A) We can change the background color of a web page by using bgcolor attribute of <body> tag.

Let us discuss an example to understand this
Write the code below and save as an .html file.

```
<html>
<head>
        <title>Welcome to my Home Page</title>
        </head>
<body bgcolor="blue">
        My name is Dileep Kumar

</body>
</html>
```

The output is

The text color can also be changed, i.e. we can change the background color and text color.
Write the code below and save as an .html file.

```
<html>
<head>
        <title>Welcome to my Home Page</title>
        </head>
<body bgcolor="black" text="yellow">
        My name is Dileep Kumar

</body>
</html>
```

The output is

If you put <body bgcolor=”black”> or
<body bgcolor=”#FFE4C4”>, then you will get the same effect on the page color, i.e. background color will be black. The # tells the browser that the number is hexadecimal.

bgcolor="red"	**bgcolor="#FF0000"**

Similarly, by using the colour name or the code shown in the following table, you will get the same colour results;

aliceblue Code=#F0F8FF	**antiquewhite #FAEBD7**	**aqua #00FFFF**	**aquamarine #7FFFD4**
azure #F0FFFF	**beige #F5F5DC**	**bisque #FFE4C4**	**black #FFE4C4**

blanchedalmond #FFEBCD	blue #0000FF	blueviolet #8A2BE2	brown #A52A2A
burlywood #DEB887	cadetblue #5F9EA0	chartreuse #7FFF00	chocolate #D2691E
coral #FF7F50	cornflowerblue #6495ED	cornsilk #FFF8DC	crimson #DC143C
cyan #00FFFF	darkblue #00008B	darkcyan #008B8B	darkgoldenrod #B8860B
darkgray #A9A9A9	darkgray #A9A9A9	darkkhaki #BDB76B	darkmagenta #8B008B
darkolivegreen #556B2F	darkorange #FF8C00	darkorchid #9932CC	darkred #8B0000
darksalmon #E9967A	darkseagreen #8FBC8F	darkslateblue #483D8B	darkslategray #2F4F4F
darkturquoise #00CED1	darkviolet #9400D3	deeppink #FF1493	deepskyblue #00BFFF
dimgray #696969	dodgerblue #1E90FF	firebrick #B22222	floralwhite #FFFAF0
forestgreen #228B22	fuchsia #FF00FF	gainsboro #DCDCDC	ghostwhite #F8F8FF
gold #FFD700	goldenrod #DAA520	gray #808080	green #008000
greenyellow #ADFF2F	honeydew #F0FFF0	hotpink #FF69B4	indianred #CD5C5C
indigo #4B0082	ivory #FFFFF0	khaki #F0E68C	lavender #E6E6FA
lavenderblush #FFF0F5	lawngreen #7CFC00	lemonchiffon #FFFACD	lightblue #ADD8E6
lightcoral #F08080	lightcyan #E0FFFF	lightgoldenrodyellow #FAFAD2	lightgreen #90EE90
lightgrey #D3D3D3	lightpink #FFB6C1	lightsalmon #FFA07A	lightseagreen #20B2AA
lightskyblue #87CEFA	lightslategray #778899	lightsteelblue #B0C4DE	lightyellow #FFFFE0
lime #00FF00	limegreen #32CD32	linen #FAF0E6	magenta #FF00FF
maroon #800000	mediumaquamarine #66CDAA	mediumblue #0000CD	mediumorchid #BA55D3
mediumpurple #9370DB	mediumseagreen #3CB371	mediumslateblue #7B68EE	mediumspringgreen #00FA9A

mediumturquoise #48D1CC	mediumvioletred #C71585	midnightblue #191970	mintcream #F5FFFA
mistyrose #FFE4E1	moccasin #FFE4B5	navajowhite #FFDEAD	navy #000080
oldlace #FDF5E6	olive #808000	olivedrab #6B8E23	orange #FFA500
orangered #FF4500	orchid #DA70D6	palegoldenrod #EEE8AA	palegreen #98FB98
paleturquoise #AFEEEE	palevioletred #DB7093	papayawhip #FFEFD5	peachpuff #FFDAB9
peru #CD853F	pink #FFC0CB	plum #DDA0DD	powderblue #B0E0E6
purple #800080	red #FF0000	rosybrown #BC8F8F	royalblue #4169E1
saddlebrown #8B4513	salmon #FA8072	sandybrown #F4A460	seagreen #2E8B57
seashell #FFF5EE	sienna #A0522D	silver #C0C0C0	skyblue #87CEEB
slateblue #6A5ACD	slategray #6A5ACD	snow #FFFAFA	springgreen #00FF7F
steelblue #4682B4	tan #D2B48C	teal #008080	thistle #D8BFD8
tomato #FF6347	turquoise #40E0D0	violet #EE82EE	wheat #F5DEB3
white #FFFFFF	yellow #FFFF00	yellowgreen #9ACD32	

Lecture 4

Today we will learn

- **Using the Font tag**
- **Align the text**

Font means a specific size and style of the text. Size means the width of the text, and style means whether the text is Arial, Times New Roman, etc.

Type the following code in notepad and save the file as .html.

```
<html>
    <head>
        <title>
            Using the Font tag
        </title>
    </head>
    <body>
    Here is normal text without any font tag. <br>
    <html>
    <head>
        <title>
            Using the Font tag
        </title>
    </head>
    <body>
    Here is normal text without any font tag. <br>
    <font face="arial" size="4"> This is a text with font type Arial and font size
4</font>
    </body>
</html>
    </body>
</html>
```

The output will be

Here is normal text without any font tag.

This is a text with font type Arial and font size 4

Q) Try the above code with font Batang, and print the output?

You can also specify the color with the font tag. i.e. color is another attribute of the font tag. For example

Type the following code in notepad and save the file as .html.

```
<html>
    <head>
        <title>
            Using the Font tag
        </title>
```

```
        </head>
        <body>
        Here is normal text without any font tag. <br>
        <html>
        <head>
                <title>
                        Using the Font tag
                </title>
        </head>
        <body>
        Here is normal text without any font tag. <br>
        <font face="arial" size="4" color=”blue”> This is a text with font type Arial and
    font size 4</font>
        </body>
    </html>
        </body>
</html>
```

The output will be
Here is normal text without any font tag.
This is a text with font type Arial and font size 4

Align the text

To align the text means whether the text will appear on the left of the page or right of the page or center of the page.

Type the following code in notepad and save as .html file.

```
<html>
        <head>
                <title>
                        Using the Align tag
                </title>
        </head>
        <body>
                My name is Dileep Kumar
                <p>I am a student of University of Suwon</p>
        </body>
</html>
```

The output will be
My name is Dileep Kumar

I am a student of University of Suwon

If there was no <p> tag, then two statements “My name is Dileep Kumar” and “I am a student of University of Suwon” would come one after the other on the same line. Whereas <p> tag inserted a new line between the two statements, doing some what the same as
 tag.

If you notice the output very carefully there is some difference between <p> tag and
 tag.

Q) What is the difference between the <p> tag and
 tag?

A) The difference is <p> tag inserts a double new line between two statements whereas
 tag inserts single new line between two statements. Actually <p> tag means paragraph tag, and <p> tag indicates the browser to start a new paragraph.

Now let us come back to aligning the text

Type the following code and save as .html file.

```
<html>
    <head>
        <title>
            Using the Align tag
        </title>
    </head>
    <body>
        My name is Dileep Kumar
        <p align="right">I am a student of University of Suwon</p>
    </body>
</html>
```

The output is

My name is Dileep Kumar

I am a student of University of Suwon

So, You can see that the second statement is aligned towards the right. Similarly the align attribute of <p> tag can be used as ‘left’ or ‘center’.

Q) Try the above code with <p align=”left”> and also with <p align=”center”>? Show what the output is?

Lecture 5

Today we will learn

- **Create a Link to new pages using <a href="">.**
- **Writing the text as a List of Items using <UL>, <LI>, <OL>.**

Creating a link to a new page mean that you make one web page which contains a link to another page and when you click on the link, another page opens. For example

Type the following code and save as first.html

```
<html>
<head>
        <title> First Page</title>
</head>
</body>
        My name is Dileep Kumar. <br>
        I am teaching at Suwon University <br>
        To know about my hobbies, <a href="hobby.html">click here</a>
</body>
</html>
```

Now what you will see in the output is, an underline below 'Click here' and the color will be blue. This is a link, which links to a new page, and all the links appear as blue color i.e.
The output will be

My name is Dileep Kumar.
I am teaching at Suwon University
To know about my hobbies click here

When you will left click mouse on the link 'click here', a new web-page hobby.html should open. But as such hobby.html is not created. So we need to create this page also. If clicking on a link displays 'The link cannot be displayed', then such a link is called as broken link. A web-site should not contain broken link.

Now you can type the code below and save as hobby.html, remember that hobby.html should be saved in the same directory which contains first.html.

```
<html>
<head>
        <title>This is my hobby page</title>
</head>
<body>
        <b>My hobbies are</b><br>
                <p align="center">Football</p>
                <p align="center">Listening Music </p>
                <p align="center">Reading</p>
</body>
</html>
```

After clicking on the ‘click here’, a new page hobby.html will open, and the output will be

My hobbies are

Football

Listening Music

Reading

The <a href=””> is also called as **anchor tag**.

Q) What will happen if the <a> tag is used as follows, and use click on “click here”
 click here ?
A) The same page will be refreshed again.

Q) I told you that both the html files i.e. first.html and hobbies.html should be in the same directory, what if both of them are not in the same directory?
A) The answer to this is relative addressing, which we will discuss in later lecture, but right now remember the word relative addressing.

I listed above my hobbies; there is a tag which helps in listing. The name of the tag is <UL> tag. <UL> tag is also known as Un-ordered List tag.

Write the code below which does not use <UL> tag, and save as hobby.html.

```
<html>
<head>
        <title>This is my hobby page</title>
</head>
<body>
        <b>My hobbies are</b><br>

                <p>Football</p>
                <p>Listening Music </p>
                <p>Reading</p>
</body>
</html>
```

After clicking on the ‘click here’, a new page hobby.html will open, and the output will be
My hobbies are

Football

Listening Music

Reading

So, there is not proper **indentation**. The hobbies football, listening music, reading should appear towards the right.

The <UL> tag will do the required indentation.

```
<html>
<head>
	<title>This is my hobby page</title>
</head>
<body>
	<b>My hobbies are</b><br>
	<ul>
		<p>Football</p>
		<p>Listening Music </p>
		<p>Reading</p>
	</ul>

</body>
</html>
```

And, now the output is

My hobbies are

- Football
- Listening Music
- Reading

So, now the hobbies football, Listening music, Reading are indented towards the right.

I can further improve upon the hobbies listing.

Q) How can I have a bullet (dark solid circle) before each hobby?
A) There is different tag called as <LI> which is used with the <UL> tag.

Type the code below in notepad and save as hobby.html

```
<html>
<head>
	<title>This is my hobby page</title>
</head>
<body>
	<b>My hobbies are</b><br>
	<ul>
		<li>Football</li>
		<li>Listening Music </li>
```

```
            <li>Reading</li>
        </ul>

</body>
</html>
```

So, the output is
My hobbies are

- Football
- Listening Music
- Reading

What I have done, I have replaced the <p> tag with the <li> tag, still keeping the <ul> tag.

Q) What if I want numbers instead of bullets?
A) For that, I have to replace <UL> with <OL>.

Please type the code below in notepad and save as hobby.html

```
<html>
<head>
        <title>This is my hobby page</title>
</head>
<body>
        <b>My hobbies are</b><br>
        <ol>
                <li>Football</li>
                <li>Listening Music </li>
                <li>Reading</li>
        </ol>

</body>
</html>
```

The output will be
My hobbies are

1. Football
2. Listening Music
3. Reading

We can also do nesting of lists, for details see below.

Nested Lists

Write the following code and save as .html file

```
<html>
<head>
	<title>Country visit</title>
</head>
<body>
	<b>The countries and cities that I have visited are</b><br>
	<UL>
		<li>Oman</li>
			<UL>
				<li>Muscat</li>
				<li>Salalah</li>
				<li>Nizwa</li>
			</UL>
		<li>South korea</li>
			<ul>
				<li>Seoul</li>
				<li>Suwon</li>
				<li>incheon</li>
			</Ul>

	</UL>

</body>
</html>
```

The output will be

The countries and cities that I have visited are

- Oman
  - Muscat
  - Salalah
  - Nizwa
- South korea
  - Seoul
  - Suwon
  - incheon

Oman and South Korea are countries. Muscat, Salalah, Nizwa are cities in Oman whereas Seoul, Suwon, Incheon are cities in South Korea.

----------Finish----------

Lecture 6

Today we will learn about

- **Create an Email link and External link.**
- **Insert graphics.**
- **Link to specific section in another document.**

Create an Email link

When you visit a web site, generally at the bottom you see a link for mail to the creator of the website. The subject of today's lecture is how to create such link. Such a link which on clicking opens an email client is called as email link.

Please type the following code in notepad and save as .html file.

```
<html>
<head>
        <title>mail</title>
</head>
<body>
        My name is Dileep Kumar. <br>
        I am studying at University of Suwon <br>
        To mail me, Please <a href="mailto:dileep02sw25@yahoo.com">Click here</a>
</body>
</html>
```

The output will be

My name is Dileep Kumar.
I am studying at University of Suwon
To mail me, Please Click here

When you will click on "Click here", mail client will open, through which you can send email.

Create an External Link

When you visit a web site, you find links to other websites. When you click on the link, another web site opens. Such a link which on clicking opens another website is called as external link

Please type the following code in notepad and save as .html file.

```
<html>
<head>
        <title>mail</title>
</head>
<body>
        My name is Dileep Kumar. <br>
        I am studying at University of Suwon <br>
```

```
The sites I visit for fun are
        <ol>
                <li><a href="http://www.yahoo.com">Yahoo</a></li>
                <li><a href="http://www.hotmail.com">Hotmail</a></li>
        </ol>
        To mail me, Please <a href="mailto:dileep02sw25@yahoo.com">Click here</a>
</body>
</html>
```

The output will be

My name is Dileep Kumar.
I am studying at University of Suwon
The sites I visit for fun are

1. Yahoo
2. Hotmail

To mail me, Please Click here

So, when you will click on "Yahoo", yahoo website will open, and if you will click on "Hotmail", Hotmail website will open.

Insert Graphics

Most of the websites that you visit contain graphics along with text. So graphics need to be inserted in a web page.

To understand this, you need an image file, please download an image file from the internet or locate for an image file on your computer. I have got an image file by the name of "asia.gif".

```
<html>
<head>
        <title>mail</title>
</head>
<body>
        My name is Dileep Kumar. <br>
        I am studying at University of Suwon <br>
        Below is the map of Asia<br>
        <img src="asia.gif" alt="map of asia">
</body>
</html>
```

And the output is

My name is Dileep Kumar.
I am studying at University of Suwon
Below is the map of Asia

Link to specific section in another document

In lecture5 we discussed about, how to make a link from one document to another document. Now we will learn about how to make a link to specific part inside a document.

Let's take the previous example

Type the following code and save as first.html

```
<html>
<head>
        <title> First Page</title>
</head>
</body>
        My name is Dileep Kumar. <br>
        I am teaching at Suwon University <br>
        My hobbies are <br>
        <a href="hobbies.html#soccer">Soccer</a><br>
        <a href="hobbies.html#music">Music</a><br>
```

```
        <a href="hobbies.html#read">Reading</a><br>


</body>
</html>
```

And type the following code and save as hobbies.html

```
<html>
<head>
        <title>This is my hobby page</title>
</head>
<body>
        <b>My hobbies are</b><br>
<a name="music">            <p align="center">Listenin Music</p></a>
                <p>
                        I like listening to music a lot.<br>
                        I like rock, instrumental music.<br>
                        My favorites are <br>
                        <ul>
                                <li>Michael Jackson</li>
                                <li>Aerosmith</li>
                                <li>Savage Garden</li>
                                <li>Celin Deon</li>
                                <li>Guns & Roses</li>
                                <li>Britny Spears</li>
                                <li>50 Cents</li>
                                <li>Richard Marx</li>
                                <li>Elton John</li>
                                <li>Bryan Adams</li>
                                <li>All Saints</li>
                                <li>Backstreet Boys</li>
                                <li>Bon Jovi </li>
                                <li>Eminem</li>
                                <li>Enrique Iglesias</li>
                                <li>Eric Clapton</li>
                                <li>George Michael</li>
                                <li>Faithless</li>
                                <li>Queeb</li>
                                <li>Cher</li>
                                <li>Janet Jackson</li>
                                <li>Jennifer Lopez</li>
                                <li>Jewel</li>
                                <li>Kylie Minogue</li>

                        </ul>
```

```
                </p>
        <a name="read">       <p align="center">Reading </p></a>
My favorites are <br>
                        <ul>
                                <li>Computer Books</li>
                                <li>Fiction</li>
                                <li>Non-fiction</li>
                                <li>Science</li>
                                <li>Newspaper</li>
                        </ul>

                <a name="soccer"><p align="center">Soccer</p></a>
                <ul>
                        My favorite players are
                                <li>David Batty, Leeds United </li>
                                <li>Marcus Bent, Ipswich Town </li>
                                <li>Craig Burley, Derby County </li>
                                <li>Mike Salmon, Ipswich Town </li>
                                <li>David Seaman, Arsenal </li>
                                <li>Craig Short, Blackburn Rovers </li>
                                <li>Dean Windass, Middlesbrough</li>

                </ul>
</body>
</html>
```

Now when you will click on “soccer”, hobbies.html will open and scroll down, so that you can see the details about soccer. Similarly, when you click on “music” , hobbies.html will open and if required, page will scroll down, so that you can see the details about music.

Links to Specific Sections within the Same Document

The above concept can also be used for linking a specific section with in the same document. For example.

```
<html>
<head>
        <title>This is my hobby page</title>
</head>
<body>
        <b>My hobbies are</b><br>
        <a href="#music">Listening Music</a><br>
        <a href="#read">Reading</a><br>
        <a href="#soccer">Soccer</a><br>
```

```
<a name="music"><p align="center">Listenin Music</p></a>
        <p>
                I like listening to music a lot.<br>
                I like rock, instrumental music.<br>
                My favorites are <br>
                <ul>
                        <li>Michael Jackson</li>
                        <li>Aerosmith</li>
                        <li>Savage Garden</li>
                        <li>Celin Deon</li>
                        <li>Guns & Roses</li>
                        <li>Britny Spears</li>
                        <li>50 Cents</li>
                        <li>Richard Marx</li>
                        <li>Elton John</li>
                        <li>Bryan Adams</li>
                        <li>All Saints</li>
                        <li>Backstreet Boys</li>
                        <li>Bon Jovi </li>
                        <li>Eminem</li>
                        <li>Enrique Iglesias</li>
                        <li>Eric Clapton</li>
                        <li>George Michael</li>
                        <li>Faithless</li>
                        <li>Queeb</li>
                        <li>Cher</li>
                        <li>Janet Jackson</li>
                        <li>Jennifer Lopez</li>
                        <li>Jewel</li>
                        <li>Kylie Minogue</li>

                </ul>


        </p>
    <a name="read">     <p align="center">Reading </p></a>
My favorites are <br>
                <ul>
                        <li>Computer Books</li>
                        <li>Fiction</li>
                        <li>Non-fiction</li>
                        <li>Science</li>
                        <li>Newspaper</li>
                </ul>
```

```
<a name="soccer"><p align="center">Soccer</p></a>
<ul>
        My favorite players are
                <li>David Batty, Leeds United </li>
                <li>Marcus Bent, Ipswich Town </li>
                <li>Craig Burley, Derby County </li>
                <li>Mike Salmon, Ipswich Town </li>
                <li>David Seaman, Arsenal </li>
                <li>Craig Short, Blackburn Rovers </li>
                <li>Dean Windass, Middlesbrough</li>

</ul>
</body>
</html>
```

Lecture 7

In this lecture, we will learn about

- **Using an Image as a Link**
- **Aligning Images**

Using an Image as a Link

I am using three images, for my hobby page. Type the following code and save as .html file.

```
<html>
<head>
        <title> First Page</title>
</head>
</body>
        My name is DILEEP Kumar. <br>
        I am teaching at Suwon University <br>
        My hobbies are <br>
        <a href="hobbies.html#soccer"><img src="soccer.gif" alt=soccer></a><br>
        <a href="hobbies.html#music"><img src="music.gif" alt=music></a><br>
        <a href="hobbies.html#read"><img src="book.gif" alt=book></a><br>
</body>
</html>
```

The above code is similar to the previous code, but the only difference is instead of text, now there are images. Earlier clicking on text opens a new page, but now clicking on link will open a new page.

Aligning Images

With the <img src= > tag, you can specify the width and height of the image i.e.
<img src=” “ width=” “ height=” “>

```
<html>
<head>
        <title> First Page</title>
</head>
</body>
        My name is DILEEP Kumar. <br>
        I am teaching at Suwon University <br>
        My hobbies are <br>
        <a href="hobbies.html#soccer"><img src="soccer.gif" alt=soccer width=100
height=100></a><br>
        <a href="hobbies.html#music"><img src="music.gif" alt=music width=100
height=100></a><br>
        <a href="hobbies.html#read"><img src="book.gif" alt=book width=100
height=100></a><br>
</body>
</html>
```

The above code will reduce the size of the image, and make the web page look more sensible.

You have some flexibility when displaying images. You can have images separated from text and aligned to the left or right or centered. Or you can have an image aligned with text. Try several possibilities to see how your information looks best.

You can align images to the top or center of a paragraph using the **ALIGN=** attributes **TOP** and **MIDDLE.** i.e.

<img src=”” align=”top/middle”>

Let us take an example, and see how image and text look without align attribute and with align attribute. First take an example of how image and text look without align attribute.

```
<html>
<head>
        <title> First Page</title>
</head>
</body>
        My name is DILEEP Kumar. <br>
        I am teaching at Suwon University <br>
        My hobbies are <br>
        <a href="hobbies.html#soccer"><img src="soccer.gif" alt=soccer width=100
height=100></a>Soccer<br>
        <a href="hobbies.html#music"><img src="music.gif" alt=music width=100
height=100> </a>Music<br>
        <a href="hobbies.html#read"><img src="book.gif" alt=book width=100
height=100></a>Book<br>
</body>
</html>
```

So, the output is

My name is DILEEP Kumar.
I am teaching at Suwon University
My hobbies are

Soccer

MusicMusic

Book

The output shows that, the text is at the bottom of the image, whereas it looks better to be text in the middle of the image or at the top of the image.

Let us take another example, and see how text and image text and image look with align attribute. We have used align=middle.

```
<html>
<head>
        <title> First Page</title>
</head>
</body>
        My name is DILEEP Kumar. <br>
        I am teaching at Suwon University <br>
        My hobbies are <br>
        <a href="hobbies.html#soccer"><img src="soccer.gif" alt=soccer width=100
height=100

align=middle></a>Soccer<br>
        <a href="hobbies.html#music"><img src="music.gif" alt=music width=100
height=100 align=middle></a>Music<br>
        <a href="hobbies.html#read"><img src="book.gif" alt=book width=100
height=100 align=middle></a>Book<br>
</body>
</html>
```

The output can be seen on the browser.

Lecture 8

In this lecture, we will learn about

- **How to create Table**
- **Escape Sequences: Special Characters**

Tables are very useful while creating HTML pages; especially tables solve the alignment problem. A table tag has headings where you explain what the columns/rows include, rows for information, and cells for each item. In the following table, the first two rows contain the heading information, each detail row explains an HTML table tag, and each cell contains a paired tag or an explanation of the tag's function.

Table Elements	
Element	**Description**
<TABLE> ... </TABLE>	defines a table in HTML. If the BORDER attribute is present, your browser displays the table with a border.
<CAPTION> ... </CAPTION>	defines the caption for the title of the table. The default position of the title is centered at the top of the table. The attribute ALIGN=BOTTOM can be used to position the caption below the table. **NOTE:** Any kind of markup tag can be used in the caption.
<TR> ... </TR>	specifies a table row within a table. You may define default attributes for the entire row: ALIGN (LEFT, CENTER, RIGHT) and/or VALIGN (TOP, MIDDLE, BOTTOM).
<TH> ... </TH>	defines a table header cell. By default the text in this cell is bold and centered. Table header cells may contain other attributes to determine the characteristics of the cell and/or its contents.
<TD> ... </TD>	defines a table data cell. By default the text in this cell is aligned left and centered vertically. Table data cells may contain other attributes to determine the characteristics of the cell and/or its contents.

```
Here is the single cell table, the simplest table possible.

<CENTER>

        <table border=1>

                <tr>

                     <td>here is a single-celled table!</td>

                </tr>
        </table>

</CENTER>
```

The output on the browser is

here is a single-celled table!

```
<html>
	<head>
		<title> Using Table Tag</title>
	</head>

	<body>
		<table border=1>
			<caption> Time Table for Mar-June 2005</caption>
			<tr>
				<th></th>
				<th>Monday</th>
				<th>Tuesday</th>
				<th>Wednesday</th>
				<th>Thrusday</th>
				<th>Friday</th>
			</tr>

			<tr>
				<td>9:30-10:20</td>
				<td>-</td>
				<td>-</td>
				<td>-</td>
				<td>-</td>
				<td><font color="blue">109</font></td>
			</tr>

			<tr>
				<td>10:30-11:20</td>
				<td>-</td>
				<td>-</td>
				<td>-</td>
				<td>-</td>
				<td><font color="blue">109</font></td>
			</tr>

			<tr>
				<td>11:30-12:20</td>
				<td>-</td>
				<td>-</td>
				<td>-</td>
```

```
        <td>-</td>
        <td><font color="blue">109</font></td>
</tr>

<tr>
        <td>12:30-1:20</td>
        <td>-</td>
        <td>-</td>
        <td><font color="blue">312</font></td>
        <td>-</td>
        <td>-</td>
</tr>

<tr>
        <td>1:30-2:20</td>
        <td>-</td>
        <td><font color="blue">306</font></td>
        <td>-</td>
        <td>-</td>
        <td>-</td>
</tr>

<tr>
        <td>2:30-3:20</td>
        <td>-</td>
        <td><font color="blue">306</font></td>
        <td>-</td>
        <td><font color="blue">309</font></td>
        <td>-</td>
</tr>

<tr>
        <td>3:30-4:20</td>
        <td>-</td>
        <td>-</td>
        <td>-</td>
        <td><font color="blue">309</font></td>
        <td>-</td>
</tr>

<tr>
        <td>4:30-5:20</td>
        <td>-</td>
        <td>-</td>
        <td>-</td>
        <td><font color="blue">309</font></td>
```

```
                    <td>-</td>
                </tr>
            <table>
        </body>
</html>
```

Time Table for Mar-June 2005					
	Monday	**Tuesday**	**Wednesday**	**Thrusday**	**Friday**
9:30-10:20	-	-	-	-	109
10:30-11:20	-	-	-	-	109
11:30-12:20	-	-	-	-	109
12:30-1:20	-	-	312	-	-
1:30-2:20	-	306	-	-	-
2:30-3:20	-	306	-	309	-
3:30-4:20	-	-	-	309	-
4:30-5:20	-	-	-	309	-

Escape Sequences: Special Characters (a.k.a. Character Entities)

Special characters are composed of short sequence of characters that are translated by the browser and displayed as a single character.
Special characters begin with an ampersand(&) and end with a semicolon (;), for example , which is used for giving spaces between words or characters

To understand   type the following code in notepad

```
<html>
    <head>
        <title>Using nbsp </title>
    </head>

    <body>
        My name is Dileep            Kumar.
    </body>
</html>
```

The output is
My name is Dileep Kumar.

Although I gave so many spaces between “Harhsit” and “Kumar”, still the output shows only one space.

This means that browser shrink the spaces, so to insert spaces, we need .

Type the following code in notepad

```
<html>
        <head>
                <title>Using nbsp </title>
        </head>

        <body>
        My name is Dileep       Kumar.
        </body>
</html>
```

The output is

My name is Dileep   Kumar.

----------Finish----------

Lecture 9

In this lecture, we will learn about

- Frames

What are Frames?

Frames are a way to put one more than one page at a time on a computer screen. Usually frames are made, to add a menu on one side, and the corresponding pages on the other side by clicking each option of the menu.

index.htm now just instructs the browser, by frame and frameset tags, to split the browser window into two or more parts. Each new part, called a "frame", has a full fledged HTML file loaded within it. In this example, menu_bar.htm is loaded in the left, and main.htm is loaded into the right.

Here is the code for the above

```
<html>
<head><title>Using Frames</title></head>

<frameset cols="15%,85%">
 <frame src="menu_bar.htm" name="sidemenu">
 <frame src="main.htm" name="mainwindow">
</frameset>
```

Note that there is no actual body coding involved when creating frames.

About <frameset> and <frame>

The frameset tag is used to declare multiple frames. As you can see in the above example, there was one frameset, and it reads as

<frameset cols="15%,85%">

This tells the browser that we are creating column of framed pages, in this case two frames, each in one column. The first one will take the 15% of the total browser screen and the second one will take 85% of the total browser screen.

After that we use frame tag, which actually loads the desired web page. Each frame must have a source, such as src=”webpage.html”. So, because we used two framed areas within the frameset, we need two frame tags, each of them to load a page.

```
<frameset cols="15%,85%">
 <frame src="menu_bar.htm" name="sidemenu">
 <frame src="main.htm" name="mainwindow">
</frameset>
```

If we would like to add a third column, we would need to add a third size definition in the cols (so that all would add up to `100%`) and another frame tag inside the frameset.

In the above case we had a vertical menu bar; we can also have a horizontal menu bar. For that we have to use rows instead of cols. For example

```
<frameset rows="15%,85%">
 <frame src="menu_bar.htm" name="sidemenu">
 <frame src="main.htm" name="mainwindow">
</frameset>
```

Linking with Frames

Now, let us take an example which you have been using, yes it is your course URL i.e. http://kumarharmuscat.tripod.com/iprogram. I am using frames in that.

```
<html>

<head>

        <title>Internet Programming</title>

</head>

        <frameset cols="20%,80%">

                <frame name="lp" src="leftpane.html">

                <frame name="rp" src="ip.html">

        </frameset>

</html>
```

So, in the above code, I have divided the whole webpage into two frames, first frame contains the menu and is on the left, second frame load the menu link.

Let us look at the coding of leftpane.html

```
<html>

<body>

<B>Intenet Programming</B>

<font size=2 type="arial">

<a href="ipcontents.html" target="rp">Course Contents</a><br>

<a href="ethics.html" target="rp">Code Of Ethics</a><br>

<a href="qb.html" target="rp">Assignment</a><br>

<a href="schedule.html" target="rp">Lecture Notes</a><br>

<a href="project.html" target="rp">Project</a><br>

</body>

</html>
```

Notice, that there is no head tag here. It is quite obvious that I don't need a head tag now, because this webpage is contained inside another webpage.

The output of this web page is

Intenet Programming

Course Contents
Grading Policy
Assignment
Lecture Notes
Project

So, when you click on "Course Contents", your course content opens, when you click on Grading policy, the page corresponding to grading policy i.e. "ethics.html" opens and same is the case with other links.

Q How come when you click on the menu option, the page opens in the second frame?
A) This is possible through the target attribute of anchor tag.

Notice two things, when I created frames, I gave a name to each frame.

```
<html>

<head>

	<title>Internet Programming</title>

</head>

	<frameset cols="20%,80%">

		<frame name="lp" src="leftpane.html">

		<frame name="rp" src="ip.html">

	</frameset>

</html>
```

So, the name of frame on the left is “lp”, and the name of frame on the right is “rp”.

If I don’t specify target attribute in the anchor tag, then on clicking the menu option, the page will open in the first frame on the left, and nothing will open in the second frame. So, I specify where I want the page to open.

There are other possibilities with the target attribute of anchor tag.

- **`target="_blank"`** - link is loaded into a new blank browser window (and your old window stays open).
- **`target="_self"`** - link is loaded into frame that link was clicked in. (this is the default selection. Leave it out if you so please.)
- **`target="_top"`** - link is loaded into current full browser window, and all frames disappear, leaving the new linked page to occupy the entire window.

Special attributes of `<frame>`

Q) Can I change the size of the frame using my mouse, I mean taking my mouse over the vertical bar, then left click and dragging it to the left or right?
A) Yes

Q) What if I don’t want the user to drag the vertical boundary?
A) The solution is below

```
<html>
```

```
<head>

        <title>Internet Programming</title>

</head>

        <frameset cols="20%,80%">

                <frame name="lp" src="leftpane.html" noresize>

                <frame name="rp" src="ip.html">

        </frameset>

</html>
```

The other useful attribute is **`scrolling`**. Say you always want a scrollbar to appear in the main window. Add `scrolling="yes"`. Want there to never be a scrollbar? Add `scrolling="no"`.

To understand more about scrolling, type the following code

```
<html>
<head>
        <title>Internet Programming</title>
</head>
        <frameset cols="20%,80%">
                <frame name="lp" src="leftpane.html">
                <frame name="rp" src="ip.html" scrolling="yes">
        </frameset>
</html>
```

More Advanced Frames

Let us discuss move advanced frames now, where we will see how to divide the browser window into more than two frames. Take an example first

```
<frameset cols="15%,85%">
 <frameset rows="20%,80%">
   <frame src="timetable.html">
   <frame src="hobbies.html" name="sidemenu">
 </frameset>
 <frame src="first.html" name="mainwindow">
</frameset>
```

In the above code, browser window is divided into two frames vertically, first frame is 15% of the total size of the browser, and the second frame is 85% of the total size of the browser. Then again the first frame is divided into two more frames, i.e. the frame which occupied 15% is again divided into two frames horizontally, and the ratio is 20% and 80%.

```
<frameset rows="50%,50%">
 <frameset cols="50%,50%">
   <frame>
   <frame>
 </frameset>
 <frameset cols="50%,50%">
   <frame>
   <frame>
 </frameset>
```

The above code will divide the browser window into four frames, each of equal size.

```
<frameset cols="33%,33%,33%">
  <frame>
  <frame>
  <frame>
</frameset>
```

The above code will divide the browser window vertically into three frames each occupying 33% of the total browser window size.

```
<frameset rows="25%,50%,25%">
 <frame>
 <frame>
 <frame>
</frameset>
```

The above code will divide the browser window horizontally into three frame, first frame occupies the 25% of the total browser window, second frame occupies the 50% of the total browser window, and third frame occupies the 25% of the total browser window.

----------Finish----------

Lecture 10

- **More about body tag- Setting up link colors**
- **More about body tag- Setting up Background Image**
- **Meta tags**

In previous lecture, you learned the `BODY` tag. The `BODY` tag has many attributes... here are the most useful ones...

- `BACKGROUND="location_of_image"` - Background image for web page. Example: If you have kitten.jpg in the same directory as your HTML file, use `<body background="kitten.jpg">` to load it as your background image.

- `BGCOLOR="name of color"` – Sets the Background Color for the web page

- `LINK="name of color"` - Color Code for Links (if left blank, most browsers default to blue.)

- `VLINK="name of color"` - Code for Links the User has Already Visited (if left blank, most browsers default to purple.)

- `TEXT="name of color"`

More About Table Tag

TWO DEMONSTRATIONS OF ROWSPAN

Item 1	Item 2	Item 3
Item 4		Item 5

```
<TABLE BORDER>
        <TR>
                <TD>Item 1</TD>
                <TD ROWSPAN=2>Item 2</TD>
                <TD>Item 3</TD>
        </TR>
```

```
        <TR>
                <TD>Item 4</TD> <TD>Item 5</TD>
        </TR>
</TABLE>
```

Item 1	Item 2	Item 3	Item 4
	Item 5	Item 6	Item 7

```
<TABLE BORDER>
        <TR>
                <TD ROWSPAN=2>Item 1</TD>
                <TD>Item 2</TD> <TD>Item 3</TD> <TD>Item 4</TD>
        </TR>
        <TR>
                <TD>Item 5</TD> <TD>Item 6</TD> <TD>Item 7</TD>
        </TR>
</TABLE>
```

DEMONSTRATION OF COLSPAN

Item 1	Item 2	
Item 3	Item 4	Item 5

```
<TABLE BORDER>
        <TR>
                <TD>Item 1</TD>
                <TD COLSPAN=2>Item 2</TD>
        </TR>
        <TR>
                <TD>Item 3</TD> <TD>Item 4</TD> <TD>Item 5</TD>
        </TR>
</TABLE>
```

DEMONSTRATION OF HEADERS, <TH>

Head1	Head2	Head3
A	B	C
D	E	F

```
<TABLE BORDER>
        <TR>
                <TH>Head1</TH> <TH>Head2</TH> <TH>Head3</TH>
        </TR>
        <TR>
                <TD>A</TD> <TD>B</TD> <TD>C</TD>
        </TR>
        <TR>
                <TD>D</TD> <TD>E</TD> <TD>F</TD>
        </TR>
</TABLE>
```

DEMONSTRATION OF COLSPAN AND HEADERS,

Head1		Head2	
A	B	C	D
E	F	G	H

```
<TABLE BORDER>
        <TR>
                <TH COLSPAN=2>Head1</TH>
                <TH COLSPAN=2>Head2</TH>
        </TR>
        <TR>
                <TD>A</TD> <TD>B</TD> <TD>C</TD> <TD>D</TD>
        </TR>
        <TR>
                <TD>E</TD> <TD>F</TD> <TD>G</TD> <TD>H</TD>
        </TR>
</TABLE>
```

DEMONSTRATION OF MULTIPLE HEADERS, COLSPAN

Head1		Head2	
Head 3	Head 4	Head 5	Head 6
A	B	C	D
E	F	G	H

```
<TABLE BORDER>
        <TR>
                <TH COLSPAN=2>Head1</TH>
                <TH COLSPAN=2>Head2</TH>
        </TR>
        <TR>
                <TH>Head 3</TH> <TH>Head 4</TH>
                <TH>Head 5</TH> <TH>Head 6</TH>
        </TR>
        <TR>
                <TD>A</TD> <TD>B</TD> <TD>C</TD> <TD>D</TD>
        </TR>
        <TR>
                <TD>E</TD> <TD>F</TD> <TD>G</TD> <TD>H</TD>
        </TR>
</TABLE>
```

DEMONSTRATION OF SIDE HEADERS

Head1	Item 1	Item 2	Item 3
Head2	Item 4	Item 5	Item 6
Head3	Item 7	Item 8	Item 9

```
<TABLE BORDER>
        <TR><TH>Head1</TH>
                <TD>Item 1</TD> <TD>Item 2</TD> <TD>Item 3</TD></TR>
        <TR><TH>Head2</TH>
                <TD>Item 4</TD> <TD>Item 5</TD> <TD>Item 6</TD></TR>
        <TR><TH>Head3</TH>
                <TD>Item 7</TD> <TD>Item 8</TD> <TD>Item 9</TD></TR>
</TABLE>
```

DEMONSTRATION OF SIDE HEADERS, ROWSPAN

Head1	Item 1	Item 2	Item 3	Item 4
	Item 5	Item 6	Item 7	Item 8
Head2	Item 9	Item 10	Item 3	Item 11

```
<TABLE BORDER>
        <TR><TH ROWSPAN=2>Head1</TH>
            <TD>Item 1</TD> <TD>Item 2</TD> <TD>Item 3</TD> <TD>Item
4</TD>
        </TR>
        <TR><TD>Item 5</TD> <TD>Item 6</TD> <TD>Item 7</TD> <TD>Item
8</TD>
        </TR>
        <TR><TH>Head2</TH>
            <TD>Item 9</TD> <TD>Item 10</TD> <TD>Item 3</TD> <TD>Item
11</TD>
        </TR>
</TABLE>
```

SAMPLE TABLE USING ALL OF THESE

```
<TABLE BORDER>
        <TR>    <TD>
                <TH ROWSPAN=2></TH>
                <TH COLSPAN=2>Average</TH>
                </TD>
        </TR>
        <TR>    <TD><TH>Height</TH><TH>Weight</TH></TD>
        </TR>
        <TR>    <TH ROWSPAN=2>Gender</TH>
                <TH>Males</TH><TD>1.9</TD><TD>0.003</TD>
```

```
        </TR>
        <TR>    <TH>Females</TH><TD>1.7</TD><TD>0.002</TD>
        </TR>
</TABLE>
```

CLEVER USES OF ROWSPAN/COLSPAN

A	1	2
	3	4
C	D	

```
<TABLE BORDER>
        <TR>
                <TD ALIGN=center ROWSPAN=2 COLSPAN=2>A</TD>
                <TD>1</TD>
                <TD>2</TD>
        </TR>
        <TR>
                <TD>3</TD>
                <TD>4</TD>
        </TR>
        <TR>
                <TD ALIGN=center ROWSPAN=2 COLSPAN=2>C</TD>
                <TD ALIGN=center ROWSPAN=2 COLSPAN=2>D</TD>
        </TR>
        <TR>
        </TR>
</TABLE>
```

ADJUSTING MARGINS AND BORDERS

A TABLE WITHOUT BORDERS

Item 1	Item 2	Item 3
Item 4		Item 5

```
<TABLE>
        <TR>    <TD>Item 1</TD> <TD ROWSPAN=2>Item 2</TD> <TD>Item
3</TD>
        </TR>
        <TR>    <TD>Item 4</TD> <TD>Item 5</TD>
        </TR>
</TABLE>
```

A TABLE WITH A BORDER OF 10

Item 1	Item 2
Item 3	Item 4

```
<TABLE BORDER=10>
        <TR>    <TD>Item 1</TD> <TD> Item 2</TD>
        </TR>
        <TR>    <TD>Item 3</TD> <TD>Item 4</TD>
        </TR>
</TABLE>
```

CELLPADDING AND CELLSPACING

A	B	C
D	E	F

```
<TABLE BORDER CELLPADDING=10 CELLSPACING=0>
        <TR>
                <TD>A</TD> <TD>B</TD> <TD>C</TD>
        </TR>
        <TR>
                <TD>D</TD> <TD>E</TD> <TD>F</TD>
        </TR>
</TABLE>
```

Cellpadding tag is used to create space between the text inside your table and the border surrounding the text.

A	B	C
D	E	F

Cellspacing tag is used to create space between different cells within your table.

```
<TABLE BORDER CELLPADDING=0 CELLSPACING=10>
        <TR>
                <TD>A</TD> <TD>B</TD> <TD>C</TD>
        </TR>
        <TR>
                <TD>D</TD> <TD>E</TD> <TD>F</TD>
        </TR>
</TABLE>
```

A	B	C

D	E	F

```
<TABLE BORDER CELLPADDING=10 CELLSPACING=10>
        <TR>
                <TD>A</TD> <TD>B</TD> <TD>C</TD>
        </TR>
        <TR>
                <TD>D</TD> <TD>E</TD> <TD>F</TD>
        </TR>
</TABLE>
```

A	B	C
D	E	F

```
<TABLE BORDER=5 CELLPADDING=10 CELLSPACING=10>
        <TR>
                <TD>A</TD> <TD>B</TD> <TD>C</TD>
        </TR>
        <TR>
                <TD>D</TD> <TD>E</TD> <TD>F</TD>
        </TR>
</TABLE>
```

ALIGNMENT, CAPTIONS, AND SUBTABLES

DEMONSTRATION OF MULTIPLE LINES IN A TABLE

January	February	March
This is cell 1	Cell 2	Another cell, cell 3
Cell 4	and now this is cell 5	Cell 6

```
<TABLE BORDER>
        <TR>
                <TH>January</TH>
                <TH>February</TH>
                <TH>March</TH>
        </TR>
        <TR>
                <TD>This is cell 1</TD>
                <TD>Cell 2</TD>
                <TD>Another cell,<br> cell 3</TD>
        </TR>
        <TR>
                <TD>Cell 4</TD>
```

```
                <TD>and now this<br>is cell 5</TD>
                <TD>Cell 6</TD>
        </TR>
</TABLE>
```

ALIGN=LEFT|RIGHT|CENTER

can be applied to individual cells or whole ROWs

January	February	March
all aligned center	Cell 2	Another cell, cell 3
aligned right	aligned to center	default, aligned left

```
<TABLE BORDER>
        <TR>
            <TH>January</TH>
            <TH>February</TH>
            <TH>March</TH>
        </TR>
        <TR ALIGN=center>
            <TD>all aligned center</TD>
            <TD>Cell 2</TD>
            <TD>Another cell,<br> cell 3</TD>
        </TR>
        <TR>
            <TD ALIGN=right>aligned right</TD>
            <TD ALIGN=center>aligned to center</TD>
            <TD>default,<br>aligned left</TD>
        </TR>
</TABLE>
```

VALIGN=TOP|BOTTOM|MIDDLE

can be applied to individual cells or whole ROWs

January	February	March
all aligned to top	and now this is cell 2	Cell 3
aligned to the top	aligned to the bottom	default alignment, center

```
<TABLE BORDER>
        <TR>
                <TH>January</TH>
```

```
            <TH>February</TH>
            <TH>March</TH>
        </TR>
        <TR VALIGN=top>
            <TD>all aligned to top</TD>
            <TD>and now this<br>is cell 2</TD>
            <TD>Cell 3</TD>
        </TR>
        <TR>
            <TD VALIGN=top>aligned to the top</TD>
            <TD VALIGN=bottom>aligned to the bottom</TD>
            <TD>default alignment,<br>center</TD>
        </TR>
</TABLE>
```

Meta Tag

The **<META>** tag can be used for a few different purposes. META tag are used in search engines. When a search engine finds your page, it adds your page to the searchable database with some information. The information that it adds is actually information contained in meta tag.

```
<meta name="description" content="description of page goes here">
<meta name="keywords" content="keywords go here">
```

Meta tags are not visible in the web page unless the user selects to 'view source'.

```
<html>

<head>
<title>Using Meta Tags </title>
<meta name="description" content="Joe's Collection of Cool Sound files
for you to use in your home page!">
<meta name="keywords" content="music sounds midi wav joe collection">
</head>

<body>
Page Goes Here
</body>

</html>
```

Lecture 11

In this lecture, we will learn about

- **Using Forms**

This topic is very important; actually this topic is the base for web programming. Forms are used for taking input from the user. After receiving input from the user, that information is generally stored in database, and for storing information in the database, either CGI, or ASP, or JSP, or some other scripting language is used. So after this, we can study ASP also.

So, whatever you will learn in this chapter, will be used later when you will learn ASP.

There is a tag for making forms and it is <form> tag. Like each tag, a form opens by <form> and closes by <form>. The <form> tag must contain method and action attribute. Method and action are two important attribute for a <form> tag, there are other attributes also which are not that important.

The method attribute of <form> tag can be either get or post i.e.

```
<form method="get">
        Or
<form method="post">
```

I will explain the difference between them later on.

Let us discuss about the other attribute first. A Form consists of some input boxes, list boxes, check boxes so as to accept input from the user. After user enter input, he presses the submit button, and all the inputs are processed. This statement "all the inputs are processed", means that the inputs are generally saved or inputs are used to generate some information for the user.

For example, if you are searching for a particular book on the internet, you have to fill up a form. That form may ask you for the name of the book, or the name of author i.e. you have to provide some input. The input provided by you is then processed, obviously HTML cannot process the input, and HTML can only accept the input from the user. The processing of input can be done using only scripting languages, like ASP, or JSP.

So to process the input, this HTML page passes the information to another web page which is made in ASP or JSP or in some other scripting language. Action attribute of the <form> tag specified the name of that web page.

So,

```
<form method="get" action="process.asp">
                Or
<form method="post" action="process.asp">
```

Let us now discuss about form elements i.e. input box, list box, check box etc, which are used for accepting inputs from the user and we will come back to <form> tag latter.

Forms: input TEXT

The above box is called as a text box; we want a text box like this in our HTML form. For this, we have a tag

<input type=”text” name=”u_name”>

Let us see how it works, type the following code in notepad and save as a html file.

```
<html>
	<head>
		<title>Using Form tag-text box</title>
	</head>

	<body>

		<form method="post" action="">
		Enter your name:&amp;nbsp;&amp;nbsp;<input type="text" name="u_name">
		</form>

	</body>
</html>
```

The output will be

Enter your name: 

There is another attribute that can be used with <input type=”text”>, and it is value.

```
<html>
	<head>
		<title>Using Form tag-text box</title>
	</head>

	<body>

		<form method="post" action="">
		Enter your name:&amp;nbsp;&amp;nbsp;<input type="text" name="u_name"
value="Dileep">
		</form>

	</body>
</html>
```

The output will be

Enter your name: Harshit

If you want the user should be able to enter password, and that password appears as '*' the screen, then

<input type="password" name="u_pass">

Type the following code

```
<html>
	<head>
		<title>Using Form tag-text box</title>
	</head>

	<body>

		<form method="post" action="">
		Enter your name:&amp;nbsp;&amp;nbsp;<input type="text" name="u_name" value="Dileep"> <br>
		Enter your password:&amp;nbsp;&amp;nbsp; <input type="password" name="u_pass">
		</form>

	</body>
</html>
```

And the output is

Enter your name: Harshit

Enter your password: **********

There are two more attributes of the <input> tag, size and maxlength. Size attribute defines the initial width of the control, and maxlength attribute specifies the maximum number of character a user can enter.

For example:
Type the following code in notepad and save as .html

```
<html>
	<head>
		<title>Using Form tag-text box</title>
	</head>
```

```
	<body>

		<form method="post" action="">
		Enter your name:&amp;nbsp;&amp;nbsp;<input type="text" name="u_name"
value="Dileep" maxlength="10"> <br>
		Enter your password:&amp;nbsp;&amp;nbsp; <input type="password"
name="u_pass">
		</form>

	</body>
</html>
```

Forms: additional input types

There are other input types that can be used with < input type= >. We have used type=text/password.

Type can also be used as <input type=checkbox/radio/image/submit/button/hidden>.

Let us take an example for each type, and try to understand this.
Type the following code which is for checkbox.

```
<html>
	<head>
		<title>Using Form tag-text box</title>
	</head>

	<body>

		<form method="post" action="">
			Which fruit do you like? <br>
		<input type="checkbox" name="fruit" value="apple"> Apple <br>
		<input type="checkbox" name="fruit" value="Mango"> mango <br>
		<input type="checkbox" name="fruit" value="Orange"> Orange <br>
		</form>

	</body>
</html>
```

The output is
Which fruit do you like?

- Apple
- mango
- Orange

Notice that in the above code,

- type=”checkbox”.
- All the input tag has a common name, i.e. name=”fruit”. You can give any name, but name should be common for all the checkboxes

If you think that most people like apples, you can pre-select it, some thing like this.
<input type="checkbox" name="fruits" value=”apple” checked> Apples

Type the following code,

```
<html>
	<head>
		<title>Using Form tag-text box</title>
	</head>

	<body>

		<form method="post" action="">
			Which fruit do you like? <br>
				<input type="checkbox" name="fruit" value="apple"
checked> Apple <br>
				<input type="checkbox" name="fruit" value="Mango">
mango <br>
				<input type="checkbox" name="fruit" value="Orange">
Orange <br>
		</form>

	</body>
</html>
```

And the output is
Which fruit do you like?

- [x] Apple
- [ ] mango
- [ ] Orange

Radio buttons are sets of circle-like selectors in which the user may only make one choice. The only difference between radio button and check box is number of selections. With checkbox user can select more than one option but with the radio button, use can only select one option.

The above code with radio button is like this.

```
<html>
	<head>
```

```
        <title>Using Form tag-text box</title>
    </head>

    <body>

        <form method="post" action="">
            Which fruit do you like? <br>
                <input type="radio" name="fruit" value="apple"> Apple
<br>
                <input type="radio" name="fruit" value="Mango"> mango
<br>
                <input type="radio" name="fruit" value="Orange">
Orange <br>
        </form>

    </body>
</html>
```

The output is
Which fruit do you like?

- Apple
- mango
- Orange

Notice that

- Type=”radio”
- All the input tag has a common name “radio”.

We will discuss later on about type=”image/button/submit/hidden”

Forms: textarea and option/select

<textarea> tag allows the user to enter multiple lines of text. It also has an opening and closing tag, like most of the form elements. It is used as follows

<textarea name=”u_text” rows=”4” cols=”10” wrap=”virtual>,

I think only one thing needs explanation here, and it is wrap, wrap=”virtual” means if user types any thing, the text should not go beyond the right side of the box.

Type the following code to understand more about <textarea>

```
<html>
    <head>
        <title>Using Form tag-TextArea</title>
```

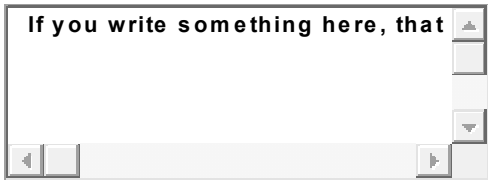
```
</head>

<body>

        <form method="post" action="">
        Write a short note about urself<br>
                <textarea name=”u_text” rows=”4” cols=”10”
wrap=”virtual>If you write something here, it will appear in the browser also.
                </textarea>
        </form>

</body>
</html>
```

Write a short note about urself



Anything you include between the opening and closing `textarea` tags will appear in the textarea box.

The `<select>` element works a lot like a radio button, except in that it used a cool drop down box. It also has a closing tag, `</select>`. Choices for the drop down box are created by using `<option>` tags.

```
 <select name="fav">
<option value="apples">apples</option>
<option value="oranges">oranges</option>
<option value="bananas">bananas</option>
</select>
```

Which fruit is your favorite?

Now let us put all this into one form, and then complete this lecture.

```
<html>
    <head>
        <title>Using Form tag-text box</title>
    </head>
```

```
    <body>

            <form method="post" action="">

Enter your name:&amp;nbsp;&amp;nbsp;<input type="text" name="u_name" value="Dileep">
<br>

Enter your password:&amp;nbsp;&amp;nbsp; <input type="password" name="u_pass"><br>


Write a short note about urself<br>
<textarea name=”u_text” rows=”4” cols=”10” wrap=”virtual>If you write something
here, that will appear in the

browser also.
</textarea><br>

Which fruit do you like? <br>
            <input type="checkbox" name="fruit" value="apple"> Apple <br>
            <input type="checkbox" name="fruit" value="Mango"> mango <br>
            <input type="checkbox" name="fruit" value="Orange"> Orange <br>

Which fruit do you like? <br>
                        <input type="radio" name="fruit" value="apple"> Apple
<br>
                        <input type="radio" name="fruit" value="Mango"> mango
<br>
                        <input type="radio" name="fruit" value="Orange">
Orange <br>


<select name="fav">
<option value="apples">apples</option>
<option value="oranges">oranges</option>
<option value="bananas">bananas</option>
</select>

            </form>

    </body>
</html>
```

The output will be

Enter your name: Harshit

Enter your password:

Write a short note about urself

If you write something here, that

Which fruit do you like?

- Apple
- mango
- Orange

Which fruit do you like?

- Apple
- mango
- Orange

apples

So, user will provide input, but after providing input he has to submit the input, for submitting input, we need a button.

So, next topic is How to make a button in HTML page. Very simple, tag for that is <input type = submit name="sub_b" >

Insert the line in the above code, and the output will be

Enter your name: Harshit

Enter your password:

Write a short note about urself

If you write something here, that

Which fruit do you like?

- Apple
- mango
- Orange

Which fruit do you like?

- Apple
- mango
- Orange

apples

Submit Query

If you click on the submit button, all the inputs will be passed to the page specified in action attribute of form tag. This topic we will discuss later when we will discuss ASP.

Lecture 17

You will learn in this Lecture

FormValidation

- Text Box validation
- Check Box validation
- Password validation
- select list box validation

Before starting with this lecture, There are few questions that deserve to be answered, What is validation and Why is it requried? Let me answer the first question and then we will move on to the second question.

So, the first question is, What is validation?
Validation test for required strings contain a specific value, a range of accepted value or a pattern of acceptable format. For example, If you make a form, and ask the user to enter name, email address, and comment. If he leaves every field blank, and click Enter. You will not get any information.
This means we are validating (checking) the values entered by the user, if he has entered some value or not, if not, then ask him to enter the value and if he has entered wrong value,then ask him to enter the correct value.

I hope you must have got the answer, why validation is required.

Next question is How validation is done, just read the following lines.

So, as soon as the user clicks Enter, There are two possibilites

1. Send the data that use has entered to the server, and check the values, if there are any, if there are no values entered by him or if wrong values are entered by him, ask him to enter the value, and move back to the same form.

2. Or, before sending the data to the server, check whether user has entered any value or not. If he has entered the value, then the value is corrent or not.

Second method is what we will follow, and is called as client side scripting, whereas the first method is called as Serve side scripting. JavaScript is popular for client side scripting and ASP, JSP, PHP are popular for server side scripting. It is high time to understand the difference between Client Side Scripting and Server Side Scripting.

When we use a scripting language, that works on the server side, it is called as Server Side Scripting language,
If we use a scripting language, that works on client side (i.e. browser), it is called as Client Side Scripting language.

So, Now let us start with validating user input.

When we make a form, we ask the user to enter username, password, and may ask him to fill up some check boxes, radio butons, and also ask him to write some comments. It is compulsay for the user to enter username, and password, but it is not compulsay for him to give his comments. So the things that are compulsary, cannot be left blank. This is one of the validation, forcing the user not to leave a field blank.

Let me list down some of the common validations needed
- Password Validation
- Text Field not blank (Name)

Below is the code for Text Box validation

```
<html>
<head>
	<script language="JavaScript">

	function validate()
	{
		firstName=document.myForm.fname.value;
		lastName=document.myForm.fname.value;
		if(firstName=="")
			window.alert("Name Field cannot be left blank");
		if(lastName=="")
			window.alert("Name Field cannot be left blank");
		switch (firstName.charAt(0))
		{
			case "0":
			case "1":
			case "2":
			case "3":
			case "4":
			case "5":
			case "6":
			case "7":
			case "8":
			case "9": window.alert("First chaaracter cannot be a number");
		}
	}

	</script>
</head>
	<body>
		<form name="myForm">
			<input type="text" name="fname"> <br>
			<input type="text" name="lname"> <br>
```

```
                <input type="submit" onClick="validate()"> <br>
            </form>
        </body>
</html>
```

And, Now we have the code for Password Validation

```
<html>
<head>
        <script language="JavaScript">

        function validate()
        {
                passwd=document.myForm.pass;
                cpasswd=document.myForm.cpass;
                if (passwd=="")
                        window.alert("Password field cannot be blank");
                if (cpasswd=="")
                        window.alert("Confirm Password field cannot be blank");

                if (passwd!=cpasswd)
                        window.alert("Passwords dont match");
        }

        </script>
</head>
        <body>
                <form name="myForm">
                        Password<input type="password" name="pass"> <br>
                        Confirm password<input type="password" name="cpass"> <br>
                        <input type="submit" onClick="validate()"> <br>
                </form>
        </body>
</html>
```

Validate Selection List

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function validateForm(objForm)
{
        var returnStatus = 1;

        if (objForm.Make.selectedIndex == 0) {
                alert("Please select a car make");
```

```
		returnStatus = 0;
	}
	else
		alert("You selected" +
objForm.Make.options[objForm.Make.selectedIndex].text + objForm.Make.value);

	if (returnStatus) {
		objForm.submit();
	}
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<FORM ACTION="test.asp" NAME="testform">
<SELECT NAME="Make">
	<OPTION VALUE="0" SELECTED>Select One</OPTION>
	<OPTION VALUE="1">Ford</OPTION>
	<OPTION VALUE="2">Chevy</OPTION>
	<OPTION VALUE="3">Pontiac</OPTION>
	<OPTION VALUE="4">Dodge</OPTION>
</SELECT>
<INPUT TYPE="BUTTON" VALUE="Send form"
onClick="validateForm(document.testform)">
</FORM>
</BODY>
</HTML>
```

Dynamically Populating a Selectin List

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function populate(objForm)
{


	if (objForm.Make.selectedIndex == 0)
	{
		alert("Please select a car make");
	}
	else
	{
		alert("You selected" +
objForm.Make.options[objForm.Make.selectedIndex].text + objForm.Make.value);
```

```
if (objForm.Make.selectedIndex == 1)
{
        objForm.Type.length=2;
        objForm.Type.options[0].text="d1";
        objForm.Type.options[0].value="1";

        objForm.Type.options[1].text="d2";
        objForm.Type.options[1].value="2";
}

if (objForm.Make.selectedIndex == 2)
{
        objForm.Type.length=2;
        objForm.Type.options[0].text="h1";
        objForm.Type.options[0].value="1";

        objForm.Type.options[1].text="h2";
        objForm.Type.options[1].value="2";
}

if (objForm.Make.selectedIndex == 3)
{
        objForm.Type.length=3;
        objForm.Type.options[0].text="m1";
        objForm.Type.options[0].value="1";

        objForm.Type.options[1].text="m2";
        objForm.Type.options[1].value="2";

        objForm.Type.options[2].text="m3";
        objForm.Type.options[2].value="3";
}

if (objForm.Make.selectedIndex == 4)
{
        objForm.Type.length=4;
        objForm.Type.options[0].text="v1";
        objForm.Type.options[0].value="1";

        objForm.Type.options[1].text="v2";
        objForm.Type.options[1].value="2";

        objForm.Type.options[2].text="v3";
        objForm.Type.options[2].value="3";

        objForm.Type.options[3].text="v4";
```

```
                        objForm.Type.options[3].value="4";
                }


        }


}
// -->
</SCRIPT>
</HEAD>
<BODY>
<FORM ACTION="test.asp" NAME="testform">
<SELECT NAME="Make" onChange="populate(document.testform)">
        <OPTION VALUE="0" SELECTED>Select One</OPTION>
        <OPTION VALUE="1">Daewoo</OPTION>
        <OPTION VALUE="2">Hyundae</OPTION>
        <OPTION VALUE="3">Mercedes</OPTION>
        <OPTION VALUE="4">Volswagen</OPTION>
</SELECT>

<SELECT NAME="Type">
</SELECT>
<INPUT TYPE="BUTTON" VALUE="Send form"
onClick="validateForm(document.testform)">
</FORM>
</BODY>
</HTML>
```

Validating a Check Box

Lecture 18

In this lecture, we will learn
-> Validating Radio Button

Validating Radio Button Selections

A radio button, provide a set of options, out of which only one can be selected. Infact, there are lot of simillarities between radio button and check box. Except one dissimillarity, which is, check box provide a lot of options to be selected whereas you can only select one option with radio button.

To know which option is selected, the following statement can be used

document.name of the form.name of the button[i].checked,

The above statement returns true if the button is selected otherwise false. The value of 'i' ranges from (0 to number_of_radio_buttons - 1).

Let us take an example.

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="JavaScript">
function validate()
{
	number_of_radio_buttons=document.myForm.hobby.length;
	for (i=0;i<=number_of_radio_buttons-1;i++)
	{
		if (document.myForm.hobby[i].checked)
		{
			flag=1;
			you_selected=document.myForm.hobby[i].value;
			window.alert(you_selected);
			return true;
		}
		else
			flag=0;
	}
	if (flag==0)
	{
			window.alert("Error! You should select one of the options");
			return false;
	}
}
</script>
```

```
</HEAD>

<BODY>
My hobbies are
        <form method="get" action="display.asp" name="myForm" onSubmit="return
validate()">
        <input type="radio" name="hobby" value="soccer"> Soccer <br>
        <input type="radio" name="hobby" value="read"> Reading <br>
        <input type="radio" name="hobby" value="music"> Listening Music <br>
        <input type="radio" name="hobby" value="travel"> Travelling <br>

        <input type="submit" value="Validate">
        </form>
</BODY>
</HTML>
```

Let us summarize, what we have learned by the above program
1. To know the number of radio buttons: document.name of form.name of radio button.length
2. To know index of radio button selected: document.name of form.name of radio button[i].selected
3. To know the value of radio button selected: document.name of form.name of radio button[i].value

Using the `Window` Object

We will discuss more about the Window object. We have been already using the Window object before.

Some of the features of Window object are

1. Creating Alert Dialog Boxes: window.alert()
2. Creating Confirmation Dialog Boxes: window.confirm()
3. Creating Dialog Boxes to get information from the user: window.prompt()
4. Opening Pages in new window: window.open()
5. Determining window size
6. Controlling scrolling of the document displayed in window
7. Scheduling the execution of function: window.setTimeout()

So, we have done all this before. This is all we have to learn about window object.

Open a Full Screen window

The following is the code to open a full screen window

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="JavaScript">
	function showWindow()
	{
		window.open("","myNewWindow","fullScreen=yes");
	}
</script>
</HEAD>
<BODY>
	<a href="a.html" onMouseOver="showWindow()" >Click1</a>
</BODY>
</HTML>
```

Handling the Parent-Child Relationship of Windows

When the window.open() method is used to open a new window from JavaScript, a relationship exist between the original window and new window so that it is possible to refer to both the windows from within JavaScript.

Well, to refer to the child window, simply do this

```
var newWindow=window.open("URL",window name);
```

Now you can refer to the new window, by newWindow.

Let us take an example

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="JavaScript">
	function showWindow()
	{
		myNewWindow=window.open("","myNewWindow");

	}

	function closeWindow()
```

```
	{
		myNewWindow.close();
	}

	function closeThisWindow()
	{
		window.close();
	}
</script>
</HEAD>
<BODY>
	<a href="" onMouseOver="showWindow()" >Take the Mouse here to open a New window</a> <br>
	<a href="" onMouseOver='closeWindow()' >Take the Mouse here to close the Opened Window</a><br>
	<a href="" onMouseOver="closeThisWindow()">Take the Mouse here to close this Window</a>
</BODY>
</HTML>
```

Let us take another example, where you will open a new window. The new window have the option for closing the parent window.

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="JavaScript">
	function showWindow()
	{
		myNewWindow=window.open("a.html","myNewWindow");

	}

	function closeWindow()
	{
		myNewWindow.close();
	}

	function closeThisWindow()
	{
		window.close();
	}
</script>
</HEAD>
<BODY>
	<a href="" onMouseOver="showWindow()" >Take the Mouse here to open a New
```

```
window</a> <br>

</BODY>
</HTML>
```

Here is the code for a.html

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="JavaScript">
        function closeOriginalWindow()
        {
                window.opener.close();
        }
</script>
</HEAD>

<BODY>
        <a href="" onMouseOver="closeOriginalWindow()">close the original
Window</a>
</BODY>
</HTML>
```

Writing into New Window

Below is the code, to write into new Window

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="JavaScript">
        function showWindow()
        {
                myNewWindow=window.open("","myNewWindow","width=200, height=200,
left=20, top=50");
                 myNewWindow.document.open();
                 myNewWindow.document.write("Take your mouse here to open    new
window");
                 myNewWindow.document.close();

        }

        function closeWindow()
        {
                 myNewWindow.close();
```

```
        }

        function closeThisWindow()
        {
                window.close();
        }
</script>
</HEAD>
<BODY>
        <a href="" onMouseOver="showWindow()" >Take the Mouse here to open a New
window</a> <br>

</BODY>
</HTML>
```

Referring Frames In JavaScript

If there are two frames on one page, name of left frame is frame1, and name of right frame is frame2,
Then frame1 can be referred in JavaScript as parent.frame1 and similarly, frame2 can be referred as paraent.frame2.

Let us take an example, first make frameset.html, which contains two frames, frame1 and frame2.

```
<frameset cols="50%,50%">
        <frame name="frame1" src="frame1.html">
        <frame name="frame2" src="frame2.html">
</framesest>
```

Here is the code for frame1.html

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="JavaScript">
        function writeFrame2()
        {
                parent.frame2.document.open();
                        parent.frame2.document.write(document.forms[0].f1text.value);
                parent.frame2.document.close();
        }
</script>
</HEAD>
<BODY>
<form>
```

```
<input type="text" name="f1text">
<input type="button" value="Write this to Frame 2" onClick="writeFrame2()">
<form>
</BODY>
</HTML>
```

To understand more about Frame object, Let me compare Frame object with Window object. In the previous topic, we saw that, window can use the document object, for ex. window.document.open(), Simmilarly, we can say frame.document.open(), and then can write into the frame, as you can see in the abov example. Infact we can use all the functions provided by the document object.
So, the moral of the story is "Frame object contain a document object".

Let us take another simple example

This is the code for frameset.html

```
<frameset cols=”50%,50%”>
        <frame name=”frame1” src=”frame1.html”>
        <frame name=”frame2” src=””>
</framesest>
```

Notice that, source for frame2 doesnot exist

Below is the code for frame1.html

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="JavaScript">
        function writeFrame2()
        {
                parent.frame2.document.location=document.forms[0].f1text.value;
        }
</script>
</HEAD>
<BODY>
<form>
Give the name of HTML file to open in Frame2
<input type="text" name="f1text">
<input type="button" value="Open File in Frame 2" onClick="writeFrame2()">
<form>
</BODY>
</HTML>
```

If Frames have no name, they can still be referred, like we referred forms. You guessed

it right, first frame is frames[0], second frame is frames[1], and so on. So we can refer to first frame as parent.frames[0].

Using Frames to Store Pseudo-Persistent Data

When you are working with frames, it is sometimes useful to be able to store information in JavaScript variables in such a way that the variable is available to both the frames.

```
<frameset cols=”50%,50%”>
        <frame name=”frame1” src=”frame1.html”>
        <frame name=”frame2” src=””>
</framesest>
```

In the above code, there are three documents.
1. Frame set document.
2. document present in frame1.
3. document present in frame2.

Actually frameset document, contains within itself two more documents, (document of frame1) and (document in frame2).

If you create any variable in document of frame1, it will not be accessible to document of frame 2. Simmilarly, if you create a variable in document of frame 2, it will not be accessible to document of frame1. So, you can create a variable in frameset document, which will be accessible to both frame 1 and frame 2.

So, now create a variable in frameset.html,
Below is the code for frameset.html

```
<script language="JavaScript">
        var pVariable="This is a persistent value"
</script>
<frameset cols=”50%,50%”>
        <frame name=”frame1” src=”frame1.html”>
        <frame name=”frame2” src=”frame2.html”>
</framesest>
```

Below is the code for frame1.html

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
</HEAD>
<BODY>
```

```
This is frame 1 <br>
<script language="JavaScript">
document.write("The value of persistent variable is "+parent.pVariable);
</script>
</BODY>
</HTML>
```

Below is the code for frame2.html

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
</HEAD>
<BODY>
This is frame 2 <br>
<script language="JavaScript">
document.write("The value of persistent variable is "+parent.pVariable);
</script>
</BODY>
</HTML>
```

Lecture 19

Using One Frame for Your Main JavaScript Code

As explained in the previous lecture, HTML offers a concept called frames that allows you to divide the available space in a given window into multiple frames into which you can load different documents.

When you are working with frames, it is sometimes useful to place all your JavaScript functions in one frame that will not change so they are easily accessible to all frames at all times.

For example, if all the functions fn1 (), fn2 () are placed inside frame1 and frame 2 want to call function fn1 (), frame 2 will use the following statement

```
Parent.frame1.fn1 ()
```

Let us take an example:

Below is the code for frameset.html

```
<HTML>
<frameset cols="50%,50%">
        <frame name="lp" src="frame1.html">
        <frame name="rp" src="frame2.html">
</frameset>
</HTML>
```

Below is the code for frame1.html

```
<HTML>
<HEAD>
<TITLE> Frame 1</TITLE>
<script language="JavaScript">
        function fn1()
        {
                window.alert("Function 1() called");
        }

        function fn2()
        {
                window.alert("Function 2() called");
        }
</script>
</HEAD>
<BODY>
        This is frame 1 <br>
        <a href="#" onClick="fn1()"> Call First Function </a> <br>
        <a href="#" onClick="fn2()"> Call Second Function</a>
</BODY>
</HTML>
```

Below is the code for frame2.html

```
<HTML>
<HEAD>
<TITLE>Frame 2</TITLE>
<script language="JavaScript">

</script>
</HEAD>
<BODY>
```

```
This is frame 2 <br>
        <a href="#" onClick="parent.lp.fn1()"> Call First Function </a> <br>
        <a href="#" onClick="parent.lp.fn2()"> Call Second Function</a>
</BODY>
</HTML>
```

Using a Hidden Frame for Your JavaScript Code

Sometimes you want to use a “hidden” frame to store a document containing nothing and only your JavaScript code.

```
<frameset cols=”0,50%,50%”>
        <frame …>
        <frame …>
        <frame …>
</frameset>
```

Let us take an example
Here is code.html

```
<HTML>
<HEAD>
<TITLE> Source code for all the frames</TITLE>
<script language="JavaScript">
        function fn1()
        {
                window.alert("function f1() called");
        }

        function fn2()
        {
                window.alert("function f2() called");
        }
</script>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Here is the frameset.html

```
<HTML>
<frameset cols="0,50%,50%">
        <frame name="codeFrame" src="code.html">
        <frame name="lp" src="frame1.html">
        <frame name="rp" src="frame2.html">
</frameset>
</HTML>
```

Below is frame1.html

```
<HTML>
<HEAD>
<TITLE> Frame 1</TITLE>
</HEAD>
<BODY>
        This is frame 1 <br>
        <a href="#" onClick="parent.codeFrame.fn1()"> Call First Function </a> <br>
```

```
        <a href="#" onClick="parent.codeFrame.fn2()"> Call Second Function</a>
</BODY>
</HTML>
```

Below is frame2.html

```
<HTML>
<HEAD>
<TITLE>Frame 2</TITLE>
</HEAD>
<BODY>
This is frame 2 <br>
        <a href="#" onClick="parent.codeFrame.fn1()"> Call First Function </a> <br>
        <a href="#" onClick="parent.codeFrame.fn2()"> Call Second Function</a>
</BODY>
</HTML>
```

Working with Nested Frames

All the examples that you have seen till now only deal with single layer frames. We can have nesting of frames that mean, a frame can be further sub divided into two frames. How to use JavaScript code with nested frames?

Here is an example to understand calling functions in Nested Frames

Below is the code for frameset.html

```
<HTML>
<frameset cols="50%,50%">
        <frame name="lp" src="frame1.html">
        <frame name="rp" src="frame2.html">
</frameset>
</HTML>
```

Below is the code for frame1.html

```
<HTML>
<HEAD>
<TITLE> Frame 1</TITLE>
</HEAD>
<BODY>
        This is frame 1 <br>
        <a href="#" onClick="parent.rp.subFrame2.fn1()"> Call Function in SubFrame 2 of Frame 2</a>
</BODY>
</HTML>
```

Below is the code for frame2.html

```
<HTML>
<frameset rows="20%,80%">
        <frame name="subFrame1" src="subFrame1.html">
        <frame name="subFrame2" src="subFrame2.html">
</frameset>
</HTML>
```

Below is the code for subFrame1.html

```
<HTML>
<HEAD>
<TITLE> Sub Frame1</TITLE>
</HEAD>
<BODY>
        This is sub Frame 1
```

```
</BODY>
</HTML>
```
Below is the code for subFrame2.html
```
<HTML>
<HEAD>
<TITLE> Sub Frame2 </TITLE>
<script language="JavaScript">
	function fn1()
	{
		window.alert("Function f1 called");
	}
</script>
</HEAD>
<BODY>
	This is subFrame 2
</BODY>
</HTML>
```

Dynamically Creating Frames in JavaScript

Dynamically creating frame means, creating the <frameset> and <frame> at runtime using JavaScript. Till now, we have been creating <frameset> using HTML. But now we will learn, How to create <frameset> at run time.

Here is the code for that.

```
<HTML>
<script language="JavaScript">
	document.open();
	document.write("<frameset cols='50%,50%'>");
		document.write("<frame src='frame1.html' name='lp'>");
		document.write("<frame src='frame2.html' name='rp'>");
	document.write("</frameset>");
	document.close();
</script>
</HTML>
```

Important thing that is worth mentioning here is that I have used single quotes (‘’) instead of (“”) in all the tags.

Lecture 20

Referring to Unnamed Frames Numerically

This is the last topic with using Frames in JavaScript. Like we did with Forms, same is the case with Frames. If we don't give any name to forms, then in JavaScript first form is referred as forms [0], second form is referred as forms [1]. This is because; JavaScript maintains an array of all the forms.
Similarly, if frames are unnamed, i.e. we don't give any name to frames, then in JavaScript first frame is referred as frames [0], second frame is referred as frames [1], and so on.

Let us take an example.
Below is the code for frameset.html

```
<HTML>
        <frameset cols="50%,50%">
                <frame src="frame1.html">
                <frame src="frame2.html">
</frameset>
</HTML>
```

Below is the code for frame1.hmtl

```
<HTML>
<HEAD>
<TITLE> Frame 1</TITLE>
<script language="JavaScript">
        function fn1()
        {
                window.alert("This is function in frame 1");
        }
</script>
</HEAD>
<BODY>
        This is frame 1 <br>
        <a href="#" onClick="parent.frames[1].fn1()"> Call Function in Frame2</a>
</BODY>
</HTML>
```

Below is the code for frame2.html

```
<HTML>
<HEAD>
<TITLE> Frame 2</TITLE>
<script language="JavaScript">
        function fn1()
```

```
        {
                window.alert("This is function in frame 2");
        }
</script>
</HEAD>
<BODY>
        This is frame 2 <br>
        <a href="#" onClick="parent.frames[0].fn1()"> Call Function in Frame1</a>
</BODY>
</HTML>
```

History Object

History Object Properties

- `Length`: Returns the number of items in the current history list

History Object Methods

- `Back()`: Moves back n items in the history list
- `Forward()`: Moves forward n items in the history list
- `Go()`: Moves to item n in the history list

There are currently no History object events. This means you cannot associate any event with the History object.

History object contains the list of URL's that your browser has visited. In other words, History object is an array of all URL's visited so far with the current open browser.

Since History object is an array, so it must have a length property, which will give you the number of URL visited so far.

Generally, if you want to go back to the previous page, you click the back button in the menu bar of your browser. And, in case you want to go back to the next page, you click the forward button
Notice: When you open a browser, back and forward button both are disabled.

If you want, you can provide that buttons on your web page, so that user can move back and forward using your buttons rather than using browser buttons.

There are three ways to implement back and forward buttons

1. Back and Forward Using History
2. Back and Forward Buttons (Simple)
3. Back and Forward Buttons Without History

Back and Forward Using History

Now, we will understand How to code back and forward button using history.

Below is the code

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script LANGUAGE="JavaScript">
        function back()
        {
                history.go(-1);
        }

        function forward()
        {
                history.go(+1);
        }
</script>
</HEAD>

<BODY>
<FORM>
<INPUT TYPE="button" VALUE="<< Back" onClick="back()">
<INPUT TYPE="button" VALUE="Forward >>" onClick="forward()">
</FORM>
</BODY>
</HTML>
```

Back and Forward Button (Simplest Way)

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script LANGUAGE="JavaScript">
        function back()
        {
                Location.href=”a.html”;
        }

        function forward()
```

```
        {
                Location.href=”b.html”
        }
</script>
</HEAD>

<BODY>
<INPUT TYPE="button" VALUE="<< Back" onClick="back()">
<INPUT TYPE="button" VALUE="Forward >>" onClick="forward()">
</BODY>
</HTML>
```

Back and Forward buttons without using history object

```
<script language="JavaScript">

var MyLocation=window.location;

var MyPage=new Array;

{

        MyPage[0]="index.html";

        MyPage[1]="alerts.htm";

        MyPage[2]="alertsText.htm";

        MyPage[3]="alertsImages.htm";

        MyPage[4]="jumpfunction.htm";

        MyPage[5]="alertConfirm.htm";

        MyPage[6]="alertConfirm2Locs.htm";

        MyPage[7]="alertOnLoad.htm";

        MyPage[8]="alertHelloEx.htm";

        MyPage[9]="alertPrompts.htm";

        MyPage[10]="documentWrite.htm";
```

```
        MyPage[11]="documentWrite2.htm";
        MyPage[12]="WindowNew.htm";
        MyPage[13]="WindowNew2.htm";
        MyPage[14]="WindowNewHyperlinks.htm";
        MyPage[15]="menu1.htm";
        MyPage[16]="menuSelectOnChange.htm";
        MyPage[17]="ArrayText.htm";
        MyPage[18]="jsFrames.htm";
        MyPage[19]="frameHyperlinks.htm";
        MyPage[20]="BackForward.htm"; }
function GoBack() {
if (ThisPageNumber <=0) {
alert("You are at the beginning of this series");
}
else{ ThisPageNumber=ThisPageNumber-1;
window.location=MyPage[ThisPageNumber];
} }
function GoForward(){
ThisPageNumber=ThisPageNumber+1;
if (ThisPageNumber >=MyPage.length) {
answer=confirm("You are at the end of the present series. "+
"Press ok to go to the beginning. Cancel to stay here");
```

```
if (answer!=0) {

ThisPageNumber=0; window.location=MyPage[ThisPageNumber]; }
ThisPageNumber=ThisPageNumber-1;

}

else {

window.location=MyPage[ThisPageNumber];

}

}

</script>
```

Lecture 21

Three ways of appliying CSS

As the heading says, there are three ways of applying CSS.

1. Inline
2. Internal
3. External

I don't want to comment which one is better; it depends on your personal preference. So you can use any one of them. But gradually, you will be able to figure out, which one suits your style of coding.

Internal

This is the easiest one, and I feel quite comfortable using this. So, I am starting with this.

```
<HTML>
<HEAD>
	<style type="text/css">
		p
		{
			color: red;
		}
	</style>
</HEAD>
<BODY>
<p> This is the first paragraph of this page, and it should appear in
red</p>

</BODY>
</HTML>
```

Let me explain, what I have done. I associated a "property: value (color: red)" pair with a HTML Tag. But p inside the <style> block is known as selector, as explained above. This also means, wherever in my page <p> tag is used, text will appear in red.

In-line

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<p style="color:red"> This is the first paragraph of this page, and it
should appear in red</p>

</BODY>
</HTML>
```

I don't think that it deserves any explanation.

External

This one is quite different and is most popular among professionals. Why is it popular, I am not goanna tell you that so easily.

How it works, Write the style in a file, and save the file with ".css" extension. Just look at the example below.
We have two files "pstyle.css" and "style_external.html".
Below is pstyle.css

```
<p>
      color:red;
</p>
```

Below is style_external.html

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<p style="color: red"> This is the first paragraph of this page, and it
should appear in red</p>
</BODY>
</HTML>
```

Q. Are you still thinking, why using CSS this way is so popular among professionals?
A. The advantage is to define all the styles in one .css file. Include the css file in every web page of your web site and use which ever style you want to use.

Lengths, Percentages and Colors

This part of the lecture is worth mentioning although it seems quite trivial. We will discuss about various measuring units for font size, width and height.
Until now we were using HTML, and in HTML if I want to define the size of some font, I will give the size in integer. But If I want the size of font in decimals (too small), it is not possible in HTML, now it is possible.
So, the coder has more control over how his web page looks.
The various measurements available are
'em': for example, font-size:2em (2em is approximately equal to height of a character)
'px': for example, font-size:12px (unit is pixels)
'pt': for example, font-size:12pt (unit is points)
'%': for example, font-size:80% (unit is percentage)

Other unit includes 'pc' (picas), 'cm' (centimeters), 'mm' (millimeters), and 'in' (inches).

- In style sheets, for text color we use 'color' as attribute and for back ground color we use 'background-color'.
- To specify the name of font, 'font-family' attribute is used.
- To specify the size of font, 'font-size' attribute is used.
- To specify whether the text is italic, use 'font-style' and possible values are 'italic/normal'.
- Another attribute is 'text-decoration', and possible values are
  - `text-decoration: overline`, which places a line above the text.
  - `text-decoration: line-through`, which puts a line through the text.
  - `text-decoration: underline`.
- Another attribute is 'text-transform', and possible values are
  - `text-transform: capitalize`, transform first character of every word into uppercase.

- `text-transform: uppercase`, turns everything into upper case.
- `text-transform: lowercase`, turns everything into lower case.

Let us take an example.

```
<HTML>
<HEAD>
	<style type="text/css">
		body
		{
			color: yellow;
			background-color:black;
			font-family: "Times New Roman";
			font-size: 14;
		}

		a
		{
			text-decoration: line-through;
			font-family: "Arial";
			text-transform:uppercase;
			font-size: 20;
		}
	</style>
</HEAD>
<BODY>
The  back ground of this page is black and text color is yellow <br>
My name is <a> Dileep </a>
</BODY>
</HTML>
```

- The attribute "letter-spacing" is used for spacing between letters. The value of the attribute is length in %, or em, or pt.
- The attribute "word-spacing" is used for spacing between words. The value of the attribute is length in %, or em, or pt.
- The attribute "text-align" is used for aligning the text. The possible value for the attribute are center, left, right, justify.
- The attribute "line-height" sets the height of the lines in a paragraph.

```
<HTML>
<HEAD>
	<style type="text/css">
		body
		{
			color: yellow;
			background-color:black;
			font-family: "Times New Roman";
			font-size: 14;
		}

		a
		{
			text-decoration: line-through;
			font-family: "Arial";
			text-transform: uppercase;
			font-size: 20;
		}
```

```
            p
            {
            letter-spacing: 0.3em;
            word-spacing: 2em;
            line-height: 1.5em;
            text-align: left;
            }
      </style>
</HEAD>
<BODY>
The  back ground of this page is black and text color is yellow <br>
My name is <a> Dileep </a>
<p>
      This is my first paragraph, <br>      trying to notice How much is
<br>letter spacing which is 0.5 em <br> word spacing 2m<br>line height
1.5em <br>text alignment, you tell me
</p>
</BODY>
</HTML>
```

Margins and Padding

Another important section of CSS, I guess all of you must be wondering, why we are discussing all these things, I mean, margins, padding, line height, word spacing, letter spacing, etc.
Well, this gives us more control over how we display and align our text on browser.

A margin is the space outside of the element.
Padding is the space inside the element.

```
<HTML>
<HEAD>
      <style type="text/css">
            h2
            {
            font-size: 1.5em;
            background-color: pink;
            margin: 5em;
            padding: 10em;
            }

      </style>
</HEAD>
<BODY>
<h2> Hey margin is 5em, and <br> padding is 10em</h2>

</BODY>
</HTML>
```

In the output, the distance between the text and where the pink box end, is the padding. The distance between any boundaries of the browser to the point where the pink box starts.

The Box Model

Margin, padding and border, they all work together. In the center you have the element, surrounding it is padding, and surrounding padding is margin box. Margin box is surrounded by border box.

Let us see how to use borders.
The various attributes are

1. border-style, the possible values are solid, dashed, dotted, double, groove, ridge, inset and outset.
2. border-width, the width of the border, it can be in em, %, etc
3. border-top-width, border-bottom-width, border-left-width, border-right-width.

```
<HTML>
<HEAD>
	<style type="text/css">
		h2 {
		border-style: solid;
		border-width: 3px;
		border-left-width: 20px;
		border-right-width: 10px;
		border-top-width: 50px;
		border-bottom-width:100px;
		border-color: red;
		}


	</style>
</HEAD>
<BODY>
<h2> Hey margin is 5em, and <br> padding is 10em</h2>

</BODY>
</HTML>
```

Let us learn another better way of making styles, We have two techniques

1. Selectors
2. ID

So, let us start out journey with Selectors, also known as Selector Class.

Selector Class

With Class Selector, you can define different type of 'styles' for the same 'selector'. Say, that you want to have two types of paragraphs in your document, one paragraph is left aligned and another paragraph is right aligned. Here is how you can do this

```
<HTML>
<HEAD>
	<style type="text/css">
		p.class1
		{
			text-align:left;
		}
		p.class2
		{
			text-align:right;
		}
	</style>
</HEAD>
<BODY>
<p class="class1">This is first paragraph, Is the test left aligned</p>
<p class="class2">This is second paragraph, Is the test right aligned</p>
</BODY>
```

```
</HTML>
```

Just Define the Class

Q. Is it possible to just define the class (no selector), and use it with the HTML tags
Ans. Yes

Q. How is it possible?
A) Simply defines the class as you used to define in "Selector Class", but leave the selector out, like this.

```
.center
{
      text-align=center
}

In <body> section,
<h1 class="center"> It looks good in the center</h1>
<p class="center"> This paragraph is center aligned</p>
```

Let us write an example to understand above concept.

```
<HTML>
<HEAD>
      <style type="text/css">
            .center
            {
                  text-align:center;
                  background:green;
            }
      </style>
</HEAD>
<BODY>
<h1 class="center"> This is h1 tag in center </h1>
<p class="center"> This paragraph is center aligned</p>

</BODY>
</HTML>
```

And the output is as follows

Like Class Selector, there is one more Selector, ID Selector. It is very much similar to Class Selector with a slight difference, even syntax is almost same.

```
h1#myheader
{
      Background: green
}
```

So, you got the difference, the difference is, "#" instead of "." And now, "myheader" is known as ID.

Take an example

```
<HTML>
<HEAD>
	<style type="text/css">
		h1#myheader
		{
		color: blue;
		background: green
		}

	</style>
</HEAD>
<BODY>
<h1 id="myheader"> This is h1 header </h1>
</BODY>
</HTML>
```

The output is

This is h1 header

CSS Comments

You can insert comments in CSS; purpose is same to write down what you want to do. Below is an example of comment

```
/*
	Here you can write CSS coment.
*/
```

Lecture 22

Nesting

If CSS is structured well, there should not be a need to use of many Class ID or selectors.

For example:

```
#top {
        background-color: #ccc;
        padding: 1em
}

#top h1 {
        color: #ff0;
}

#top p {
        color: red;
        font-weight: bold;
}
```

Now h1 contains the “attribute:value” pair of selector ID “#top” also, and another “attribute:value” pair i.e. “color:#ff0”.

CSS Dimension Properties

```
I: How to Set the Width and Height of an image
<html>
<head>

<style type="text/css">
img.normal
{
height: auto;
width: auto
}

img.big
{
height: 80px;
width: 100px
}

img.small
{
height: 30px;
width: 50px
}
</style>

</head>

<body>
```

```
<p><b>Note:</b> Netscape 4 does not support the "height" property, and the
"width"
property does not work on images.</p>

<img class="normal" src="back.gif" width="72" height="40">
<br><br>
<img class="big" src="back.gif" width="72" height="40">
<br><br>
<img class="small" src="back.gif" width="72" height="40">

</body>
</html>
```

Dimension Properties:

NN: Netscape, **IE**: Internet Explorer, **W3C**: Web Standard

Property	**Description**	**Values**	**Example**

height	Sets the height of an element	auto
		length
		%
			img
			{
			height: 230px
			}

line-height	Sets the distance between lines	normal
		number
		length
		%
			p
			{
			max-height: 100px
			}

max-height	Sets the maximum height of an element	none *length* *%*	

max-width	Sets the maximum width of an element	none
		length
		%
			h2
			{
			max-width: 500px
			}

min-height	Sets the minimum height of an element	*length*
		%
			p
			{
			min-height: 10px
			}

min-width	Sets the minimum width of an element	*length*
		%
			h2
			{
			min-width: 50px
			}

width	Sets the width of an element	auto *%* *length*	Refer example below

CSS Classification Properties

The classification property allows you to control

1. How to display an element.
2. Set where an image will appear in another element.
3. Position an element relative to its normal position.
4. Position an element using an absolute value.
5. To control the visibility of an element

Property	**Description**	**Values**	
clear	Sets the sides of an element where other floating elements are not allowed	left right both none	

cursor	Specifies the type of cursor to be displayed	*url*
		auto
		crosshair
		default
		pointer
		move
		e-resize
		ne-resize
		nw-resize
		n-resize
		se-resize
		sw-resize
		s-resize
		w-resize
		text
		wait
		help
			h2
			{
			cursor:
			crosshair
			}

display	Sets how/if an element is displayed	none inline block list-item run-in compact marker table inline-table table-row-group table-header-group table-footer-group	

		table-row table-column-group table-column table-cell table-caption	
float	Sets where an image or a text will appear in another element	left right none	Refer example below
position	Places an element in a static, relative, absolute or fixed position	static relative absolute fixed	
visibility	Sets if an element should be visible or invisible	visible hidden collapse	

Creating a Horizontal menu

```
<html>
<head>
<style type="text/css">
body
{
cursor: e-resize
}
ul
{
float:left;
width:100%;
padding:0;
margin:0;
list-style-type:none;
}
a
{
float:left;
width:6em;
text-decoration:none;
color:white;
background-color:purple;
padding:0.2em 0.6em;
border-right:1px solid white;
}
a:hover {background-color:#ff3300}
li {display:inline}
</style>
</head>

<body>
<ul>
<li><a href="#">Link one</a></li>
<li><a href="#">Link two</a></li>
<li><a href="#">Link three</a></li>
<li><a href="#">Link four</a></li>
```

```
</ul>

<p>
In the example above, we let the ul element and the a element float to
the left.
The li elements will be displayed as inline elements (no line break
before or after the element). This forces the list to be on one line.
The ul element has a width of 100% and each hyperlink in the list has a
width of 6em (6 times the size of the current font).
We add some colors and borders to make it more fancy.
</p>

</body>
</html>
```

An Example of Float

```
<html>
<head>
<style type="text/css">
img
{
float:right
}
</style>
</head>
<body>

<p>
In the paragraph below, we have added an image with style
<b>float:right</b>. The result is that the image will float to the right in
the paragraph.
</p>

<p>
<img src="logocss.gif" width="95" height="84" />
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>

</body>
</html>
```

Another Example on the same line of Float

```
<html>
<head>
<style type="text/css">
div
{
float:right;
width:120px;
```

```
margin:0 0  15px 20px;
padding:15px;
border:1px solid black;
text-align:center;
}
</style>
</head>

<body>
<div>
<img src="logocss.gif" width="95" height="84" /><br />
CSS is fun!
</div>
<p>
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>

<p>
In the paragraph above, the div element is 120 pixels wide and it contains
the image.
The div element will float to the right.
Margins are added to the div to push the text away from the div.
Borders and padding are added to the div to frame in the picture and the
caption.
</p>

</body>
</html>
```

A Home Page without Using Tables

```
<html>
<head>
<style type="text/css">
div.container
{
width:100%;
margin:0px;
border:1px solid gray;
line-height:150%;
}
div.header,div.footer
{
padding:0.5em;
color:white;
background-color:gray;
clear:left;
```

```
}
h1.header
{
padding:0;
margin:0;
}
div.left
{
float:left;
width:160px;
margin:0;
padding:1em;
}
div.content
{
margin-left:190px;
border-left:1px solid gray;
padding:1em;
}
</style>
</head>
<body>

<div class="container">
<div class="header"><h1 class="header">kumarharmuscat.tripod.com</h1></div>
<div class="left"><p>"Never increase, beyond what is necessary, the number
of entities required to explain anything." William of Ockham (1285-
1349)</p></div>
<div class="content">
<h2>Free Web Building Tutorials</h2>
<p>At My Site you will find all the Web-building tutorials you need,
from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.</p>
<p>W3Schools - The Largest Web Developers Site On The Net!</p></div>
<div class="footer">Copyright 1999-2005 by Dileep Kumar</div>
</div>

</body>
</html>
```

Pseudo Classes

The syntax for Pseudo Class is

```
Selector:pseudoclass
{
        attribute1: value1;
        attribute2: value2;
        .
        .
        attributen: valuen;
}
```

```
a.snowman:link {
        color: blue;
}
```

```
a.snowman:visited {
        color: purple;
}

a.snowman:active {
        color: red;
}

a.snowman:hover {
        text-decoration: none;
        color: blue;
        background-color: yellow;
}
```

Lecture 2

In this lecture we will learn

1. Arrays
2. Expressions and Operators
3. Functions
4. If-else construct
5. Switch Construct
6. loop constructs
   - For
   - While
   - do-while

More about data types

As I told in my previous lecture, we can declare a variable by

var variable name;

for example:

var x;

The variable 'x' can store integers, characters, string, float, boolean.

There are five types of data or five data types in Javascript.
Integer
character
string
float
boolean

For all the data types, there is a common keyword **var**.
boolean values are only true or false.

String is a sequence (연달아 일어남, 연속) of characters enclosed (둘러싸다) inside double quotes (“ ”). For example “hello” .

Integer is a numeric (수) value without decimal. For example 2

Float is a numeric value with decimal. For example 2.5 or 2.0

character is a single character enclosed (둘러싸다) inside single quote (' '). For example 'a'.

Arrays

Like other programming languages, Javascript supports creating arrays.

An array is a collection of data of same type.

By data of same type, I mean, all the data can be either integer or character or float or string or boolean but not a combination (결합, 짝맞춤, 배합) of either ([긍정문에서] (둘 중)) of these.

Like C language, starting index is 0.

For example:

```
<body>
	<script language="JavaScript">
	<!--
		var myArray = new Array (5);
		myArray [0] = 0;
		myArray [1] = 1;
		myArray [2] = 2;
		myArray [3] = 3;
		myArray [4] = 4;
		myArray [5] = 5;

		var secondArray = new Array(6,7,8,9);

		// printing the elements of the first array
		document.write("The elements of myArray are <br>");
		document.write(myArray[0] + "<br>");
		document.write(myArray[1] + "<br>");
		document.write(myArray[2] + "<br>");
		document.write(myArray[3] + "<br>");
		document.write(myArray[4] + "<br>");
		document.write(myArray[5] + "<br>");

		printing the elements of the second array
		document.write("The elements of SecondArray are <br>");
```

```
                document.write(secondArray[0] + "<br>");
                document.write(secondArray[1] + "<br>");
                document.write(secondArray[2] + "<br>");
                document.write(secondArray[3] + "<br>");


        // -->
        </script>
</body>
```

There are 2 ways of declaring arrays in the above example.

1. var array_name = new Array (size of array);
   array_name[0] = $value_1$;
   array_name[1] = $value_2$;
   array_name[2] = $value_3$;
   .
   .
   .
   array_name[size of array] = $value_n$;

2. var array_name = new Array ($value_1$, $value1_2$, $value_3$,.....,$value_n$);

Expressions and Operators

Expression is a sequence (연달아 일어남, 연속) of operators and operands.

For example: 2 + 3 .
The above expression consist of 2 operands and 1 operator.

- + is an operator
- 2 and 3 are operands.

Another example: z = x + y
The above expression consist of 3 operands and 2 operators.

- + and = are operators. (= is called as assignment operator)
- x, y and z are operands.

There are 4 types of expressions.

1. Arithmetic expression
2. Logical expression
3. String Expression
4. Relational expression

Arithmetic Expression

Expression formed using arithmetic operator are called as arithmetic expression.
There are 5 types of Arithmetic operators.

1. addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/) - gives the quotient of division operation.
5. modulus (%) - gives the remainder of division operation

Relational expression

Expressions formed using comparison operator are called as relational expression.
There are 6 types of Comparison operator.

1. equal to (==)
2. not equal to (!=)
3. Less than (<)
4. Less than or equal to (<=)
5. greater than (>)
6. greater than or equal to (>=)

Return value of a relational expression is boolean i.e. true or false.

Logical Expression

Logical expressions are used to combine (결합, 짝맞춤, 배합) different comparisons. There are 2 types of logical operators.

1. Logical AND (&&)
2. Logical OR (||)

For example:

```
a = (2==3);          ---------------------- (1)
c = (2 != 3);        ---------------------- (2)
e = a && c           ---------------------- (3)
```

Equation (1) and (2) are relational expression.
Equation (3) is a logical expression.

Logical expression also return boolean values i.e. True or false.

String Expression

Expression formed using Strings and '+' operator are called as String expression.
'+' operator has a dual (둘의;둘을 나타내는, 양자의) functionality. When '+'

operator is used with numbers it functions as an arithmetic operator. When '+' operator is used with strings it functions as a concatenation operator.

Example of '+' as concatenation operator:

```
var name = " huh";
var greeting = "Welcome";
var welcome =  greeting + name;
```

welcome variable contains "Welcome huh".

Operator Precedence(앞섬)

Arithmetic operators has the highest priority (앞섬). After arithmetic operator comes comparison operator and then comes logical operator.

Within ((장소·시간·거리·범위·한계 등)) each group, operators have priority or precedence. For example, within the arithmetic operators, '%' has highest priority, followed by '*' and '/' and then at the end is '+' and '-'.

The priority of arithmetic operator is

1. '%'
2. '*', '/'
3. '+', '-'

Remember these rules

1. Operators are executed in the order of their priority, i.e. operators with highest priority are executed first.
2. Operators having same priority are executed in Left to right order.
3. you can override (보다 우위에 서다) operator priority by enclosing (둘러싸다) an expression inside parenthesis (괄호, 소괄호).

For example:

```
var c = 5/9  * F – 32;                ---------------- (1)
```

In the above expression we are converting (변하게 하다, 전환하다) Fahrenheit (화씨의) to Celsius (섭씨의).

If F = 50, then

```
var c = 5/9 * 50 – 32;
```

There are 3 operators on RHS: '/', '*', '-'. Among (...의 사이에, ...의 가운데에, ...에 둘

러싸여) '/', '*', '-' operators, '/' and '*' have highest priority. That means either of '/' or '*' will be executed first.

'/' and '*' have equal priority, that means, they will evaluated from left to right.

In equation (1), '/' appear on the left, so first '/' operation is executed. That means 5/9 is executed first, and 5/9 = 0.556

var c = 0.556 * 50 - 32

After '/' comes '*', that means 0.56 * 50 = 27.8

var c = 27.8 -32

And then comes '-', that means 27.8 – 32 = -4.2

var c = -4.2

If we apply parenthesis around F – 32 i.e

var c = 5 / 9 * (F - 32) ---------- (2)

If F = 50, then

var c = 5 /9 * (50 -32)

Rule 3 is applied first, parenthesis are always evaluated first, that means (50 – 32) = 18.

var c = 5 /9 * 18

Now rule 2 is applied, because '/' and '*' have same priority. This means, execute from left to right.

var c = 0.556 * 18

finally answer is

var c = 10.

Functions

Like many other programming languages, Javascript also support functions. A function is a self (자기, 자신) contained (억제[자제]하는, 조심스러운) operation that can be called by function name. If function expects some input parameters (【수학·컴퓨터】 파라미터, 매개 변수) then pass few parameters also.

To call a function, you simply use the following form:

functionName (argument1, argument2, argument3,.....);

If a function has no input arguments, you still need parenthesis , for example:

functionName ();

Also, if a function returns some value, you can use the function call to assign (배당하다) the returned value to some variable, for example:

```
var returnValue = functionName ();
```

Functions are of two types,

1. System Defined Functions
2. User Defined Functions

System defined functions are those which are provided by the system, whereas User defined functions are those for which are made by the user.

An example of System defined function

```
<body>
        <script language="Javascript">
        <!--
                window.alert("How are you");
        // -->
        </script>
</body>
```

Output should be a pop-up dialog with message, "How are you".

The above Javascript code, uses a system defined function,window.alert().

Window is an object. Function alert() is provided by window object.

Confirming with User

There is another function, which you can be used to ask a question from user, and get his response (대답) as True or False. The name of the function is window.confirm(); Window object provides confirm() function, which displays a dialog box containing text message, and provides two options, "OK" and "Cancel".

This function also returns true or false, depending upon what user clicks. If user clicks "OK", confirm() returns true, and if user clicks "Cancel", then confirm() returns false.

For example:

```
<body>
        <script language="JavaScript">
        <!--
                var result=window.confirm("How are you");
                document.write(result);
        // -->
        </script>
</body>
```

Output should be "true", if you click "Ok" or "false', if you click "Cancel".

Taking input from User

Sometimes we may need to take some input from the user. We can ask a user to enter his name or to enter a number. Javascript provides a function window.prompt(). 'window' is an object in Javascript and prompt() is a function of window object.

To understand window.prompt(), here is an example:

```
    <body>
        <script language="Javascript">
            var name = window.prompt("enter your name");
            window.alert("welcome " + name);
        </script>
    <body>
```

Creating your own Functions

Like I told above, functions are of two types, System defined functions and User defined functions. We have already discussed about System Defined Functions, Now we will discuss about User defined functions.

Creating a User Defined function goes like this

```
function functionName()
{
```

```
        statement1;
        statement2;
        .
        .
        .
        statementn;
}
```

For example:

```
<HTML>
<HEAD>
        <script language="JavaScript">
                function displayHello()
                {
                        document.write("Hello");
                }
        </script>
</HEAD>
<BODY>

        <script language="JavaScript">
                displayHello();
        </script>

</BODY>
</HTML>
```

In the above program, I have declared a function "displayHello()" in the head section of the document. Function "displayHello()" is called in the body section of the document.

Passing an Argument to Your Functions

Arguments are needed in a function to pass (유월절) the

values. To create a function that accepts arguments, you must specify names for each argument in the argument definition.

Syntax of function declaration with input arguments,

```
function functionName(argumentName1)
{
	statement1;
	statement2;
	.
	.
	.
	statementn;
}
```

For example:

```
<HTML>
<HEAD>
	<script language="JavaScript">
		function square(num)
		{
			var result=num * num;
			document.write(result);
		}
	</script>
</HEAD>
<BODY>

	<script language="JavaScript">
		square(10);
	</script>

</BODY>
</HTML>
```

Return Values From Function

Now, we will discuss another example where function will return a value. In the previous example, our function received a number, the square of that number was computed, and displayed. But now, our function will receive a number, compute the square and return the square back.

To return a value, return command is used.

Syntax:

```
function functionName()
{
	statement1;
	statement2;
	.
	.
	statementn;
	return value;
}
```

For example:

```
<HTML>
<HEAD>
	<script language="JavaScript">
		function square(num)
		{
			var result=num * num;
			return result
		}
	</script>
</HEAD>
<BODY>
```

```
        <script language="JavaScript">
                var ans=square(10);
                document.write(ans);
        </script>

</BODY>
</HTML>
```

Passing Multiple Parameters to Your Function

In the Example above, we discussed about passing single argument, now we will discuss about how to pass more than one argument. It is very simple similar to the code above.

```
function functionName(argument1, argument2, argument3,.....................)
```

We will write a program that passes two arguments to a function, that function multiples the value contained in those two arguments, and returns back the result.

```
<HTML>
<HEAD>
        <script language="JavaScript">
                function multiply(num1,num2)
                {
                        var result=num1 * num2;
                        return result;
                }
        </script>
</HEAD>
<BODY>

        <script language="JavaScript">
                var ans=multiply(10,20);
                document.write(ans);
```

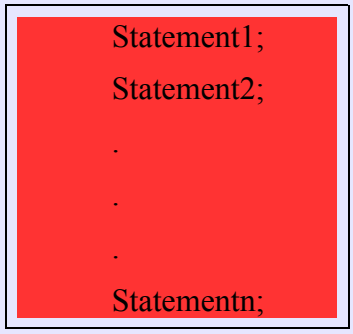
```
        </script>


</BODY>
</HTML>
```

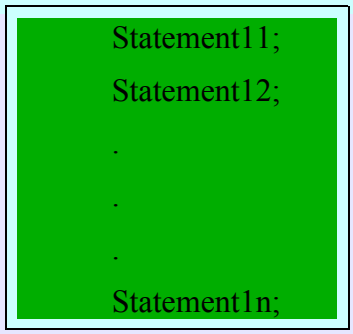
If-Construct

Syntax

```
if (condition or comparison expression)
{
```




```
}
else
{
```




```
}
```

If condition evaluates (평가하다) to true, first block is executed. If condition evaluates to false, second block is executed.

For example:

```
x=2;
y=3;
if (x == y)
{
      document.write ("x and y are equal");
```

```
	}
	else
	{
		document.write(“x and y are not equal”);
	}
```

Another example of If-construct
In this example, we will ask the user for his Id and password. If he enters correct Id and password, we will welcome him, otherwise an error message is displayed.

```
<html>
<head>
<script language="javascript">

var username="user";
var password="haveaniceday";
var entered_username=prompt("enter username: ", " ");
var entered_password=prompt("enter password: ", " ");

if ((entered_username == username) && (entered_password == password))
{
  document.write("You have been Logged in!")
}
else
{
  document.write("Sorry, either the username or password is incorrect.")
}

</script>
</head>

<body>
</body>
</html>
```

Switch-Case Construct

If we want to use multiple if, which we do when have lot of conditions to check and do something. For example, If there are 3 types of user: administrator, developer, and user.

If user enters his password, we display “you have user access”.
If administrator enters his password, we display, “you have administrator access”
If developer enters his password, we display, “You have developer access”

In the above scenario (【연극】 대본), we need to user more than 1 If-else. Using more than 1 if-else can some times become clumsy. We can use switch-case instead (그 대신에) of using more than one if-else.

For example

```
<html>
<head>
<script language="javascript">

var password_administrator="ice";
var password_developer="snow";
var password_user="water";

var entered_password=prompt("enter the secret code to continue: ", " ");

      switch (entered_password) {
         case "ice":
alert("you have administrator access");
            break
         case "snow":
alert("you have developer access");
            break
         case "water":
alert("you have user access");
            break
         default:
alert("password incorrect!");
            break
      }

</script>
</head>

<body>
</body>
</html>
```

Looping statements

loops are used when we want a part of code to be repeated (되풀이하다, 반복하다) till the condition is true. As the condition becomes false, loop terminates (<행동·상

태 등을> 끝내다).

There are three types of looping constructs available in Javascript.

1. For
2. While
3. do-while

Syntax of For:

```
for (initialize statement; condition; increment/decrement statemtn)
{
        statement1;
        statement2;
        .
        .
        .
        statementn.
}
```

An example of for loop.

```
<html>
        <head>
                <title>Associating a Function with a Link</title>
                <script language="JavaScript">
                <!--
                        for(var i=0;i<10;i++)
                                document.write(i+"<br>");
                // -->
                </script>
        </head>
        <body">

        </body>
</html>
```

Output for the above code is display numbers from 0 to 9.

Syntax of while loop

```
initialize statement;
while (condition)
{
        statement1;
        statement2;
        .
        .
        .
        statementn;
        increment/decrement statement;
}
```

Following is a while equivalent of the above code.

```
<html>
        <head>
                <title>Associating a Function with a Link</title>
                <script language="JavaScript">
                <!--
                        var i=0;
                        while (i<10)
                        {
                                document.write(i+"<br>");
                                i=i+1;
                        }
                // -->
                </script>
        </head>
        </body>
</html>
```

Exercise:
What is the output of the following program?

```
<html>
```

```
<head>
<script language="javascript">

var hour;
var ampm;

for (hour=1; hour<24; hour++)
{
  if (hour < 12) {ampm="am"}
  if (hour >= 12) {ampm="pm"}

  if (hour < 13)
  {
    document.write(hour + ampm)
  }
  else
  {
    document.write((hour-12) + ampm)
  }
  document.write("<br>");
}

</script>
</head>

<body>
</body>
</html>
```

In do-while, a block of code is executed first before testing the condition.
Syntax is

```
do
{
      statement1;
      statement2;
      .
      .
      .
      statementn;
} (condition);
```

Statements inside do-while are executed at least once irrespective ([보통 다음 성구로]) of whether ([간접의문문의 명사절을 이끌어] ...인지 어떤지) condition evaluates to true or false.

Here is an example that uses do-while loop, In this example, we ask the user to enter any number between 1 – 12, and 0 to exit.

The output is name of month corresponding (상응하는, 일치하는, 대응하는) to a number. i.e. If user enters 1, output is January. if user enters 2, output is February. If user enters 12, output is December.

```
<html>
<head>
<script language="javascript">

var requested_month=0;
var MonthArray=
[
"",
"January",
"February",
"March",
"April",
"May",
"June",
"July",
"August",
"September",
"October",
"November",
"December"
]

document.write("Enter a a number between 1 and 12, and I will tell you the name of the month.<br>");
do
{
requested_month=prompt("Enter a number between 1 and 12 (or 0 to exit)", 0);
document.write(MonthArray[requested_month] + "<BR>");
} while (requested_month != "0")

</script>
</head>

<body>
</body>
```

Exercise:
Write the above program with using while loop.

Data Type Conversion (변하게 하다, 전환하다)

To understand what is data type conversion, and why we need data type conversion, let us take an example.

```
<html>
<body>
<script language="JavaScript" type="text/javascript">

var firstNumber = prompt("Enter the first number","");
var secondNumber = prompt("Enter the second number","");
var theTotal = firstNumber + secondNumber;
document.write(firstNumber + " added to " + secondNumber + " equals " +
   theTotal);

</script>
</body>
</html>
```

What is the output, if user enters firstNumber=2 and secondNumber=3, Output should be 5.

But if you will execute above program, you will be disappointed (실망시키다) to find that output is 23.

Why is output 23 instead of 5?
Ans: We know that function prompt () returns a string, and if we use '+' operator with strings, they are concatenated (사슬같이 잇다).

To solve (<문제 등을> 풀다, 해석하다) this problem we will take help of data type conversion. Data type conversion means convert data type.

What is the solution?
Solution is convert String to int or float. There are 2 conversion functions parseInt() and parseFloat().

parseInt(): This functions takes a string and convert it to an integer.
For example:

parseInt("123") = 123
parseInt("123Abc") = 123

parseFloat(): This function take a string and convert it to a real number.

What is the output for the following?
1. parseInt("ABC")
2. parseInt("ABC123")

Find it and I will ask you in the class.

Now, modify (수정하다) the above code so that it gives correct output i.e 5.

```
<html>
<body>
<script language="JavaScript" type="text/javascript">

var firstNumber = prompt("Enter the first number","");
var secondNumber = prompt("Enter the second number","");
var theTotal = parseInt(firstNumber) + parseInt(secondNumber);
document.write(firstNumber + " added to " + secondNumber + " equals " +
   theTotal);

</script>
</body>
</html>
```

What is NaN?

If you write document.write(parseInt("abc")), output will be NaN. NaN means **N**ot **a** **N**umber.

If you use parseInt() or parseFloat() and input argument is a string that is empty or doesn't start with a number, output is NaN.

There is a function in Javascript isNaN(x), which checks whether 'x' is NaN or not.
For example:

isNaN("123") returns false
isNaN("ABC") returns true
isNaN("123ABC") returns true

So, Before converting any String to a number we can check beforehand (미리) whether string consist (**(**부분·요소로**)** 되어**[**이루어져**]** 있다) of numbers or not. If string consist of numbers, we can convert it otherwise (**[**접속사적으로**]** 만약 그렇지 않으면) display an error message.

Below is a program which checks for user input. If user input is not valid, we generate an error message.

```
<html>
<body>
<script language="JavaScript" type="text/javascript">

var firstNumber = prompt("Enter the first number","");
var secondNumber = prompt("Enter the second number","");
var ifn,isn;
if (isNaN(firstNumber))
      alert("this is Not a Number");
else
      ifn = parseInt(firstNumber);
if (isNaN(secondNumber))
      alert("this is not a number");
else
      isn = parseInt(secondNumber);
var theTotal = ifn+ isn;
document.write(firstNumber + " added to " + secondNumber + " equals " +
   theTotal);
```

```
</script>
</body>
</html>
```

Multi Dimensional Array

In my previous lecture, I taught you about 1-dim arrays. 1-dim arrays are those that have only one index.
Like 1-dim arrays, there are 2-dim, 3-dim and multi-dim arrays.

2-dim arrays are those that have 2-dimensions.

First answer this question, What is the output of following code

```
<html>
<body>
<script language="JavaScript" type="text/javascript">
var a = new Array("a",2,"b",3,"c",4);
for (i=0;i<a.length;i++)
        document.write("a["+i+"] = "+a[i]+"<br>");
</script>
</body>
</html>
```

Two things are worth (**...**의 가치가 있는) notice (통지**,** 통보) in the above program

1. Unlike (같지 않은**,** 다른**,** 닮지 않은) arrays in other languages, arrays in Javascript can store values of different data types. For example:

```
var a = new Array("a",2,"b",3,"c",4);
```

2. I told in my previous lecture also, that new keyword create an object. This means, 'a' is an object of Array.
   And length is a variable of Array Object in Javascript, So a.length returns length of array 'a'.

To understand 2-dim arrays, Let us take an example.

Suppose, we want to store a company's employee information in an array. Each employee has
-name
-age
-address

One way to do this, is store the information in a 1-dim array sequentially (잇달아 일어나는**,** 연속하는**,** 순차적인). For example:

Index	Data Stored
0	Name1
1	Age1

Index	Data Stored
2	Address1
3	Name2
4	Age2
5	Address2
6	Name3
7	Age3
8	Address3

var employee = new
Array(“Huh”,23,”suwon”,”cho”,30,”hwaseong”,”Chong”,28,”Guro”,”chae”,40,”bundang”);

or

employee[0] = “huh”;
employee[1] = 23;
employee[2] = “suwon”;

employee[3] = “cho”;
employee[4] = 30;
employee[5] = “hwaseong”;

employee[6] = “chong”;
employee[7] = 28;
employee[8] = “guro”;

employee[9] = “chae”;
employee[10] = 40;
employee[11] = “bundang”;

employee is a 1-dim array, employee[0], employee[1], employee[2] stores information about one employee.
empoyee[3], employee[4], employee[5] stores information about 2nd employee.

2nd way of doing this, is store the information in a 2-dim arrays.

Index	0	1	2
0	Name1	Name2	Name3
1	Age1	Age2	Age3
2	Address1	Address2	Address3

var employee = new Array(); // employee is an array
employee[0] = new Array(); // employee[0] is also an array
employee[1] = new Array(); // employee[1] is also an array
employee[2] = new Array(); // employee[2] is also an array

The first index (0) belongs (**(...의)** 소유물이다) to the employee array; the second index (0) belongs to the employee[0] array.

employee[0][0] = “huh”;
employee[0][1] = “23”;
employee[0][2] = “suwon”;

employee[1][0] = “cho”;
employee[1][1] = “30”;
employee[1][2] = “hwaseong”;

employee[2][0] = “chong”;
employee[2][1] = “28”;
employee[2][2] = “guro”;

employee[3][0] = “chae”;
employee[3][1] = “40”;
employee[3][2] = “bundang”;

Here is the complete program

```
<html>
<body>

<script language="JavaScript" type="text/javascript">

var employee = new Array();

employee[0][0] = “huh”;
employee[0][1] = “23”;
employee[0][2] = “suwon”;

employee[1] = new Array();
employee[1][0] = “cho”;
employee[1][1] = “30”;
employee[1][2] = “hwaseong”;


employee[2] = new Array();
employee[2][0] = “chong”;
employee[2][1] = “28”;
employee[2][2] = “guro”;


document.write("Name : " + employee[1][0] + "<br>");
document.write("Age : " + employee[1][1] + "<br>");
document.write("Address : " + employee[1][2]);

</script>

</body>
</html>
```

Answer these?

```
var myArray = new Array(); Is it 1-dim or 2-dim array or 3-dim or 4-dim or 5-dim?
myArray[0] = new Array(); Is it 1-dim or 2-dim array or 3-dim or 4-dim or 5-dim?
```

```
myArray[0][0] = new Array(); Is it 1-dim or 2-dim array or 3-dim or 4-dim or 5-dim?
myArray[0][0][0] = new Array(); Is it 1-dim or 2-dim array or 3-dim or 4-dim or 5-dim?
myArray[0][0][0][0] = new Array(); Is it 1-dim or 2-dim array or 3-dim or 4-dim or 5-dim?
```

Javascript-An Object based language

In this topic, we will look at most important aspect of Javascript programing 'Objects'. **Javascript itself consist of Objects and browser also made of collection of object.**

For example:
One Javascript object that you have used is **Array**.
You have also used 2 objects **document** and **window**, these are objects provided by browser.
We can create our own objects.

A brief introduction about objects

Those who have been doing programming in C++ or Java already know about objects, I still fell the need to give an introduction about objects for those who are first time using it.

To understand what are objects, we take an example. We will take an example of car.

How do we define a car? We define a car as a blue car with 4 wheel drive, automatic gear, power steering, etc. Color, 4 wheel, automatic gear, 4 wheel drive, power steering are properties of a car. If I make an object of type Car, color, automatic gear, 4 wheel drive, power steering will be attributes or property (재산, 자산) of object. In Javascript, we represent attributes as properties for an object.

How do we use a car? We turn in the key, press the pedal, change the gear, and then move the steering. This can also be defined as behavior (행동, 거동, 행실, 품행) of car. In Javascript, we represent the behavior as methods or functions of an object.

We can pass arguments to a function, for example if gear() is a function, then one of its argument can be gear number, gear 1 or 2 or 3 or 4 or reverse.
Similarly, speed() function can return me the speed of the car.
Or OilLeft() function can return the amount of oil left in the car.

If we want to define an object for car, we need to defined attributes and functions for the car object. In Object based programming, we try to model real life things by an Object. And objects are defined by their attributes and functions.

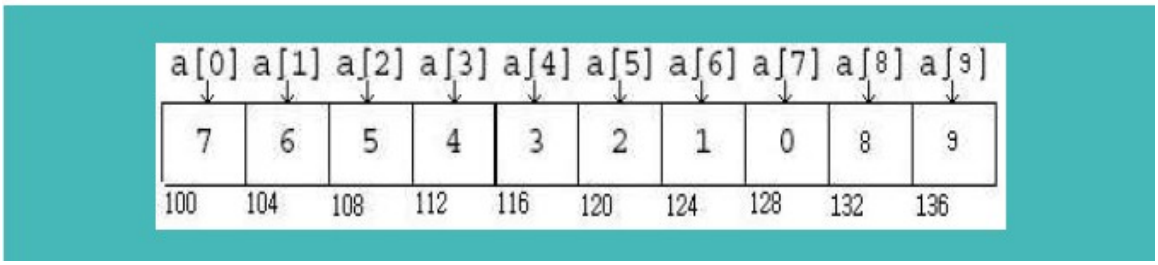
Basic definition of an Object is “A thing with attributes and functions”.

Take an example, We used array.length which returns length of the array, now array is an object and length is an attribute of array object. Length attribute of array object, returns number of elements stored in the array.

Let us do to anatomy (해부;해부학;해부술) of Array object.

We create an array by the statement
var num = new Array(7, 6, 5, 4, 3, 2, 1, 0, 8, 9);

'num' is a variable name. In the above example, right hand side of the statement create



an array in memory.

Starting address of the array is 100. 'num' stores only the starting address of array i.e. 'num' stores value 100.

new is a keyword in Javascript for creating object. In the above example, we are creating an object of type Array. Object is created in memory and its starting address is 100 which is stored in variable 'num'.

Let us take another example of using Date Object

```
var d = new Date("1 Jan 2000");
```

'd' is a variable which can store an integer, character, string etc, in the above example variable 'd' stores address of an object of type Date.

'new' is a keyword that creates an object. In the above example we are creating an object of type Date. The object of type Date takes some space in memory and has a starting address. 'd' stores starting address of object of type Date.

Let us try to understand more about objects and variables.
I am starting with an example of simple variables.

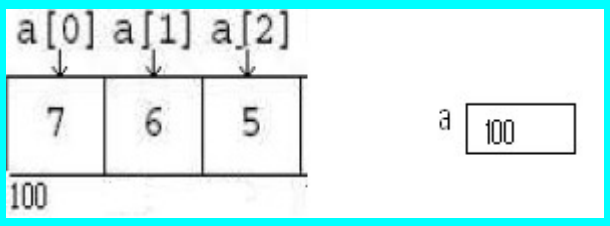
```
1. var a = 10;
2. var b = a;
3. document.write(a); // Output is 10
4. document.write(b); // Output is 10
5. b = 20;
6. document.write(a); // Output is 10
7. document.write(b); // Output is 20
```

a and b are 2 different variables.
Line 1, variable a stores value 10.
Line 2, variable b also stores the value value stored in a (10). So b also stores value 10.
Line 3 output is 10 and line 4 output is 20.

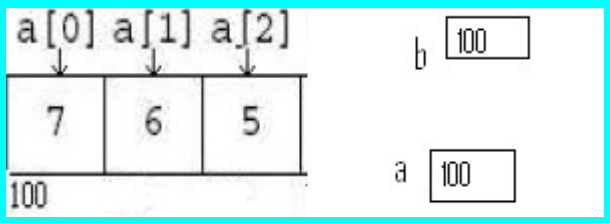
Line 5, we change the value stored in b. New value stored in b is 20 now. But value stored in a does not change which is still 10.
Line 6 output is 10, because value stored in a does not change.
Line 7 output is 20, because value stored in b changes from 10 to 20.

The above was a simple example using variables. Next we take an example using objects.

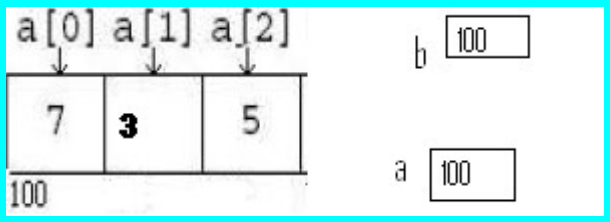
1. var a = new Array(7, 6, 5);



2. var b = a;



3. document.write(a[0] + " "+ a[1]+ " "+ a[2]); // output is 7 6 5
4. document.write(b[0] + " "+ b[1]+ " "+ b[2]); // output is yeh
5. b[1] = 3;



6. document.write(a[0] + " "+ a[1]+ " "+ a[2]); // output is 7 3 5
7. document.write(b[0] + " "+ b[1]+ " "+ b[2]);// output is 7 3 5

Line 1, creates an object of type array and stores the address of object in a. Suppose Object of type array has starting address 100, then a stores 100.

Line 2, variable b gets the value stores in a, which is 100. i.e. B = 100.

Line 3, output is 7 6 5.

Line 4, output is 7 6 5, because b also contain the address 100 which is same as address stored in a.

Line 5. We change the value if b[1] to 2, because a and b both contain the same address b[1] and a[1] refer to same location of memory.

Line 6 output is 7 3 5 and line 7 output is 7 3 5.

Using an Object Properties and Methods

Every object provide (지급하다) some methods and properties. For example, we can find the length of the array, For example:

var a = new Array(7 , 6 , 5);
var len = a.length; // output is 3

length is a property of Array object. It returns the length of the array.
Address of Array object is stored in the variable a. So a can access all the properties of Array object.

a . name of property

Simillary,

a . name of function(), here function can be replaced by actual function name.
For example
Date d = new Date("15 Aug 1953")
dd = d.getDate(); // dd = 31

getDate() is a function of Date Object.

Primitives and Objects

You should now have a good idea about the difference between primitive (원시의, 초기의; 태고의, 옛날의) data, such as numbers and strings, and object data, such as Dates and Arrays.

We have string, number and boolean data types. In fact there are String, Number, and Boolean objects corresponding (유사한) to the three string, number, and Boolean primitive data types. For example, to create a String object containing the text "I'm a String object," we can use

```
var myString = new String("I'm a String object");
```

The String object has the length property just as the Array object does. This returns the number of characters in the String object. For example

```
var lengthOfString = myString.length;
```

would store the data 19 in the variable lengthOfString.

But what if we had declared a primitive string called mySecondString holding the text "I'm a _primitive string" like this:

```
var mySecondString = "I'm a primitive string";
```

and wanted to know how many characters could be found in this primitive string?

So, for our primitive string mySecondString, we can use the length property of the String object to find out the number of characters it contains. For example

```
var lengthOfSecondString = mySecondString.length;
```

would store the data 22 in the variable lengthOfSecondString

If we declare a primitive string and then treat it as an object, such as by trying to access one of its methods or properties, JavaScript would know that the operation we're trying to do won't work if it's a primitive string. It will only work if it's an object; for example, it would be valid if it were a String object. **In this case, JavaScript converts the plain text string into a temporary String object, just for that operation**.

JavaScript Native Objects
Broswer Objects

JavaScript Native Objects

So far we have just been looking at what objects are, how to create them, and how to use them. Now, let's take a look at some of the more useful objects that are native to JavaScript, that is, those that JavaScript makes available for us to use.

We won't be looking at all of the native JavaScript objects, just some of the more commonly used ones, namely the String object, the Math object, the Array object, and the Date object.

String Objects

Like most objects, String objects need to be created before they can be used. To create a String object, we can write

```
var string1 = new String("Hello");
var string2 = new String(123);
var string3 = new String(123.456);
```

However, as we have seen, we can also declare a string primitive and use it as if it were a String object, letting JavaScript do the conversion to an object for us behind the scenes. For example

```
var string1 = "Hello";
```

The length Property

The length property simply returns the number of characters in the string. For example

```
var myName = new String("Paul");
document.write(myName.length);
```

will write the length of the string "Paul" (that is, 4) to the page.

The charAt() and charCodeAt() Methods—Selecting a Single Character from a String

The charAt(pos) method returns the character at position 'pos'.

charAt() treats the positions of the string characters as starting at 0, so the first character is at index 0, the second at index 1, and so on.

For example, to find the last character in a string, we could use the code

```
var myString = prompt("Enter some text","Hello World!");
var theLastChar = myString.charAt(myString.length - 1);
document.write("The last character is " + theLastChar);
```

In the first line we prompt the user for a string, with the default of "Hello World!" and store this string in the variable myString.

In the next line, we use the charAt() method to retrieve the last character in the string. We use the index position of (myString.length - 1). Why? Let's take the string "Hello World!" as an example. The length of this string is 12, but the last character position is 11 since the indexing starts at 0. Therefore, we need to subtract one from the length of the string to get the last character position.

In the final line, we write the last character in the string to the page.

The charCodeAt() method is similar in use to the charAt() method, but instead of returning the character itself, it returns a number that represents the decimal character code in the Unicode character set for that character.

For example, to find the character code of the first character in a string, we could write

```
var myString = prompt("Enter some text","Hello World!");
var theFirstCharCode = myString.charCodeAt(0);
document.write("The first character code is " + theFirstCharCode);
```

The above code will get the character code for the character at index position zero in the string given by the user, and write it out to the page.

Ascci values of character go in order so, for example, the letter A has the code 65, B has 66, and so on. Lowercase letters start at 97 (a is 97, b is 98, and so on). Digits go from 48 (for the number 0) to 57 (for the number 9).

Character	***Ascii value***
A	65
B	66
C	67
.	.
.	.
.	.
Z	91
a	97
b	98
.	.
.	.
z	123
0	48
1	49
.	.
.	.
9	58

```
<html>
<head>
<script language="JavaScript" type="text/javascript">
function checkCharType(charToCheck)
{
   var returnValue = "O";
   var charCode = charToCheck.charCodeAt(0);

   if (charCode >= "A".charCodeAt(0) && charCode <= "Z".charCodeAt(0))
   {
      returnValue = "U";

   }
   else if (charCode >= "a".charCodeAt(0) && charCode <= "z".charCodeAt(0))
   {
      returnValue = "L";
```

```
   }
   else if (charCode >= "0".charCodeAt(0) && charCode <= "9".charCodeAt(0))
   {
      returnValue = "N";
   }
   return returnValue;
}
</script>
<head>

<body>
<script language="JavaScript" type="text/javascript">

var myString = prompt("Enter some text","Hello World!");
switch (checkCharType(myString))
{
   case "U":
      document.write("First character was upper case");
      break;
   case "L":
      document.write("First character was lower case");
      break;
   case "N":
      document.write("First character was a number");
      break;
   default:
      document.write("First character was not a character or a number");
}
</script>
</body>
</html>
```

The fromCharCode() Method—Converting Character Codes to a String

The method fromCharCode() can be thought of as the opposite to charCodeAt(), in that you pass it a series of comma-separated numbers representing character codes, and it converts them to a single string.

However, the fromCharCode() method is unusual in that it's a *static* method—we don't need to have created a String object to use it with. Instead, we can use the String expression

For example, the following lines put the string "ABC" into the variable myString.

```
var myString;
myString = String.fromCharCode(65,66,67);
```

What is the output of the follwing code?

```
var myString = "";
var charCode;

for (charCode = 65; charCode <= 90; charCode++)
{
      myString = myString + String.fromCharCode(charCode);
}

document.write(myString);
```

The indexOf() and lastIndexOf() Methods—Finding a String Inside Another String

The methods indexOf() and lastIndexOf() are used for searching for the occurrence of one string inside another. A string contained inside another is usually termed a *substring*.

Both indexOf() and lastIndexOf() take two parameters:

- The string you want to find
- The character position you want to start searching from (optional)

If you dont specify the 2nd parameter, function starts searching from position 0.

The return value of indexOf() and lastIndexOf() is the character position in the string at which the substring was found. If the substring is found at the start of the string, then 0 is returned. If there is no match, then the value -1 is returned.

```
<script language="JavaScript" type="text/javascript">

var myString = "Hello paul. How are you Paul";
var foundAtPosition;

foundAtPosition = myString.indexOf("Paul");
alert(foundAtPosition);

</script>
```

This code should result in a message box containing the number 24, which is the character position of "Paul". You might be wondering why it's 24, which clearly refers to the second "Paul" in the string, rather than 6 for the first "paul". Well, this is due to case sensitivity again.

We've seen indexOf() in action, but how does lastIndexOf() differ? Well, whereas indexOf() starts searching from the beginning of the string, or the position you specified in the second parameter, and works towards the end, lastIndexOf() starts at the end of the string, or the position you specified, and works towards the beginning of the string.

```
<script language="JavaScript" type="text/javascript">

var myString = "Hello Paul. How are you Paul";
var foundAtPosition;

foundAtPosition = myString.indexOf("Paul");

alert(foundAtPosition);

foundAtPosition = myString.lastIndexOf("Paul");
alert(foundAtPosition);

</script>
```

There will be 2 outputs in this example, first output is 6. Because indexOf() starts searching from the beginning it will match “Paul” with the first Paul in string mystring which occurs at position 6. So 6 is returned.

Second output is 24, because lastIndexOf() starts searching from the end. It will start from the end moving towards beginning, in the way it finds “Paul” in the string myString at position 24. So 24 is returned.

The substr() Method—Copying Part of a String

If we wanted to cut out part of a string and assign that cut out part to another variable, we would use the substr() method.

The method substr() takes two parameters: the start position and the end position of the part of the string we want. The second parameter is optional; if you don't include it, all characters from the start position to the end of the string are included.

For example, if our string is "JavaScript" and we want just the text "Java", we could call the method like so:

```
var myString = "JavaScript";
var mySubString = myString.substr(0,4);
alert(mySubString);
```

Why the end position is 4? It helps to think of the parameters as specifying the *length* of the string being returned: the parameters 0 and 4 will return (4 - 0) number of characters starting at and including the character at position 0.

Let us take an example where I can use substr() and lastIndexOf() function together.

Suppose I want to extract the name of the file from the URL "http://mywebsite/temp/myfile.htm".

```
1. var fileName = window.location.href;
2. var pos =  fileName.lastIndexOf("\");
3. fileName = fileName.substr(pos+1);
4. document.write("The file name of this page is " + fileName);
```

Line1 stores "http://mywebsite/temp/myfile.htm" in variable fileName. window.location.href returns the URL from the addressbar of browser.

We can use lastIndexOf() function to find last occurrence of "\" in variable fileName. Line 2 find the last occurrence of "\" in variable fileName and stores the result in pos.

Then we extract the string following last "\". We use substr function for that in line3. Finally we print the extracted substring.

The toLowerCase() and toUpperCase() Methods—Changing the Case of a String

If we want to change the case of the string, javascript provides 2 functions for that toLowerCase() and toUpperCase.

```
var myString = "I Donot Care About Case".
window.alert(myString.toLowerCase());
window.alert(myString.toUpperCase());
```

There will be 2 outputs, first output will print "i dont care about case".

Second output will print "I DONT CARE ABOUT CASE".

Name of the function	*Description*
big:	Returns the string in big tags.
blink:	Returns the string in blink tags.
bold	Returns the string in b tags.
fixed	Returns the string in tt tags.
fontcolor	Returns the string in font tags with the color attribute set.
fontsize	Returns the string in font tags with the size attribute set.
italics	Returns the string in i tags.
small	Returns the string in small tags.
strike	Returns the string in strike tags.
sub	Returns the string in sub tags (subscript effect).
super	Returns the string in sup tags (superstring effect).

Below is the code to understand the above methods.

```
<body>

	<script language="JavaScript">
	<!--

		var str1="Hello";
		document.write("This is normal text"+"<br>");
		document.write(str1.big()+"<br>");
		document.write(str1.bold()+"<br>");
		document.write(str1.fixed()+"<br>");
		document.write(str1.fontcolor("blue")+"<br>");
		document.write(str1.fontsize(30)+"<br>");
		document.write(str1.italics()+"<br>");
		document.write(str1.small()+"<br>");
		document.write("How are you"+str1.sub()+"<br>");
		document.write("2"+str1.sup()+"<br>");
		document.write(str1.toLowerCase()+"<br>");
		document.write(str1.toUpperCase()+"<br>");
	// -->
	</script>

</body>
```

Output should be

This is normal text
Hello
Hello
`Hello`
Hello

Hello

Hello
Hello
How are you$_{\text{Hello}}$
2^{Hello}
hello
HELLO

If we want to appply more than one formatting option, you can specify them in a single statement. To understand this, refer to example below.

```
<body>
	<script language="JavaScript">
	<!--

		var str1="Hello";
		document.write("This is normal text"+"<br>");
		document.write(str1.big().bold().italics().toUpperCase());
	// -->
	</script>

</body>
```

Output should be
This is normal text
HELLO

Math Object

The Math object is a little unusual in that JavaScript automatically creates it for you. There's no need to declare a variable as a Math object or define a new Math object before being able to use it, making it a little bit easier to use.

Properties:

- Math.E

Math.LN10

- Math.LN2
- Math.LOG10E
- Math.LOG2E
- Math.PI

Functions

•Math.abs(x)	:	Returns the absolute value of x.
•Math.ceil(x)	:	Rounds a number up.
•Math.floor(x)	:	Rounds a number down.
•Math.random()	:	Returns a random number between 0 and 1.
•Math.round(x)	:	Math.round(25.9) = 26 and Math.round(25.2) = 25
•Math.sqrt(x)	:	Calculates the square root of x.
•Math.max(x,y)	:	Returns the larger of two numbers
•Math.min(a,b)	:	Returns the smaller of two number
•Math.pow(x,y)	:	Returns the y raise to the power x.
•Math.log(x)	:	Returns the natural log of x.
•math.exp(x)	:	Returns the exponential of x.
•Math.sin(x)	:	Returns the sine of x.
•Math.cos(x)	:	Returns the cos of x.
•Math.tan(x)	:	Returns the tan of x.
•Math.asin(x)	:	Returns the arc sine of x in radians
•Math.acos(x)	:	Returns the arc cosine of x in radians
•Math.atan(x)	:	Returns the arc tan of x in radians.

The properties of the Math object include some useful math constants, such as the PI property (giving the value 3.14159 and so on).

The abs() Method

The abs() method returns the absolute value of the number passed as its parameter. Essentially, this means that it returns the positive value of the number. So -1 is returned as 1, -4 as 4, and so on. However, 1 would be returned as 1 because it's already positive.

For example, the following code would write the number 101 to the page.

```
var myNumber = -101;
document.write(Math.abs(myNumber));
```

The ceil() Method

The ceil() method always rounds a number up to the next largest whole number or integer. So 10.01 becomes 11, and –9.99 becomes –9 (because –9 is greater than –10). The ceil() method has just one parameter, namely

the number you want rounded up.

For example, the following code writes two lines in the page, the first containing the number 102 and the second containing the number 101.

```
var myNumber = 101.01;
document.write(Math.ceil(myNumber) + "<br>");
document.write(parseInt(myNumber));
```

The floor() Method

Like the ceil() method, the floor() method removes any numbers after the decimal point, and returns a whole number or integer. The difference is that floor() always rounds the number down. So if we pass 10.01 we would be returned 10, and if we pass –9.99, we will see –10 returned.

The round() Method

The round() method is very similar to ceil() and floor(), except that instead of always rounding up or always rounding down, it rounds up only if the decimal part is .5 or greater, and rounds down otherwise.

For example

```
var myNumber = 44.5;
document.write(Math.round(myNumber) + "<br>");

myNumber = 44.49;
document.write(Math.round(myNumber));
```

would write the numbers 45 and 44 to the page.

Parameter	*parseInt() returns*	*ceil() returns*	*floor() returns*	*round() returns*
10.25	10	11	10	10
10.75	10	11	10	11
10.5	10	11	10	11
–10.25	–10	–10	–11	–10
–10.75	–10	–10	–11	–11
–10.5	–10	–10	–11	–10

The random() Method

The random() method returns a random floating-point number in the range between 0 and 1, where 0 is included and 1 is not. This can be very useful for displaying random banner images or for writing a JavaScript game.

```
<html>
<body>
<script language="JavaScript" type="text/javascript">
var throwCount;
var diceThrow;
for (throwCount = 0; throwCount < 10; throwCount++)
{
```

```
   diceThrow = (Math.floor(Math.random() * 6) + 1);
   document.write(diceThrow + "<br>");
}

</script>
</body>
</html>
```

We want diceThrow to be between 1 and 6. The random() function returns a floating-point number between 0 and just under 1. By multiplying this number by 6, we get a number between 0 and just under 6. Then by adding 1, we get a number between 1 and just under 7. By using floor() to always round it down to the next lowest whole number, we can ensure that we'll end up with a number between 1 and 6.

If we wanted a random number between 1 and 100, we would just change the code so that Math.random() is multiplied by 100 rather than 6.

The pow() Method

The pow() method raises a number to a specified power. It takes two parameters, the first being the number you want raised to a power, and the second being the power itself. For example, to raise 2 to the power of 8 (that is, to calculate 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2), we would write Math.pow(2,8)—the result being 256.

Number Object

As with the String object, Number objects need to be created before they can be used. To create a Number object, we can write

```
var firstNumber = new Number(123);
var secondNumber = new Number('123');
```

However, as we have seen, we can also declare a number as primitive and use it as if it were a Number object, letting JavaScript do the conversion to an object for us behind the scenes. For example

```
var myNumber = 123.765;
```

The toFixed() Method

The toFixed() method is new to JavaScript 1.5 and Jscript 5.5—so basically it's available in Netscape 6+ and IE 5.5+ only. The method cuts a number off after a certain point. Let's say we wanted to display a price after sales tax. If our price is $9.99 and sales tax is 7.5%, that means the after-tax cost will be $10.73925. Well, this is rather an odd amount for a money transaction—what we really want to do is fix the number to no more than two decimal places. Let's create an example.

```
var itemCost = 9.99;
var itemCostAfterTax = 9.99 * 1.075;
document.write("Item cost is $" + itemCostAfterTax + "<br>");
itemCostAfterTax = itemCostAfterTax.toFixed(2);
document.write("Item cost fixed to 2 decimal places is " + itemCostAfterTax);
```

The first document.write() will output the following to the page:

```
Item cost is $10.73925
```

However, this is not the format we want; instead we want two decimal places, so on the next line, we enter

```
itemCostAfterTax = itemCostAfterTax.toFixed(2);
```

We use the toFixed() method of the Number object to fix the number variable that itemCostAfterTax holds to two decimal places. The method's only parameter is the number of decimal places we want our number fixed to.

This line means that the next document.write displays

```
Item cost fixed to 2 decimal places is $10.74
```

Lecture 5

This lecture, will learn about

1. Browser Objects
2. Window Object
3. Document Object
4. Location Object
5. Navigator Object
6. History Object
7. Screen Object
8. Events

Not only is Javascript object-based, but the browser is also made up of objects. When Javascript is running in the browser, we can access the browser's objects in exactly the same way we used Javascript's native objects in the last chapter. But what kinds of objects does the browser provide for us to use?

The browser makes available to us number of objects. For example, there is a window object corresponding to the window of the browser. We have already been using two methods of this object, namely the alert() and prompt() methods. For simplicity we previously called these functions, but they are in fact methods of the browser's window object

Another object made available by the browser is the page itself, represented by the document object. Again, we have already used methods and properties of this object. We have also been using the write() method of the document object to write information to the page.

A variety of other objects exist, representing a lot of the HTML that we write in the page. For example, there is an img object for each <img> tag that we use to insert an image into our document.

The collection of objects that the browser makes available to us for use with Javascript is generally called the *Browser Object Model (BOM)*. Some books refer to BOM as DOM (Document Object Model).

But before starting with BOM, I would like to introduce events.

Connecting Code to Web Page Events

What are events?

Events occur when something in particular happens or when user initiates some thing on the browser. For example:

1. User clicking on a page – This is onClick event.
2. User clicking on a hyper link – This is onClick event
3. Moving mouse pointer over some text – This is onMouseOver event
4. Moving mouse pointer out of some text – This is onMouseOut event.

5. When you write a URL on address bar of browser and click enter, that loads a page – There is an event associated with loading of page called as onLoad.
6. When you leave a web page or close browser – There is an event associated with closing browser or leaving a web page called as onUnLoad.

These were some of the most common used events. Take a real life example: We want to make menu pop up after we take mouse over some menu item.

In the above scenario, taking the mouse over a menu item is a onMouseOver event. We can associate a function say f1()with onMouseOver event. Whenever user will take his mouse over menu item, f1() will be called. Obviously f1() will display popmenu.

Let's create a simple HTML page with a single hyperlink, given by the element <A>. Associated to this element is the Anchor object. One of the events the Anchor object has is the click event.

```
<html>
<body>
<A href="somepage.htm" name="linkSomePage">
      Click Me
</A>
</body>
</html>
```

When I click on hyperlink “Click me”, page somepage.htm is loaded on the browser. Now, if I want I can associate an event with this anchor tag. Whenever user clicks his mouse on “Click me”, an event is fired. Actually even in the above example, event is fired, but we havenot written the code to capture it.

```
<A href="somepage.htm" name="linkSomePage" onClick = “capture_click()”> Click Me </A>
```

Above line means, when user clicks his mouse on “Click Me”, 2 things happen

1. onClick event is fired, which is also captured and hence function capture_click() is called.
2. Load the page somepage.html.

Below is a complete program for capturing onClick event associated with <a>.

```
<html>
<head>
        <script language="JavaScript">
                function capture_click()
                {
                        alert("User clicked his mouse on Click Me");
                }
        </script>
</head>
<body>
<A href="somepage.htm" name="linkSomePage" onClick="capture_click()">
        Click Me
</A>
```

```
</body>
</html>
```

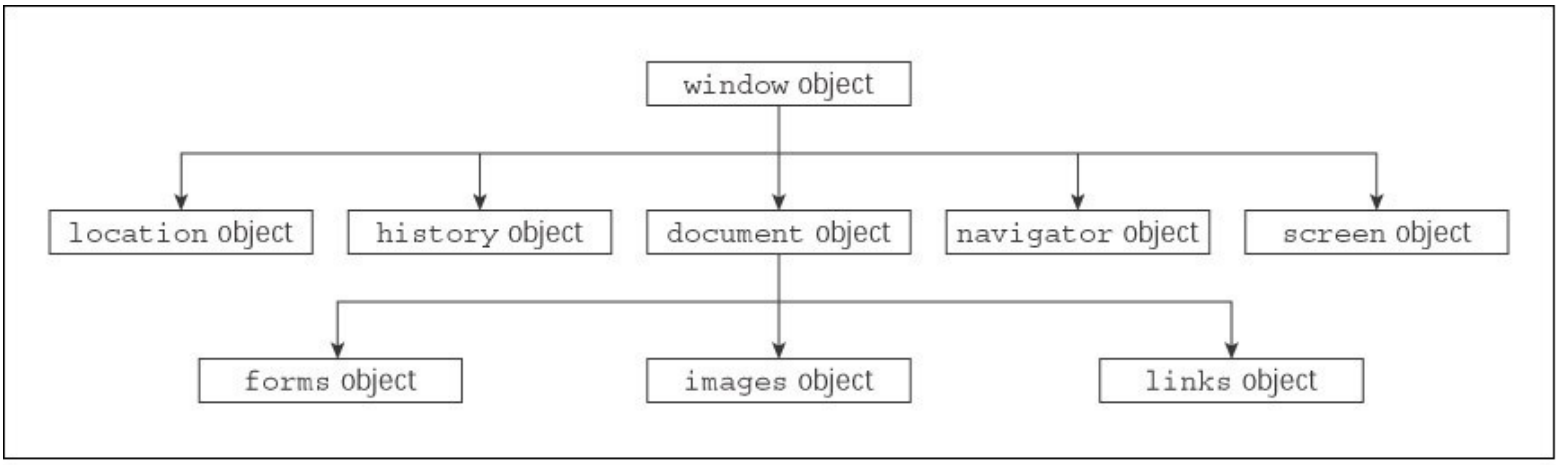
I can also associate onMouseOver event with the above anchor tag. I just need to write onMouseOver instead of OnClick.

```
<html>
<head>
	<script language="JavaScript">
		function capture_Mouseover()
		{
			alert("mouse over me");
		}
	</script>
</head>
<body>
<A href="somepage.htm" name="linkSomePage" onMouseOver="capture_Mouseover()">
	Click Me
</A>
</body>
</html>
```

Later we will see use of onMouseOut event with images object. We will also see usage of onLoad and onUnLoad event.

Introduction to Browser Objects

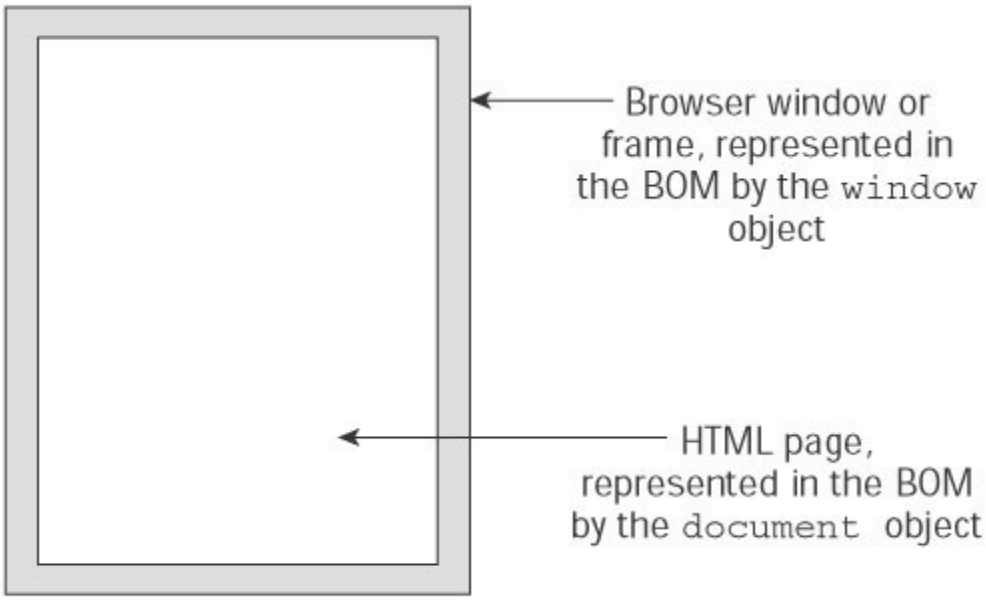
When Javascript is running in a web page, it has access to a large number of other objects made available by the web browser. Unlike native objects, we do not need to create browser objects. They are available to use through the browser.



The BOM has a hierarchy. At the very top of this hierarchy is the window object. You can think of this as representing the frame of the browser and everything associated with it, such as the

scrollbars, navigator bar icons, and so on.

Contained inside our window frame is the page. The page is represented in the BOM by the document object. You can see this in the figure below.



- If we want to make some change in the window like width, height etc, we will use window object (We will learn this later).
- If we want to make some change in page like background color, text color, images etc, we will use document object.

The Window Object

The window object represents the browser's frame or window in which your web page is contained

Below is the list of some of the things that we can do using the properties of Window Object.

1. We can find what browser is running.
2. The pages the user has visited.
3. The size of the browser window.
4. The size of the user's screen, and much more.
5. To access and change the text in the browser's status bar.
6. Change the page that is loaded.
7. Open new windows.

The above list is not complete, I have listed few things that you can do using properties available through Window Object.

The window object is a *global object*, which means we don't need to use its name to access its properties and methods. For example, the alert() function we have been using since the beginning is, in fact, the alert() method of the window object. Although we have been using this simply as

```
alert("Hello!");
```

We could write

```
window.alert("Hello!");
```

However, since the window object is the global object, it is perfectly correct to use the first version.

Some of the properties of the window object are themselves objects. These are

1. document – represents our page.
2. navigator – holds information about the browser, like type and version of browser.
3. history – contains the name of pages visited by user.
4. Screen – contains information about the size of screen like width, height, resolution etc.
5. location – contains the name and location of currently open web page.

To display a message in the window's status bar, we need to use the window object's defaultStatus property. To do this we can write

```
window.defaultStatus = "Hello and Welcome";
```

or, because the window is the global object, we can just write

```
defaultStatus = "Hello and Welcome";
```

Here is a program for the same.

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">
window.defaultStatus = "Hello and Welcome";
</script>
</head>
</html>
```

The History Object

The history object keeps track of each page that the user visits. It enables the user to click the browser's Back and Forward buttons to revisit pages. We have access to this object via the window object's history property.

We can either use window.history or simply history.

- Like the native JavaScript Array object, the history object has a length property. This can be used to find out how many pages are in the history array.
- The history object has the back() and forward() methods. When they are called, the location of the page currently loaded in the browser is changed to the previous or next page that the user has visited.
- The history object also has the go() method. This takes one parameter that specifies how far forward or backward in the history stack you want to go. For example, if you wanted to return the user to the page before the previous page, you'd write

```
history.go(-2);
```

To go forward three pages, you'd write

```
history.go(3);.
```

Note that go(-1) and back() are equivalent, as are go(1) and forward().

Below is a program that demonstrates use of history object.

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">

        function goBack()
        {
                alert("back");history.back();
        }
```

```
        function goNext()
        {
                alert("forward");history.forward();
        }
</script>
</head>
<body>
<a onClick = goBack()> Back</a> <br>
<a onClick = goNext()> Next </a>
</body>
</html>
```

The location Object

The location object contains lots of useful information about the current page's location. It contain the URL (Uniform Resource Locator) for the page, but also the server hosting the page, the port number of the server connection, and the protocol used. This information is made available through the location object's href, hostname, port, and protocol properties.

1. location.href – returns the URL of the page that is currently open in the browser
2. location.hostname – returns the hostname part of URL that is currently open in the browser.
3. location.port – returns the port number, http by default uses port 8080.
4. location.protocol – returns the protocol used for accessing internet, we currently use HTTP or ftp etc.

Below is a program that returns the URL that is currently open in the browser. Then user is prompted with a question to enter a new URL. Our program will receive user answer and open a new URL.

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">

        function getURL()
        {
                aURL = window.location.href;
                alert("URL of your browser is " + aURL);
                uURL=window.prompt("Enter the url of the page that you want to open");
                window.location.href = "http://" + uURL;
        }



</script>
</head>
<body>
<a onClick = goBack()> Back</a> <br>
<a onClick = goNext()> Next </a>
</body>
</html>
```

The Screen Object

The screen object property of the window object contains a lot of information about the display capabilities of the client machine. Its properties include the height and width properties, which indicate the vertical and horizontal range of the screen in pixels. Another property of the screen object, is the colorDepth property. This tells us the number of bits used for colors on the client's screen.

1. window.screen.width – It returns the width of user computer screen.

2. Window.screen.height- It returns the height of user computer screen.
3. Window.screen.colorDepth – It returns the number of bits used for each pixel. It returns a value of 1, 4, 8, 15, 16, 24, or 32. 1 means, each pixel on your screen is composed of 1 bit and can display one of the $2^1 = 2$ colors. Similarly, if 8 is returned, that means, each pixel on your screen is composed of 8 bits and can display one of the $2^8 = 256$ colors

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">

	function getWindowProperties()
	{
		windowProperties = "Width of browser is "+ screen.width + " pixels";
		windowProperties += ", Height of browser is " + screen.height + " pixels";
		windowProperties += " and Color depth of screen is" + screen.colorDepth + " bits";

		alert(windowProperties);
	}


</script>
</head>
<body>
<a onClick = "getWindowProperties()"> Get Window Properties</a> <br>
</body>
</html>
```

Using the document Object

We've already come across some of the document object's properties and methods, for example the write() method. We will be learning more about document Object. We can do a lot with document Object, I will start with a very simple example of document object i.e. Setting the background color and gradually we will learn more about using document object.

Below is a program which sets the background color of the page using bgColor property of document Object. By default background color of web page is white. We will provide three color options to user pink, skyblue, yellow. User clicks on the color and the screen background color changes.

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">

	function setColor(colo)
	{
		switch(colo)
		{
			case 1 : document.bgColor = "pink";break;
			case 2 : document.bgColor = "skyblue";break;
			case 3 : document.bgColor = "yellow";break;
		}
	}
</script>
</head>
<body>
Change Colo:    <a onClick = "setColor(1)" href="#"> pink</a>  <a onClick = "setColor(2)"
href="#"> skyblue</a>  <a onClick = "setColor(3)" href="#"> yellow</a>
</body>
</html>
```

We have used document.bgColor property to set the background color of web page.

Now let's look at some of the slightly more complex properties of the document object. These properties have in common the fact that they all contain arrays. We will be discussing images, and links array. Images and Links array are properties of document object.

The images Array

As we know, we can insert an image into an HTML page using the following tag:

```
<img ALT="USA" name=myImage src="usa.gif">
```

The browser makes this image available for us to script in JavaScript by creating an img object for it with the name myImage. In fact, each image on your page has an img object created for it.

Each of the img objects in a page is stored in the images[] array. This array is a property of the document object. The first image on the page is found in the element document.images[0], the second in document.images[1], and so on

```
<HTML>
<HEAD>
<TITLE> Open a New Document </TITLE>
<script language="JavaScript">
        function display_wandh()
        {
                window.alert(document.images.length);
                n=document.images.length
                for(i=0;i<n;i++)
                        window.alert("name="+document.images[i].name+", width="+document.images[i].width+",
height="+document.images[i].height);
        }
</script>
</HEAD>
        <img src="book.gif" name="book" width=100 height=100>  <br>
        <img src="music.gif" name="music" width=200 height=300> <br>
        <img src="soccer.gif" name="soccer" width=100 height=200> <br>

<BODY>
<a href="#" onClick="display_wandh()"> Display Width and Heigh of all the image </a> <br>
</BODY>
</HTML>
```

In the following code, as you move the mouse over the image, image changes, and as soon as you take the mouse out of the image, same image appears again.

```
<HTML>
<HEAD>
<TITLE> Open a New Document </TITLE>
```

```
<script language="JavaScript">
	function changeImage()
	{
		book.src="music.gif";
	}

	function reinstate()
	{
		book.src="book.gif";
	}
</script>
</HEAD>
	<img src="book.gif" name="book" width=100 height=100 onMouseOver="changeImage()"
onMouseOut="reinstate()">
<BODY>
</BODY>
</HTML>
```

Below is a program that shows image randomly. Now, each time you move the mouse over the image, you will see a different image.

```
<HTML>
<HEAD>
<TITLE> Open a New Document </TITLE>
 </HEAD>
<BODY>
<script language="JavaScript">
	 function changeImage()
	{
		var myImage=new Array(4);
		myImage[0] = "soccer.gif";
		myImage[1] = "asia.gif";
		myImage[2] = "music.gif";
		myImage[3] = "book.gif";

		var rand_number = Math.absolute(Math.random() * 4);
		document.image.src=myImage[rand_number];
	}
```

```
</script>
        <img src="book.gif" onMouseOver="changeImage()" onMouseOut="changeImage()" name="image">
</BODY>
</HTML>
```

Using Links Object

We defined an event with anchor tag <a>, using onClick = “name_of_function()”. Like Javascript makes an array of images for each image on web page. Similarly, Javascript also makes an arraty of links for each <a> on web page.

First anchor tag is referred as document.links[0]

second anchor tag is referred as document.links[1] and so on.

```
<html>
<body>
<script language="JavaScript">
function linkSomePage_onclick()
{
   alert('This link is going nowhere');
   return false;
}
</script>

<A href="somepage.htm" name="linkSomePage">
   Click Me
</A>

<script language="JavaScript" type="text/JavaScript">
   window.document.links[0].onclick = linkSomePage_onclick;
</script>
</body>
</html>
```

Lecture6

This lecture is a continuation of previous lecture. In the previous lecture, we studied about following window objects

- screen object.
- images object
- links object
- history object
- location object.

This lecture we will study about last window object which is navigator object.

Due to so many browsers available, there is a problem that arises of browser compatability. Browser compatability means, a CSS or Javascript code works perfectly in one browser doesnot seem to render the same effect on other browser. This creates a real headache for developers.

Every developer should make sure that web pages developed by him should work perfectly across all browsers Or he can display a message suggesting the user to upgrade his browser.

To display a message for the user, developer should be able to determine what sort of browser is being used by user? This is where navigator objects come for help.

Actually there are 2 ways to determine if our code will work in user browser, Two ways are

1. Checking browser by feature
2. Using Navigator object.

Checking browser by feature

Lets take an example, if you have 3 guys in front of you, and you need to determine which country each guy belong to. You can do that by looking at the features of each guy.
For example: We have 2 guys, an american, chinese and south african

```
if (skin color == white)
{
	type = american and chinse.
	If (eyes = small)
		type = chinese;
	else
		type = american;

}
else
	type  = south african;
```

The above code can determine if a guy is an american or chinese or south african.

We determine the user browser, by checking the properties. For example, document.all property is supported only by IE 4.0 and higher and not by netscape browser.

```
if (document.all)
{
	browser= "IE";
	version = "4.0";
}
```

But the above statement is true for IE 4.0, IE 5.0 and IE 6.0 but false for all Netscape browser.

There is one property which is not supported by IE 4.0 and it is window.clipboardData. Property window.clipborardata returns false on IE 4.0 browser but returns true on IE 5.0 and IE 6.0

```
if (document.all)
{
	browser= "IE";
	version = "4.0";

	if (window.clipboardata)
		version = "5.0";
}
```

Similarly, document.layers property is only supported by Netscape browser 4.0 and higher and not by IE browsers.

```
if (document.layers)
{
	browser = "NN";
	version = "4.0";
}
```

Property document.layers return false for IE browser, but returns true for all Netscape browser. To check further about the version number, we use property window.sidebar, which returns true for version 6.0 and returns false for version 4.0.

```
if (document.layers)
{
	browser = "NN";
	version = "4.0";
	if  (window.sidebar)
		version = "6.0 +";
```

}

So, the complete code is

```
<html>
<body>

<script>
var browser = "Unknown";
var version = "0";

// NN4+
if (document.layers)
{
        browser = "NN";
        version = "4.0";

       if (window.sidebar)
       {
              version = "6+";
       }
}
else if (document.all)
{
       browser = "IE";
       version = "4";

       if (window.clipboardData)
       {
              browser = "IE"
              version = "5+"
       }

}
document.write(browser + " " + version);

</script>

</body>
</html>
```

Using Navigator Object

Lets focus on the problem of finding the nationality. One way of finding nationality of a guy is by looking at his features. Another method can be by just looking at his passport. Each guy must have a passport and by looking inside passport we can find his nationality.

But there is one problem with this approach. Passport can be illegaly made.

Similarly,We can use navigator object to know the type of browser being used by the user. But some browsers that are not popular, disguise themselves as IE or NN.

Remember that navigaor object is a property of window object.

navigator.appName – this property returns the model of browser i.e. IE or NN etc.
navigator.userAgent – this property returns lot of information such as the browser version, operating system, and browser model.

Code below finds the browser name, version.

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">

function getBrowserName()
{
  var lsBrowser = navigator.appName;
  if (lsBrowser.indexOf("Microsoft") >= 0)
  {
    lsBrowser = "MSIE";
  }
  else if (lsBrowser.indexOf("Netscape") >= 0)
  {
    lsBrowser = "NETSCAPE";
  }
  else
  {
    lsBrowser = "UNKNOWN";
  }
  return lsBrowser;
}
function getOS()
{
  var userPlat = "unknown";
  var navInfo = navigator.userAgent;
  if ((navInfo.indexOf("windows NT") != -1)
      || (navInfo.indexOf("windows 95") != -1 )
      || (navInfo.indexOf("windows 98") != -1 )
      || (navInfo.indexOf("WinNT") != -1 )
      || (navInfo.indexOf("Win95") != -1 )
      || (navInfo.indexOf("Win98") != -1 ))
  {
    userPlat = "Win32";
  }
  else if(navInfo.indexOf("Win16") != -1)
  {
```

```
    userPlat = "Win16";
  }
  else if(navInfo.indexOf("Macintosh") != -1)
  {
    userPlat = "PPC";
  }
  else if(navInfo.indexOf("68K") != -1)
  {
    userPlat = "68K";
  }
  return userPlat;
}
function getBrowserVersion()
{
  var findIndex;
  var browserVersion = 0;
  var browser = getBrowserName();
  if (browser == "MSIE")
  {
  browserVersion = navigator.userAgent;
  findIndex = browserVersion.indexOf(browser) + 5;
  browserVersion = parseInt(browserVersion.substring(findIndex,findIndex + 1));

  }
  else
  {
    browserVersion = parseInt(navigator.appVersion.substring(0,1));
  }
  return browserVersion;
}

</script>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">

var userOS = getOS();
var browserName = getBrowserName();
var browserVersion = getBrowserVersion();
if (browserVersion < 4 || browserName == "UNKNOWN" || userOS == "Win16")
{
  document.write("<h2>Sorry this browser version is not supported</h2>")
}
else if (browserName == "NETSCAPE")
{
  location.replace("NetscapePage.htm");
}
else
{
```

```
  location.replace("MSIEPage.htm");
}

</script>
<noscript>
  <h2>This website requires a browser supporting scripting</h2>
</noscript>
</body>
</html>
```

The problem is that less known browsers disguise themselves as IE or NN but donot support full functionality of IE or NN. So this method of finding type of browser is not used. 1st metod is more popular.

Project

Title: Make a home page for yourself

What your home page should contain:

- ➔ Home Page, which contains a Menu
- ➔ Menu contains at least 4 menu items (You decide about name of menu items)
- ➔ A web page for each menu item

Now the most important part, Your project will be graded on the following basis

- ➔ Creativity: This means, how you have managed to place graphics and text on the page. You page should have a professional touch
- ➔ Amount of information available on the home page: Well, you home page should not look too much populated and not even too empty.
- ➔ Color combination chosen
- ➔ How well you have used JavaScript (You have to decide where to use the JavaScript)
- ➔ There should be a feed back form.

To make this project, browse the internet, and go through the web sites of popular people, and try to get the idea about how to make a home page.