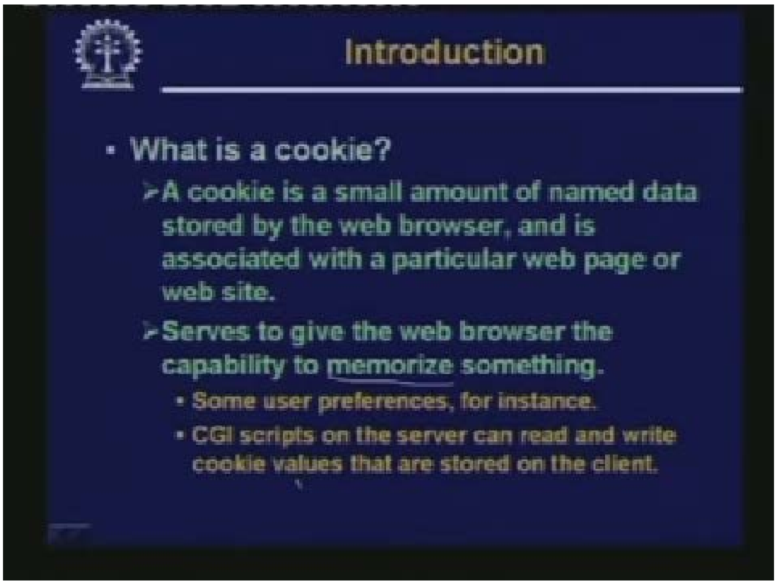
Internet Technology
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No #27
Using Cookies

Now we shall be continuing our discussion on Javascript. Now if you recall our last two classes, we had talked about the basic object-oriented features of the language Javascript. How a Javascript program executes on the browser in the internet environment? And we shall also looked into a number of examples where we saw how we can use Javascript to provide a number of pretty useful features which can make a web page interactive or more attractive in general. So today we shall be starting our discussion on something called cookies. How we can use cookies along with Javascript? So the first thing is to define a cookie.

(Refer Slide Time: 01:52)

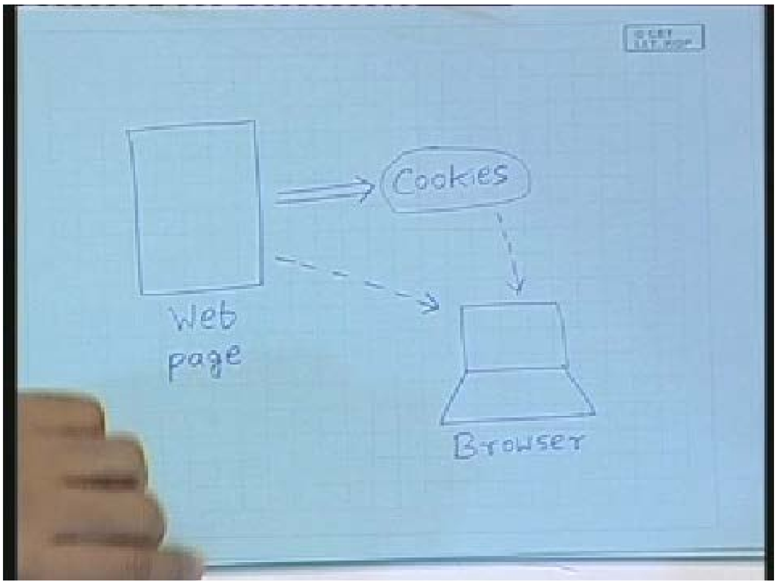


Introduction

- What is a cookie?
  - A cookie is a small amount of named data stored by the web browser, and is associated with a particular web page or web site.
  - Serves to give the web browser the capability to memorize something.
    - Some user preferences, for instance.
    - CGI scripts on the server can read and write cookie values that are stored on the client.

What is a cookie? Now a cookie by definition is a small amount of named data stored by the web browser and is associated with a particular web page or web site. So here there are a few things we need to understand. Number one a cookie is supposed be something which is regarded as a named data. A data with an assigned name to it and it is also associated with some web page. Actually the concept is like this.

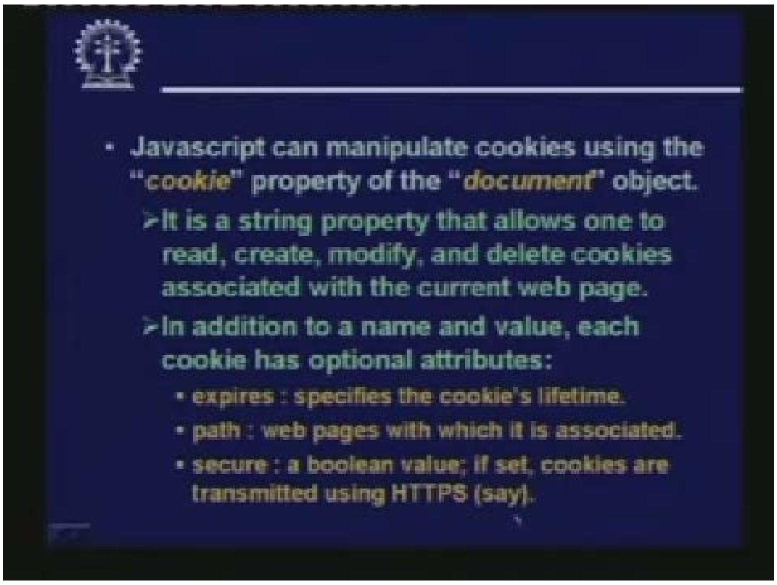
(Refer Slide Time: 02:35)



Suppose we have a particular web page. This web page has been downloaded from site and is being displayed on your browser. Now what I am saying is that along with this web page you define, such named data called cookies. These cookies also get downloaded onto your so called web client or the browser. If this is browser, then along with the web client, [03:18 audio not clear] web page cookies also get downloaded and stored on the browser. Now these cookies can serve the purpose of maintaining sessions, conveying useful information across pages and so on. We shall see some of the typical usages of cookies later. But first let us try to understand how such name value pairs can be stored on the browser. On the browser machine on the machine which I s running the browser and how the values can be accessed. So continuing with the definition, a cookie basically serves to give or provide the web browser the capability to memorize something.

Here since we are saying that cookies are some data which we are storing in the browser itself. What this means is that, the browser can use this cookies for the purpose of memorizing something. Some of the typical things that web browser may like to memorize maybe some user preferences which some web sites do use. Like some web sites try the maintain, the usage profile of a particular user. And depending on the profile certain choices or selections or certain popular options that are often exercised by the users are considered by default. This means that you a have customized menu or customized page depending on your choices or the options y have exercised in the recent past cookies can help you to do that. And the second thing is that some server-side scripts like the CGI scripts have the capability to read and write the cookie values stored on the client. So cookies are something which are stored on the client machine. But some server-side scripts can have access to those cookies and can read the values of cookies.

(Refer Slide Time: 05:33)



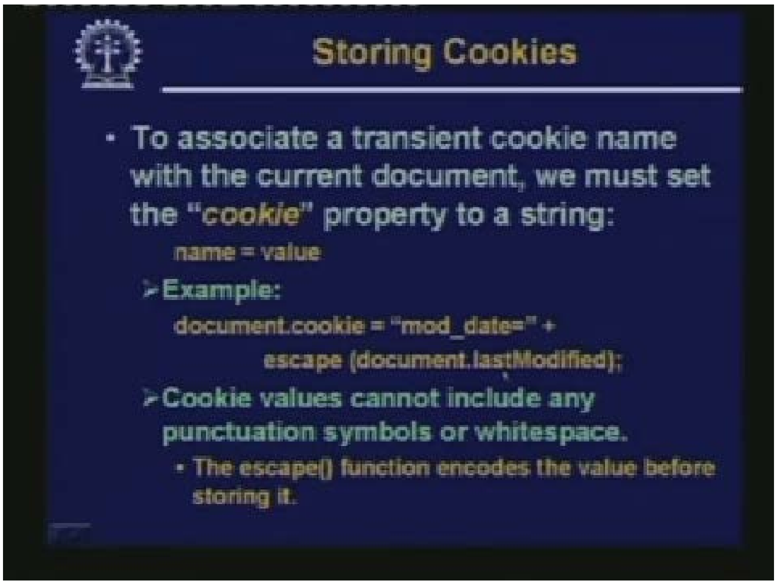
- Javascript can manipulate cookies using the "*cookie*" property of the "*document*" object.
  - It is a string property that allows one to read, create, modify, and delete cookies associated with the current web page.
  - In addition to a name and value, each cookie has optional attributes:
    - expires : specifies the cookie's lifetime.
    - path : web pages with which it is associated.
    - secure : a boolean value; if set, cookies are transmitted using HTTPS (say).

Now in Javascript we can manipulate cookies in a fairly simple way. We use the document object for the purpose. Document object as we shall see later the different properties that are associated with this. Cookie is one of the properties. In the document object which if you recall it actually signifies the document that is presently being displayed in the selected window cookie is a property which in fact stores or contains the values of these named data that we are storing on the browser machine. Now the cookie property is basically a string this property nothing but a string which can allow a Javascript code to read write modify and delete the so called cookies associated with the current web page. Now a cookie has a number of properties.

A cookie will be having a name and a value some typical cookie maybe say I have downloaded a page I want to store who is the author of the page that I can store as the name of page that is name of the cookie and the actual name of the author which is the value of the cookie so as the name value pair I can store some information about the page for example. So in addition to the name and value a cookie can have several other attributes like we can specify how long a cookie would be valid. We can specify the cookies life time whether it is valid only throughout the session. When I am login out of the page will the cookie be destroyed or will it remain across sessions. It also mentions a path a path will tell which pages will have this particular cookie associated with them.

And thirdly there is a property called secure. This is a Boolean value this can have the value either true or false if it is true if set, then cookies are transmitted using secure HTTP HTTPS. For example, see the idea is here like this in many cases the information this is stored in the cookie, can be pretty confidential. It can be as confidential as the user name and password it can also be. For ex instance be your credit card number. Some web sites if it choices to design the pages in that they can store this kind of information in the form of cookies. But these cookies you can set or mark as secure so that whenever the server-side script asks for the value of these cookies and these cookies need to be transferred back to the server they will be transferred in an encrypted form. And this encrypted form of transfer is guaranteed by having the so called secure HTTP links HTTPS.

(Refer Slide Time: 08:59)



Storing Cookies

- To associate a transient cookie name with the current document, we must set the "*cookie*" property to a string:

  name = value

  - Example:

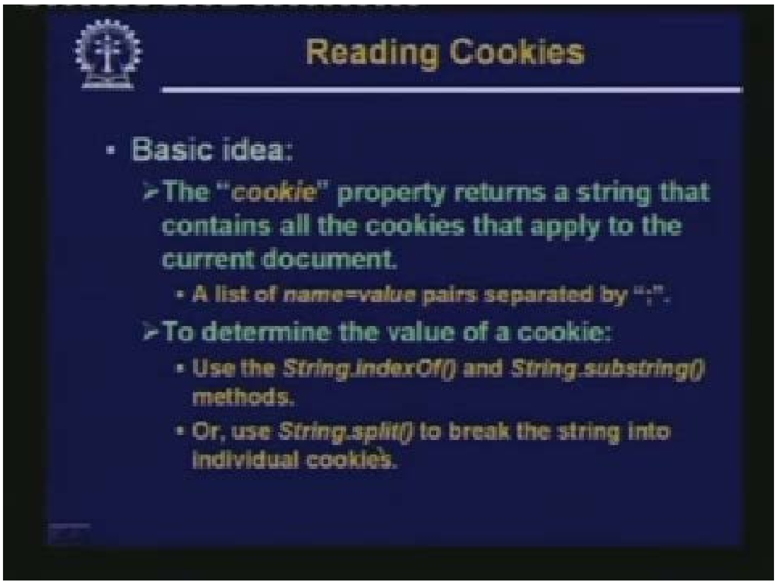
    ```
    document.cookie = "mod_date=" +
             escape (document.lastModified);
    ```

  - Cookie values cannot include any punctuation symbols or whitespace.
    - The escape() function encodes the value before storing it.

Now let us see how we can store cookies on the browser. You associate the cookie with the current document as I had said. We must use the cookie property of the document object. Now the cookie value is specified as a string of this form name equal to value. An example is shown here. For the present document, document.cookie equal to I can write first a string this is the name I am giving to the cookie mod underscore date equal to. This is the string concatenated with a function escape which contains document.lastModified. Here actually I am trying to store a cookie which will store the modification date of the document in a cookie whose name is mod underscore date.

Now the actual the lastModified value of the document can be retrieved from the lastModified attribute of the document. And the purpose of this escape fun is to remove all spaces and special characters from this lastModified string. Because one thing is to be remembered that a cookie value cannot include any punctuation symbols or white spaces. If there are white spaces or any kind of punctuation symbols they have to be converted into their equivalent hexadecimal values using escape sequences. The escape function does exactly that.

(Refer Slide Time: 10:52)

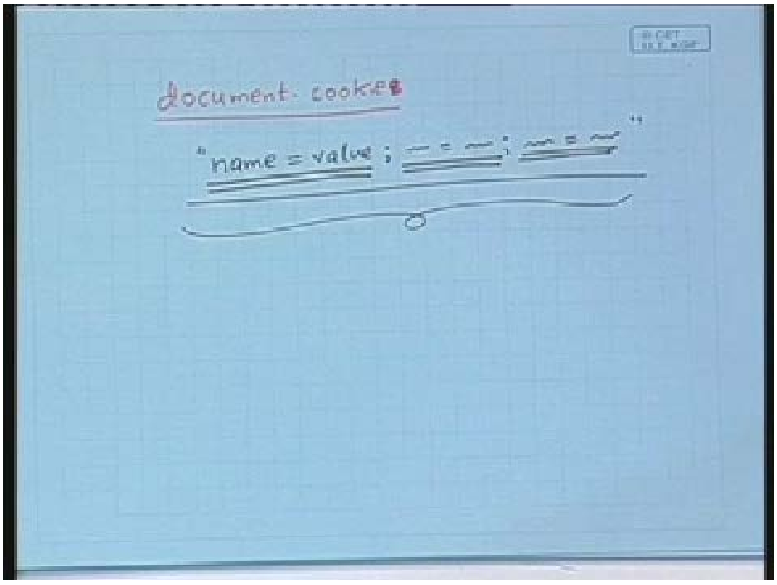


Reading Cookies

- Basic idea:
  - The "*cookie*" property returns a string that contains all the cookies that apply to the current document.
    - A list of *name=value* pairs separated by ";".
  - To determine the value of a cookie:
    - Use the *String.indexOf()* and *String.substring()* methods.
    - Or, use *String.split()* to break the string into individual cookies.

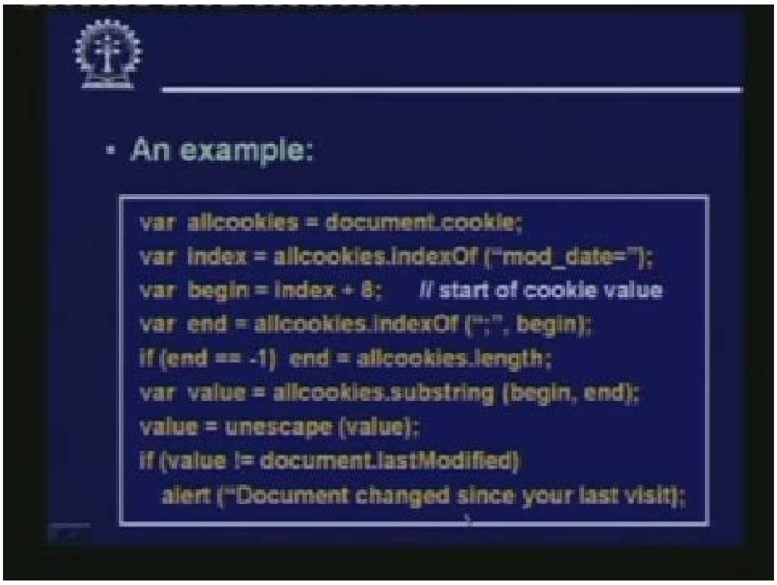
Now after you have stored the cookie you can retrieve the cookies in a similar way by accessing the cookie property. Now whenever you access the cookie property of the document, what you will be getting is a string which will contain all the cookies that apply to the current document, a list of name value pairs separated by semicolon. So actually what we mean here is that whenever you access cookies property you get something like this.

(Refer Slide Time: 11:27)



Suppose you access document.cookies. Cookie actually is what you get is this. You will be getting a string which will consist of name value pairs name equal to value semicolon; some other name equal to some other value semicolon; some other name equal to some other value. In this way it will continue. So this string will contain all the cookies. They will be delimited by this semicolon and in the cookies the name of the cookie will be first then an equal to sign then the value of cookie. So this document.cookie will return this entire character string. The whole cookie list which is delimited by semicolons. Now let us see how we can access or determine the value of a particular cookie. Now we shall see an example. Now we shall be using the indexOf and this split functions that are available on string objects. Javascript supports a number of methods on string objects. We shall be using indexOf and split. See indexOf is basically a function or a method which searches for a particular substring within a string and returns the starting point of that substring and the string; and similarly split. Split can break up a string into individual substrings an array of substring. We shall see this.

(Refer Slide Time: 13:16)



- An example:

```
var allcookies = document.cookie;
var index = allcookies.indexOf ("mod_date=");
var begin = index + 8;     // start of cookie value
var end = allcookies.indexOf (";", begin);
if (end == -1)  end = allcookies.length;
var value = allcookies.substring (begin, end);
value = unescape (value);
if (value != document.lastModified)
   alert ("Document changed since your last visit);
```

Now the example is like this. First we are retrieving this document.cookie. So here as I had said all the cookies delimited by semicolon in the form of a long character string is returned. It is returned into a variable called allcookies. Next we are invoking this. This allcookies is a string. We are invoking the indexOf method and we are searching for this string mod underscore date equal to. Because, this is the name of the cookie we had given and this is the string we are searching for in the cookie. So some where we would be getting this particular string and that index is returned by this function. So we are storing it in the variable index.

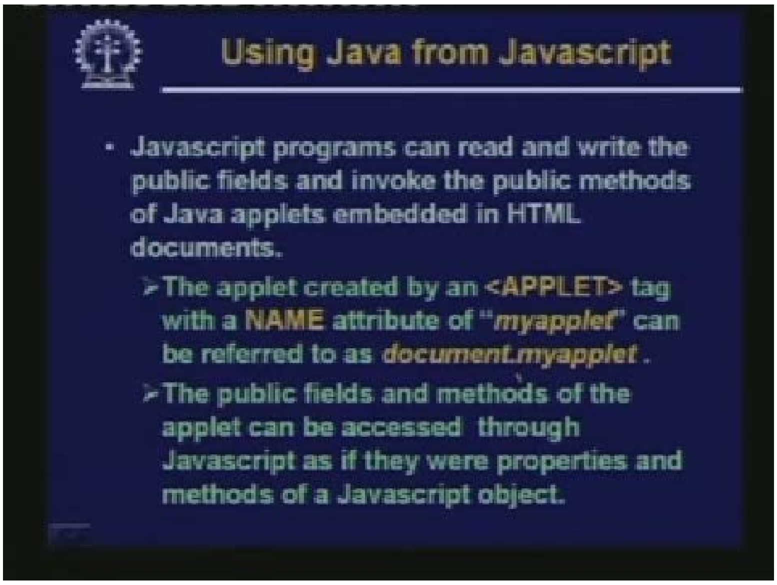
Now since from the beginning we have to skip 8 places to reach the end from where the value will start. We add 8 to this index to indicate from where the cookie will start. Actually the value of the cookie will start and to mark the end of the cookie value. We are again using an indexOf function method with a second optional parameter which tells that from which position to start the search and we are looking for this semicolon. So from this mode date portion we are trying to look for the next semicolon between which the actual value of the cookie would be present. But however if the cookie is the last cookie in the list.

This semicolon will not be there ok so the second indexOf function will be false it will be returning a value of minus 1. So if the value of end returns this minus 1 which means that this particular cookie is the last cookie. So then you assign the length of the cookie to end that is the last index and the value of the cookie is returned by using the function substring from begin to end. This entire substring will be the value of the substring and the value you can use unescape to convert all those special character escape sequences to their white spaces and punctuation symbols, you get the original value of the cookie.

Now for instance you can use this modification date cookie to check whether the value you got from the cookie is the same as the modification date of the document or not. If you find that they are not the same, then you can raise an alert that the document has been modified since the last visit. So the last time you had visited the document it was something else. So in this way you can actually utilize cookies to store some information about a document which you can check access and checks sometimes later when you visit that same page again. So cookies can be used for this kind of applications.

Now let us see Java and Javascript, how they can work together. There are two things we had talked about Javascript how Javascript code can be embedded inside an HTML file which can be downloaded and executed from the browser. Now we shall see later Java programs as applets can also be included in the web pages and downloaded compiled Java code byte code can be also interpreted and executed on the browser itself. Now some applications may demand that from Java you should be able to call some functionality of Javascript and vice versa. Let us see how this can be done. So first we will see how Java can be invoked along with Javascript.

(Refer Slide Time: 17:36)

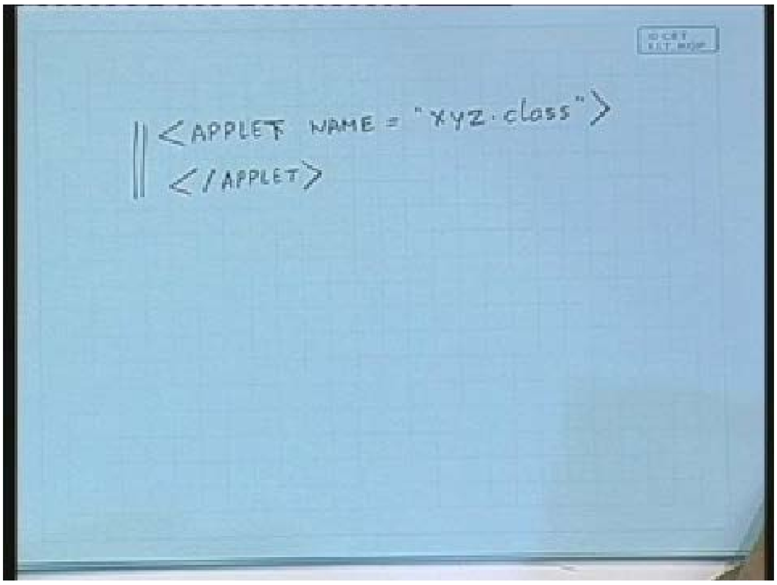


Using Java from Javascript

- Javascript programs can read and write the public fields and invoke the public methods of Java applets embedded in HTML documents.
  - ➢ The applet created by an <APPLET> tag with a NAME attribute of "*myapplet*" can be referred to as *document.myapplet* .
  - ➢ The public fields and methods of the applet can be accessed through Javascript as if they were properties and methods of a Javascript object.

So calling Java utilities means the methods and properties from Javascript. The basic concept is this. From a Javascript program we can read or write. The public fields or call the public methods; if they are not public we would not be able to this. So just to repeat again in the object-oriented paradigm inside an object there can be methods, there can some properties or variables which maybe either public which may be private. If they are private they can be accessed only from within the object. If they are public they can be accessed from outside. Now here in this context we say only the properties and methods of a Java object Java program indicate which indicates an object can be accessed which are public in nature. This is quite natural and an applet in Java is created by using an APPLET tag in HTML. The APPLET tag contains a NAME attribute which specifies the NAME of an applet.

(Refer Slide Time: 19:07)

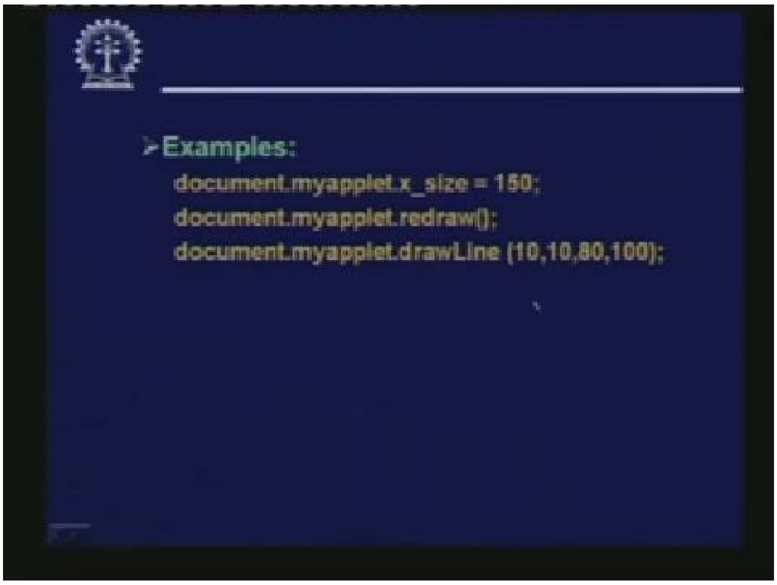


```
<APPLET NAME = "xyz.class">
</APPLET>
```

So actually when we specify a Java program inside an HTML document. We write like this APPLET NAME equal to we can give some name to the applet XYZ.class, then we can have an end APPLET. There are other attributes also. We shall them later but this is the basic thing. So we can identify an applet by a name. So there are two things of course the example I have shown just now is complete we have to specify two things. We have to specify the name of the applet also we have also to specify the file applet file which has to be downloaded. Now here I have specified the name as dot class, but actually dot class typically indicates the name of the source file. So these details we shall see later the exact syntax when we talk about Java applets how they can be included in web pages and so on.

This we shall see in our next lecture. But for the time being what is important to remember is that whenever you are including an applet in an HTML code the applet can be assigned a name using the NAME attribute of the APPLET tag. For example if I give a name of myapplet then NAME equal to myapplet then from Java script I can access this applet object as document dot myapplet as if this myapplet object appears as part of document object. Now once we access the applet like this all public fields and methods can be accessed subsequently just like as if they are properties of methods of the Javascript object. Here are some examples.

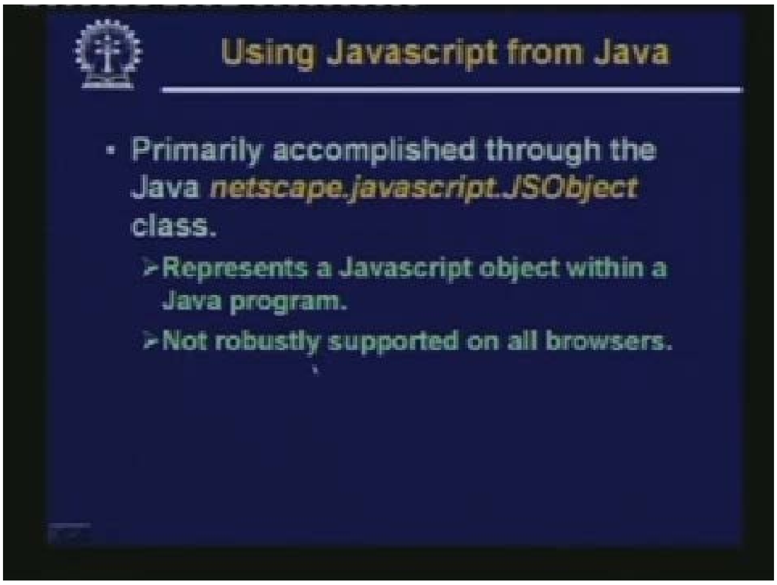
(Refer Slide Time: 21:10)



- Examples:
  document.myapplet.x_size = 150;
  document.myapplet.redraw();
  document.myapplet.drawLine (10,10,80,100);

Suppose within the applet x underscore size was a public variable, I can set that variable to 150 like this. Redraw if it a method I can invoke the method. I can invoke a method draw line with some parameter values which I can I pass like this 10, 10, 80, 100. So in this way any method any attribute which are there pertaining to the applet it can be invoked from within the Javascript code.

(Refer Slide Time: 21:51)

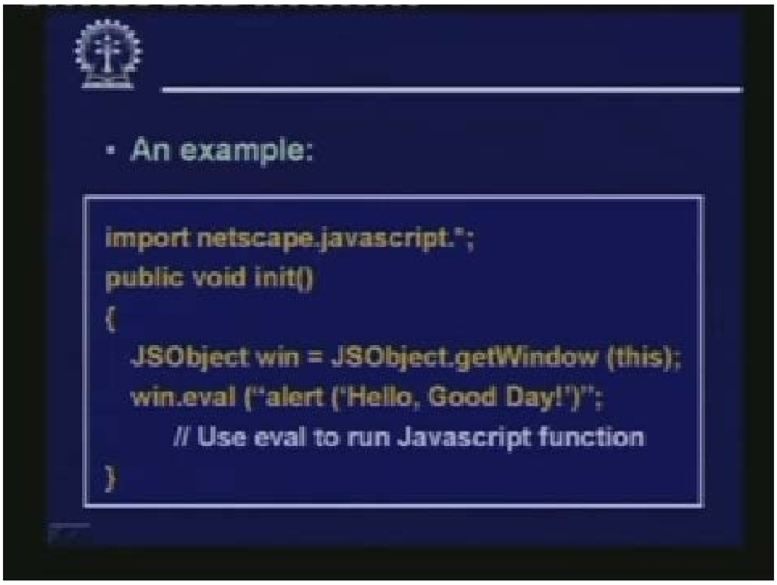


Using Javascript from Java

- Primarily accomplished through the Java *netscape.javascript.JSObject* class.
  - Represents a Javascript object within a Java program.
  - Not robustly supported on all browsers.

Now if you look at the reverse calling Javascript code from Java programs. This is not so straight forward as the other case we have just seen. This is usually accomplished through some non-standard class netscape.Javascript. JSObject is one such class which is slightly non-standard and not portable across different platforms. Now this particular class represents a Javascript object within a Java program. As I had said, this feature will not be supported on all browsers. Even if they are supported, some features maybe supported, some features maybe not. So if it is not very much recommended to use this feature when you are developing some white pages or some applications. Now a small example illustrates how we can invoke Javascript functionalities.

(Refer Slide Time: 22:52)



- An example:

```
import netscape.javascript.*;
public void init()
{
   JSObject win = JSObject.getWindow (this);
   win.eval (“alert (‘Hello, Good Day!’)”;
        // Use eval to run Javascript function
}
```

This is a small Java program snippet. Java applet in fact, here you will have to import netscape.Javascript.star to include all the relevant classes. In this function I am defining an object called win which is being obtained from the getWindow method of the current this means current. So the current window is treated as an object and is returned in win and I can invoke a method called eval on this object. Within quotes in the parameter we have actual specified a Javascript code. So when this is executed the alert statement of Javascript is executed and we get an alert box displayed on the screen.

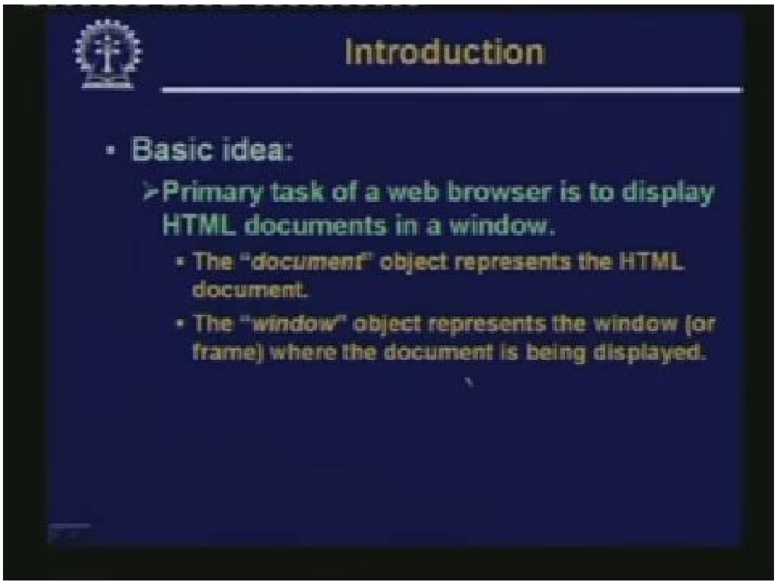
So in this is just a simple example to show you. In this way you can see how we can call some simple Javascript utilities from Java. Next we shall have a quick look at the common Javascript objects that are available for use. Now we had seen or looked at a number of Javascript examples. We had actually used a number of such objects. In those examples starting from window document history and so on. Now we shall try to systematically look at the different object types and the different attributes and methods which are available to be used along with those objects in Javascript.

(Refer Slide Time: 24:56)

“window” Object

So the first object that we will look at is the window object this the most basic kind of object.

(Refer Slide Time: 25:05)

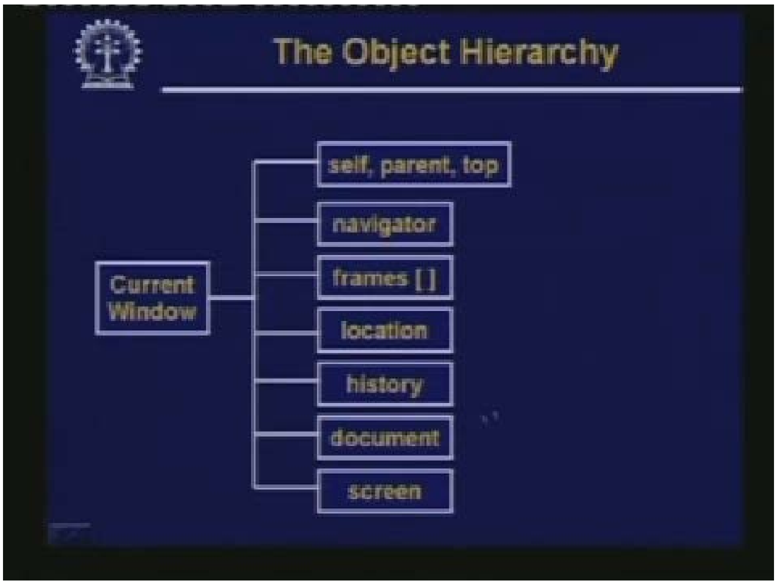


Introduction

- Basic idea:
  - Primary task of a web browser is to display HTML documents in a window.
    - The “*document*” object represents the HTML document.
    - The “*window*” object represents the window (or frame) where the document is being displayed.

And the primary task of a web browser as I have said is to display HTML documents in a window. So actually when you display an HTML document in a window, there are two things, what is being displayed and number two where it is being displayed. Now what is being displayed is the present document and where it is being displayed it can be the single window. Or if there are frames you can specify in which frame you are displaying the document. So there are two things you need to specify when you are talking about a document being displayed. Number one, what is being displayed that is the document number. Two where it is displayed that is in which window. Document and window are two related objects we require in this case. So as I have just said, the document object will represent the HTML document whereas the window object will represent the window or the frame where the document is actually being displayed.

(Refer Slide Time: 26:26)

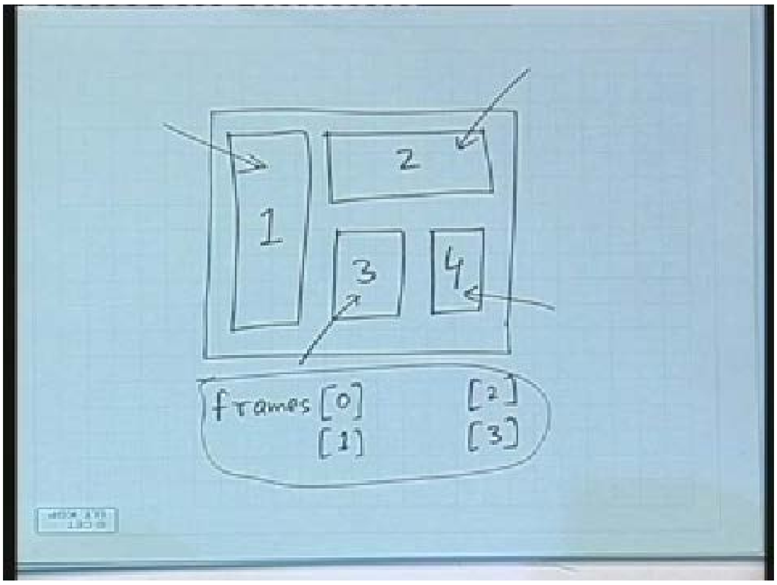


The Object Hierarchy

Now this diagram shows the object hierarchy of window. Here what we are trying to say is that from the window object what are the sub objects which are inherited from window. Which means under what are the other objects that are included by default. Now in Javascript this diagram shows almost all the objects which are included under windows. Self-parent top these are actually self-referential structures because self-parent and top are also of window type. Now within the current window if you say window.self, it means the same window. If you mention window.top which means it will represent the top level window under which the present window is embedded.

If you mention window.parent it will refer to the window from where directly this window under question is included. So the parent window, so top parent and self all there three are self-referential object. You can say they all refer to some other window object. Now in addition to this, there is the navigator object whose use usages we have seen earlier in order to get the versions of the browsers. Frames if a window has several frames, you will get an array of frame objects.

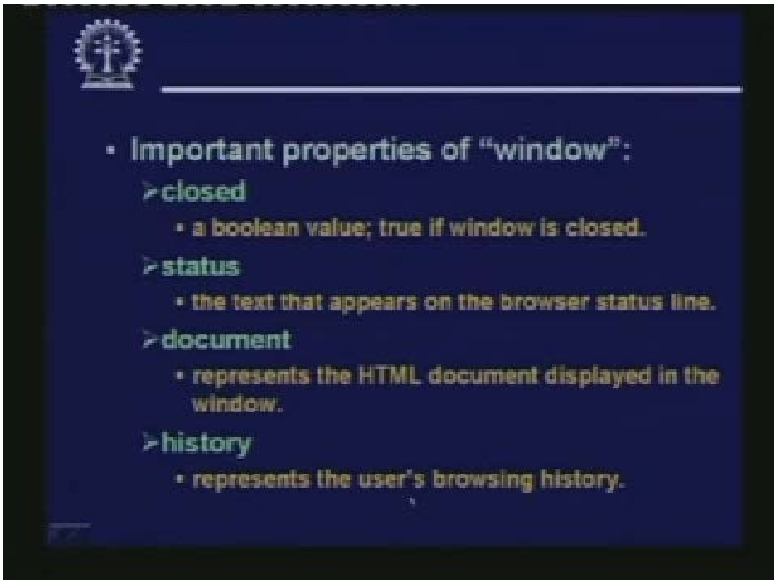
(Refer Slide Time: 28:14)



The idea is like this. Suppose you have a window being display on the screen and this window consists of several frames. This is frame 1, this is frame 2, this is the third frame, this the fourth frame. Now these frames objects sub category. This will actually refer to an array of frames which are reference by frame 0, frame 1, frame 2 and frame 3. Frame 0 will contain this particular frame. Frame1 contain this particular frame. Frame 2 will be this and frame 3 will be this. Now in the [29:14 word not clear] sense each individual arrangement of the frame array is also in window. So actually this frame array whatever we are talking about. This is actually an array of windows which are actually frames embedded inside a bigger window.

So we can have this frames array object inside the window location. We shall see later we have seen some typical usages. We can specify the URL of the document being displayed in the present window using location history. This also we have seen document. In fact document is one object which is included in a window. But under document again there are other objects we shall see these later. And in the window there is something called screen object. Again screen actually specifies some display properties like color font sizes etcetera. These are properties of screens we shall see these.

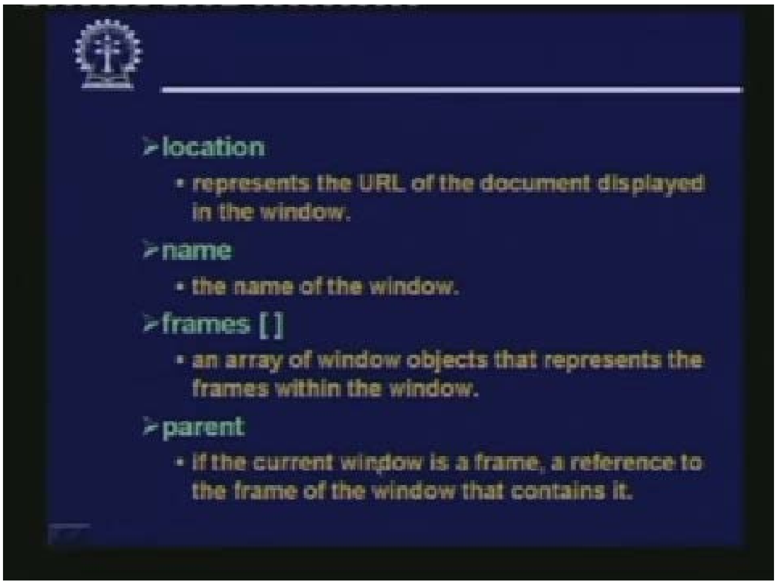
(Refer Slide Time: 30:19)



- Important properties of "window":
  - closed
    - a boolean value; true if window is closed.
  - status
    - the text that appears on the browser status line.
  - document
    - represents the HTML document displayed in the window.
  - history
    - represents the user's browsing history.

First let us look at some of the important properties of the window object. Now when you are using a window object you can use any of these properties in any suitable way you want. Closed is a property a Boolean value which indicates if a window is presently open or closed; if it is true, it means the window is closed presently. Status is a property which contains a character string. This character string is one which is displayed on the status bar of the window on top. Whatever is displayed there that is contained in the status property as a string document, you already know this represents the document that is being displayed inside the window history. This also we have seen represents the user's browsing history. So the list of the URL's has been visited by the user in the access of time.

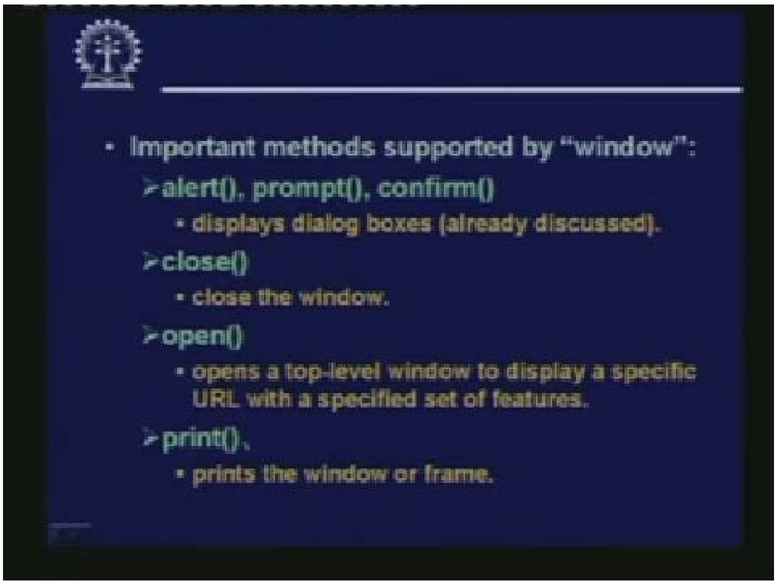
(Refer Slide Time: 31:28)



- location
  - represents the URL of the document displayed in the window.
- name
  - the name of the window.
- frames [ ]
  - an array of window objects that represents the frames within the window.
- parent
  - if the current window is a frame, a reference to the frame of the window that contains it.

Location, we have also seen this. The URL which represents the URL of the document that is presently being displayed if you change the location attribute value the document automatically gets changed. So in this way we had implemented redirection of pages also we had seen this, a window can have a name. Frames are an array of window objects. As I have said these represent the frames within the windows; parent is also a window or a frame which contains the present window. And some of the methods which you can use along with the window. Many have them we have already used them seen them in examples.

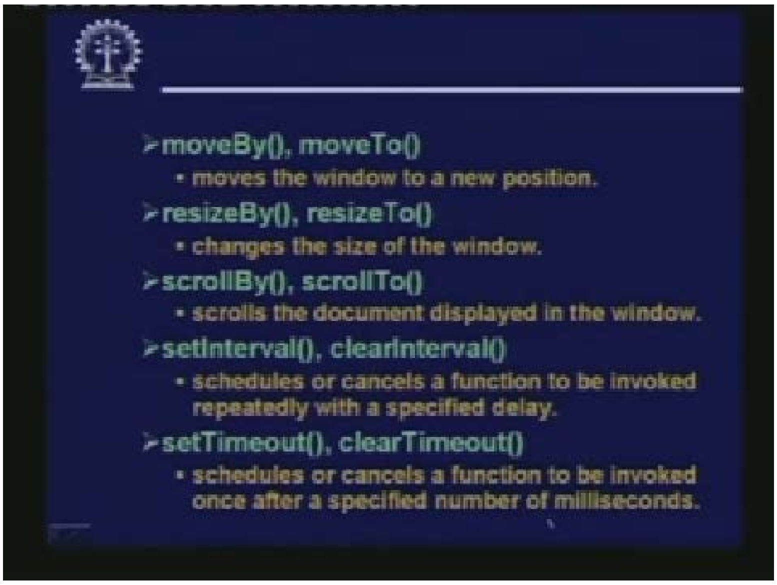
(Refer Slide Time: 32:21)



- Important methods supported by "window":
  - alert(), prompt(), confirm()
    - displays dialog boxes (already discussed).
  - close()
    - close the window.
  - open()
    - opens a top-level window to display a specific URL with a specified set of features.
  - print().
    - prints the window or frame.

Like alert prompt confirm, these are the methods which you can use only in the context of a window object. So if it is alert, so a window will be popped up with respect to the present window. So these are different ways of basically displaying dialogue boxes with user confirmation user options etcetera. Close, you can use to close the window. Open also we had seen how to use it you can. You can open a window. A new window with desired properties like you recall size whether you want a scroll bar or not, whether you want to resize or not, etcetera. So you can just open a new you can also specify the URL what is to be displayed in the window. And lastly there is a method called print. This is a useful utility which you can invoke to print the content of a window. Many page, in many pages you often find that there is an optional button that says print the contents of this window. It is actually a Javascript utility which calls this print method of the window object. If the print method is invoked the contents of the window gets printed.

(Refer Slide Time: 33:48)

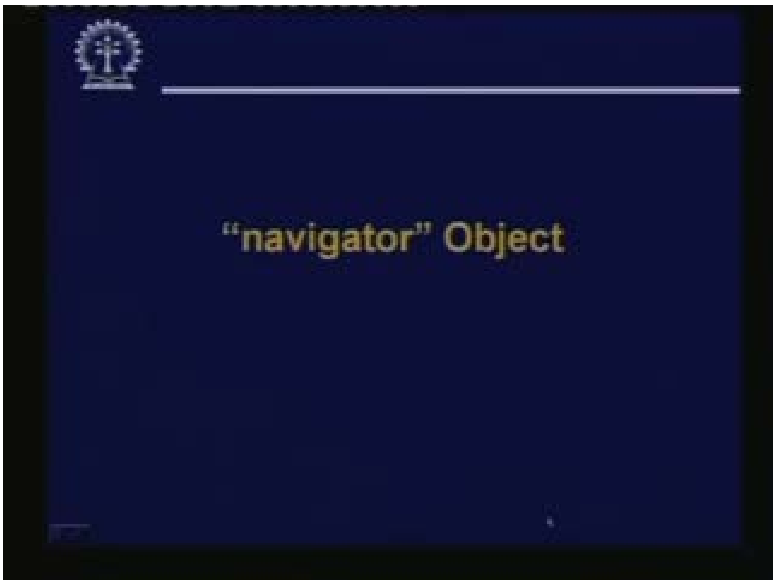


- moveBy(), moveTo()
  - moves the window to a new position.
- resizeBy(), resizeTo()
  - changes the size of the window.
- scrollBy(), scrollTo()
  - scrolls the document displayed in the window.
- setInterval(), clearInterval()
  - schedules or cancels a function to be invoked repeatedly with a specified delay.
- setTimeout(), clearTimeout()
  - schedules or cancels a function to be invoked once after a specified number of milliseconds.

There are some other functions or methods moveBy or moveTo. There are used to move a window from one location to the next. moveBy specifies the offset with respect to the present location. What is the x offset and y offset of the new position, but in moveTo we specify the absolute x y coordinate. Similarly when you want to resize the window, we can invoke these functions resizeBy. Here we specify a ratio “0.5” “0.8” “1.5” resizeTo will actually specify the x y dimension of the new window. Similarly when you want to scroll up down or right left we can specify the percentage of scroll or the actual number of lines to scroll setInterval clearInterval. These are methods which can be used to schedule or cancel. Some function to be invoked repeatedly within a specified delay.

You just remember this setInterval function or method can be used to schedule a particular method to be called repeatedly after some given delays. So whenever we need this kind of repetitive behaviour, a function will be called repeatedly after some delay you should call this setInterval function. But if you want to call a method or function only once after a specified delay you rather use setTimeout. And once you give clearInterval or cleaTimeout, this initial setting is cleared. In both cases the delay is specified in milliseconds.

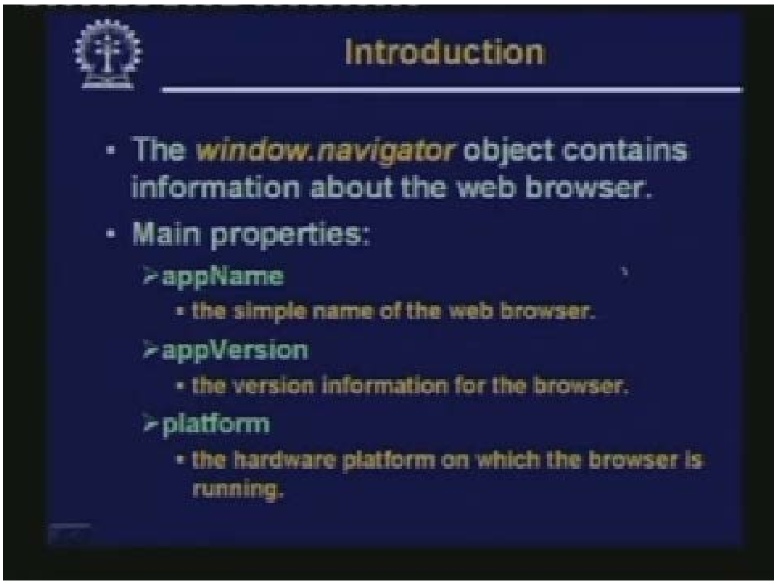
(Refer Slide Time: 35:44)



“navigator” Object

Now let us look at the navigator object. In the examples earlier we had seen that the purpose of the navigator object basically is to enable a Javascript program to gain access about some of the details about the browser.

(Refer Slide Time: 36:07)

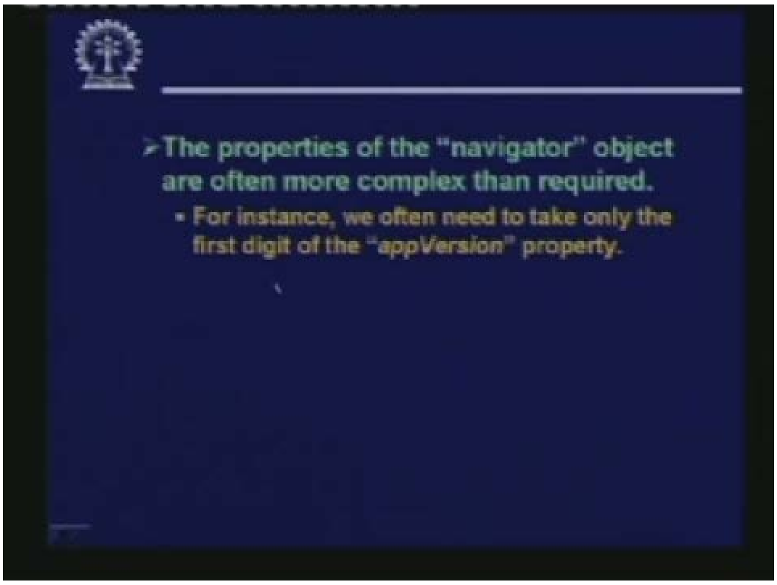


Introduction

- The *window.navigator* object contains information about the web browser.
- Main properties:
  - appName
    - the simple name of the web browser.
  - appVersion
    - the version information for the browser.
  - platform
    - the hardware platform on which the browser is running.

So here in the navigator object which I had said is an object under windows. So you access it as window dot navigator. This basically contains information about the web browser. There three main properties you sue here. Two of them you have already seen in the previous example. appName, this specifies the name of the web browser Netscape or Internet Explorer. These kinds of names you get here. appVerison, here you get the version number from the browser we had seen in that examples that in addition to this simple version number you get a lot of other information like your hard drive platform etcetera and if you simply call the platform attribute you invoke these. If you access this you will get the name of the hard drive platform on which the browser is running. That means whether it is an Intel machine. It is a Pentium machine or it is a SUN's Solaris platform or HP UX [37:14 word not clear]. You will get the details of the platform on which your browser is running. So all these properties you can use with the windows navigator object to get some information about the browser.

(Refer Slide Time: 37:31)



- The properties of the "navigator" object are often more complex than required.
  - For instance, we often need to take only the first digit of the "appVersion" property.

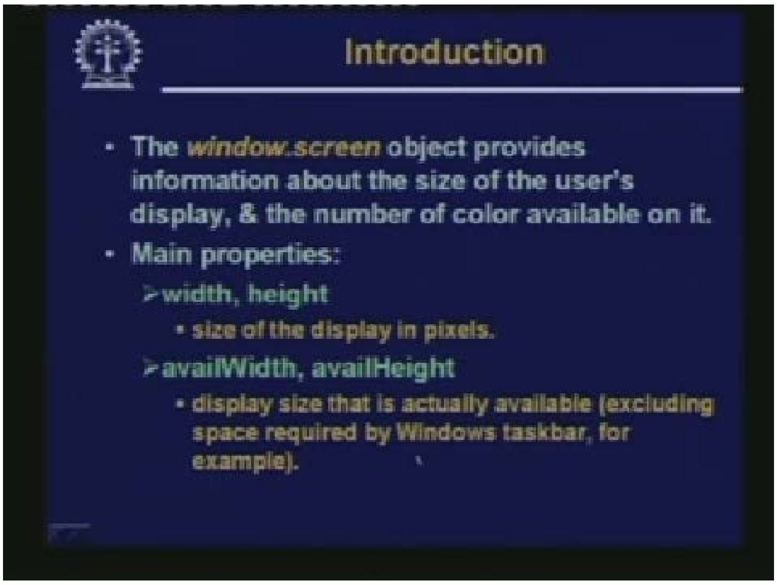
And as I have said, some of the properties of the navigator object are more complex than what is required. For example I have seen that in the applet, in the application, appVerison property you get a lot more information than required. Typically we need only the first digit to get the version number version number 3, 4 or 5 of the particular browser you are using, that is often considered to be enough. But this string actually what it returns is a very long string. So you can use the substring function to extract the first digit of this string and use it for whatever you want.

(Refer Slide Time: 38:15)

"screen" Object

Next let us come to the screen object screen. As I had mentioned this refers to the area where the document is displayed under window.

(Refer Slide Time: 38:29)

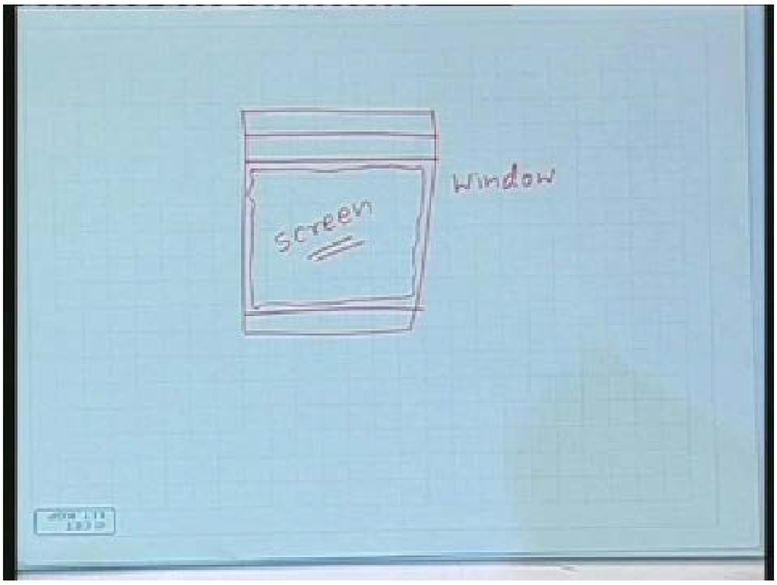


Introduction

- The *window.screen* object provides information about the size of the user's display, & the number of color available on it.
- Main properties:
  - width, height
    - size of the display in pixels.
  - availWidth, availHeight
    - display size that is actually available (excluding space required by Windows taskbar, for example).

So you just try to understand the difference between a window and the screen.

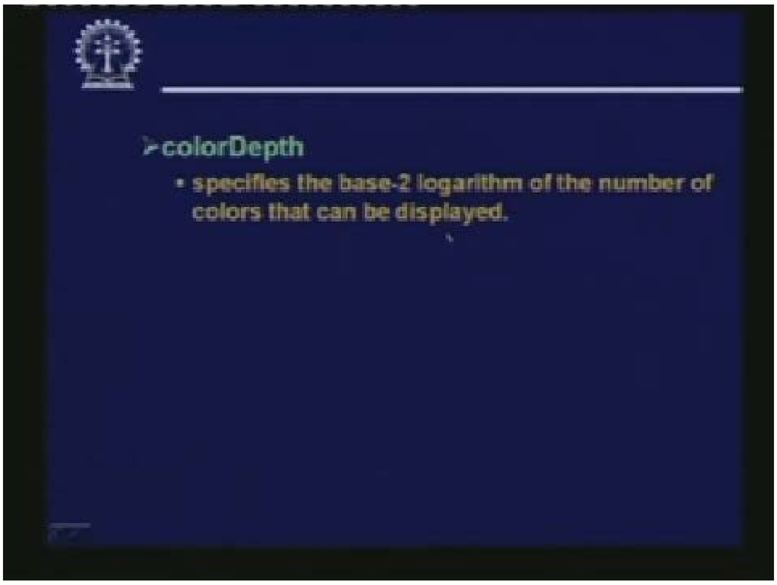
(Refer Slide Time: 38:36)



You see, this is a window object. In the window object there are so many things. There can be a title bar on the top. There can some menus on the top; there can be a status bar on the bottom. So effectively this will be the area where your document will be displayed. So this area can be considered as your screen. So let us see what are the properties of the screen attribute or object you need to specify. So the windows screen object will provide you information about the size of the display and also some of its capabilities like the number of colors you can use on it. Main properties are width and height which will specify the size of the display.

If the unit of width and height are in pixels, it also availWidth, availHeight which are the available width and height where the area that is occupied by menus taskbar etcetera they are excluded from the size. That means from the total width and height you take out. The areas or the portions which are taken up by the default window taskbars the remaining portion. That is your available width and available height. So in an application you can have access to your available width and available height. In order to know how you can best display or you can configure the display in a suitable way depending on the available space.

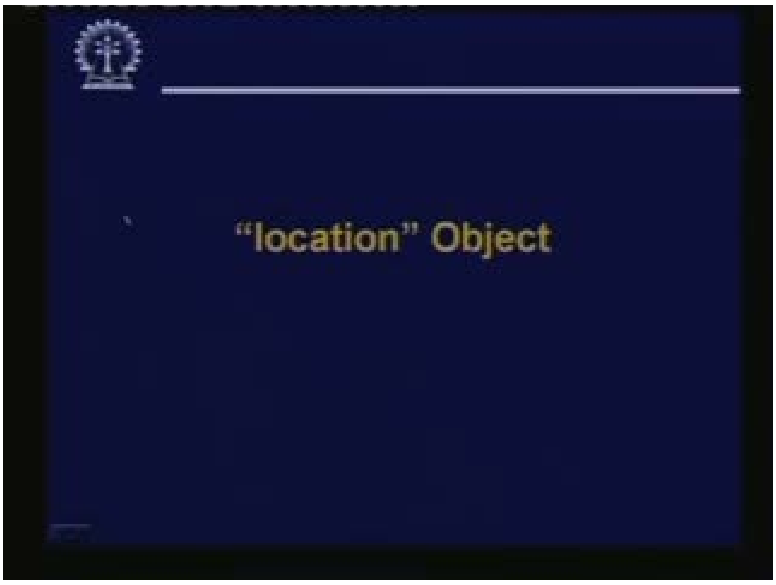
(Refer Slide Time: 40:31)



➢colorDepth
- specifies the base-2 logarithm of the number of colors that can be displayed.

And the last attribute is something called colorDepth. Now in colorDepth here basically this is the way of specifying that how many colors you can support, it typically specifies the base two logarithm of the number of colors that can be displayed which means if your display supports 256 colors the value will be 8 log 256 to the base 2. But in general the value can be less than the log 2 of the maximum number of colors because some colors may not be actually usable. Depending on the pallets you have been using. But in general you can use it to actually indicate how many colors are available to you.

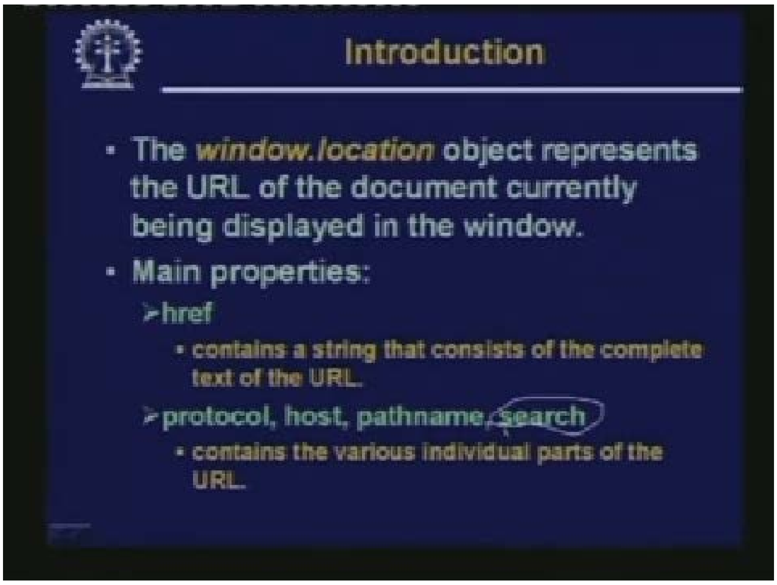
(Refer Slide Time: 41:23)



"location" Object

Next let us come to the location object.

(Refer Slide Time: 41:32)

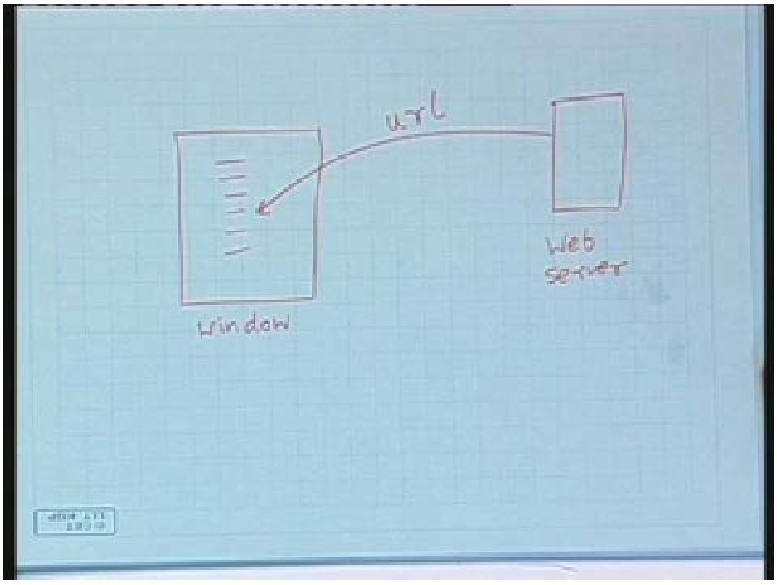


Introduction

- The *window.location* object represents the URL of the document currently being displayed in the window.
- Main properties:
  - href
    - contains a string that consists of the complete text of the URL.
  - protocol, host, pathname, search
    - contains the various individual parts of the URL.

Now as I had said the location object actually is something using which you can specify the URL of the document which is displayed in the window. So here the idea is again like this.

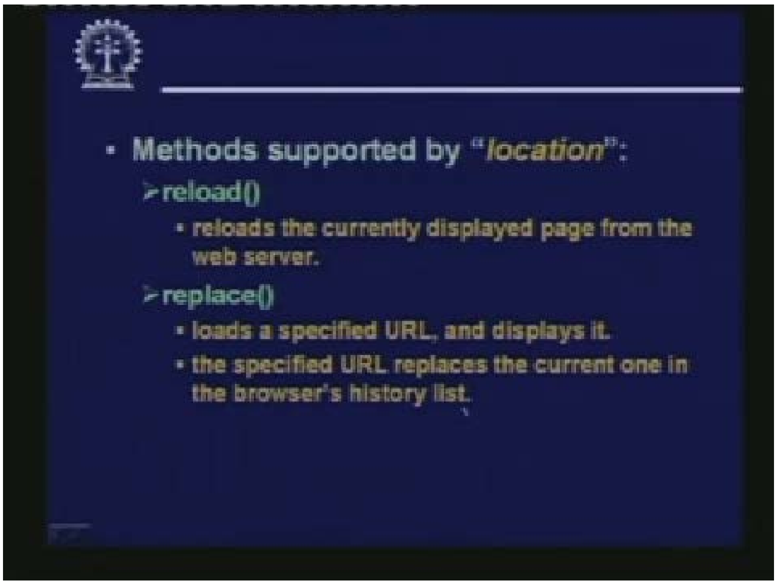
(Refer Slide Time: 41:44)



You have the window. This is your window and whatever is being displayed in the window this has come from somewhere else. This is a web server. So the location object actually talks about the URL from where this document has come. This is the significance of the location object associated with a document or a window. So the window.location will represent the URL of the document which you can see presently displayed in the window. There are several properties here again. The first property which includes a number of things is the href. Href is actually a string which consists of the complete URL. Like href can contain within quotes http://www.iitkgp.ernet.in. This kind of a complete URL you can specify and this value will be available against the href attribute of location object.

But the different components of the URL you can also access separately by the attributes protocol host pathname search. Protocol in the precious example will be http post will be www.iitkgp.ernet.in pathname. If within the host you have specified some subdirectories slash docx slash index or HTML that portion will be in the search. Path pathname and lastly there is a search option you can specify here. This is if you are using the GET method of submitting the URL, you can give a question mark followed by some keywords or some data which will be passed to the web server. So this search attribute will be containing the data that will be following the question mark.

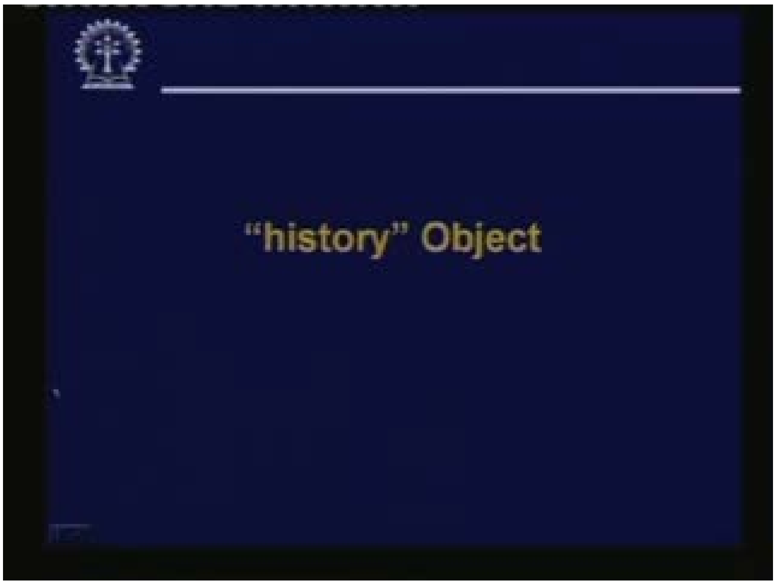
(Refer Slide Time: 44:03)



- Methods supported by "*location*":
  - reload()
    - reloads the currently displayed page from the web server.
  - replace()
    - loads a specified URL, and displays it.
    - the specified URL replaces the current one in the browser's history list.

And in the location object there are two methods which are supported. One is reload; reload see, with location you have the URL. If you invoke reload the page will be again be loaded from the web server. That is you want to refresh a page sometime if you invoke reload the page will be reloaded from the web server. Now you know that in the browser you have a default reload button this functionality you can also have directly from a Javascript method which can be invoked from the Javascript or you can have replace. Replace is not reload; rather here we specify some other URL. So some other URL will be loaded and will be replacing the presently displayed page on the window. Now if you use replace, then this specified URL will also replace the current one in the browser history list. It is not that the new page will be considered as a page which was visited after I had visited the previous page no. Within the history I am replacing the history information by this new information replace will do exactly that.

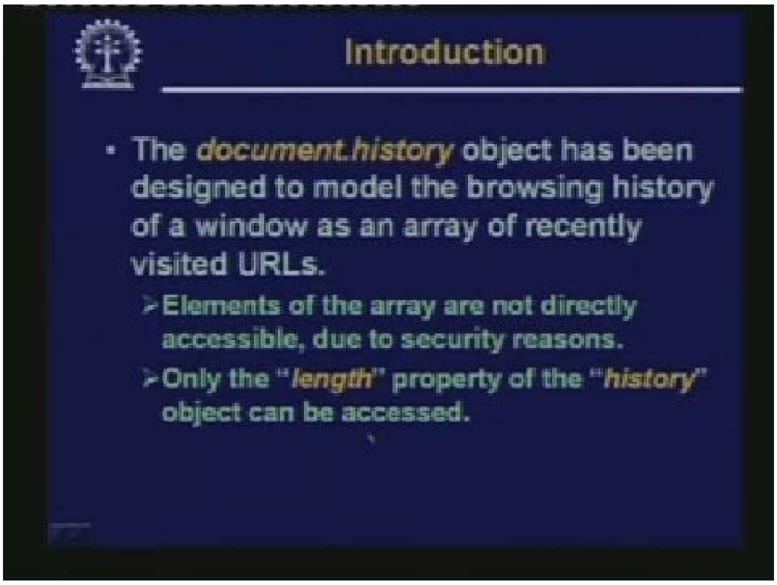
(Refer Slide Time: 45:25)



"history" Object

Now the history object we had already seen earlier.

(Refer Slide Time: 45:29)

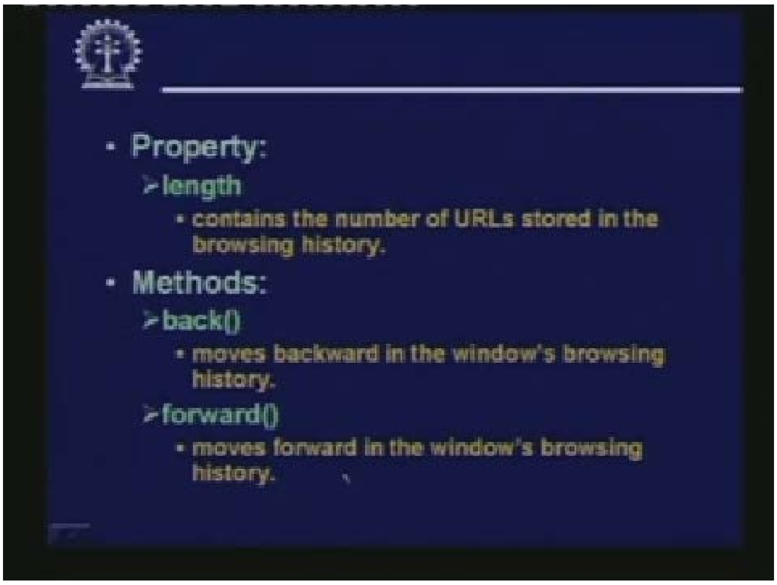


Introduction

- The *document.history* object has been designed to model the browsing history of a window as an array of recently visited URLs.
  - Elements of the array are not directly accessible, due to security reasons.
  - Only the "*length*" property of the "*history*" object can be accessed.

History object is basically used to model the browsing history of the window. It actually represents a string which is considered to be an array of recently visited URLs. Now from security point of view the elements of the array or the individual URLs are not directly accessible from Javascript. Because if you are allowed to do, so then a Javascript program as part of a downloaded HTML document can have access to the pages that a user is browsing which can invade on the secrecy of the particular user. So normally you are not allowed to look at the individual URL's in the history. But rather what you can see is the length how many URL's are contained in the history.

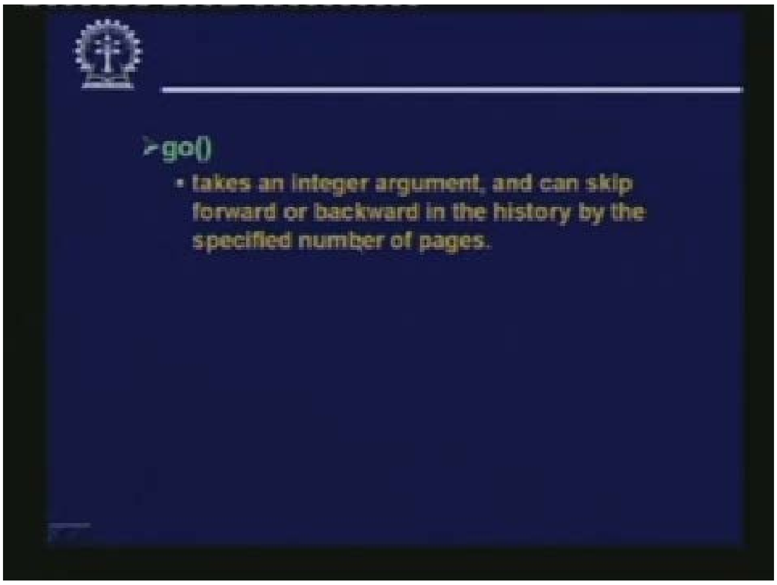
(Refer Slide Time: 46:32)



- Property:
  - length
    - contains the number of URLs stored in the browsing history.
- Methods:
  - back()
    - moves backward in the window's browsing history.
  - forward()
    - moves forward in the window's browsing history.

And some of the properties there is you can have access to the length. As I have said this will contain the number of URL's stored in history. Regarding methods the back and forward method, you have already seen through examples. You can move forward, you can move backward. See you can move forward and backward in history without accessing the URL's directly you can display the next page or the previous page on the window. That is what you are allowed to do and instead of moving back and forward by a single page in terms of history at a time.

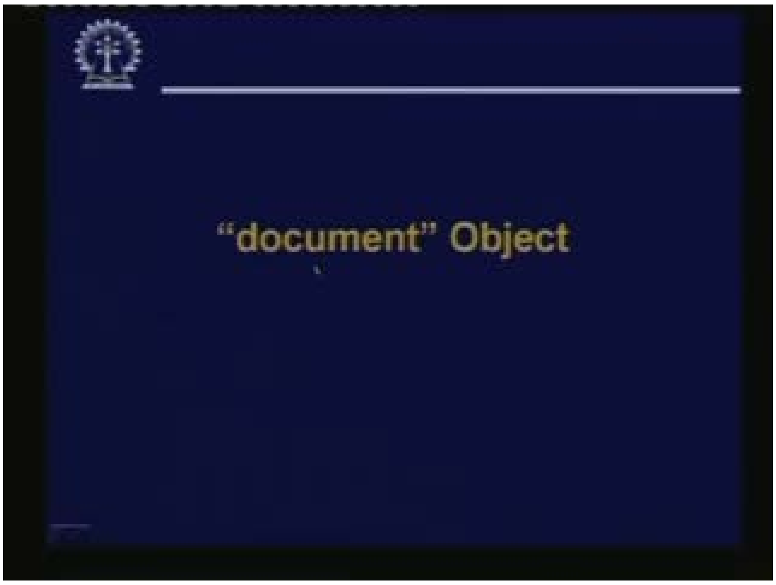
(Refer Slide Time: 47:11)



- go()
  - takes an integer argument, and can skip forward or backward in the history by the specified number of pages.

You can also you the go method which takes as argument an integer which specifies the number of pages to skip in the history list. Like if you give go within bracket 5, it will be the 5th page. In the future if you give go minus 3, the third page in the past. So in this way you can skip arbitrary number of pages to go forward and backward in the history list.

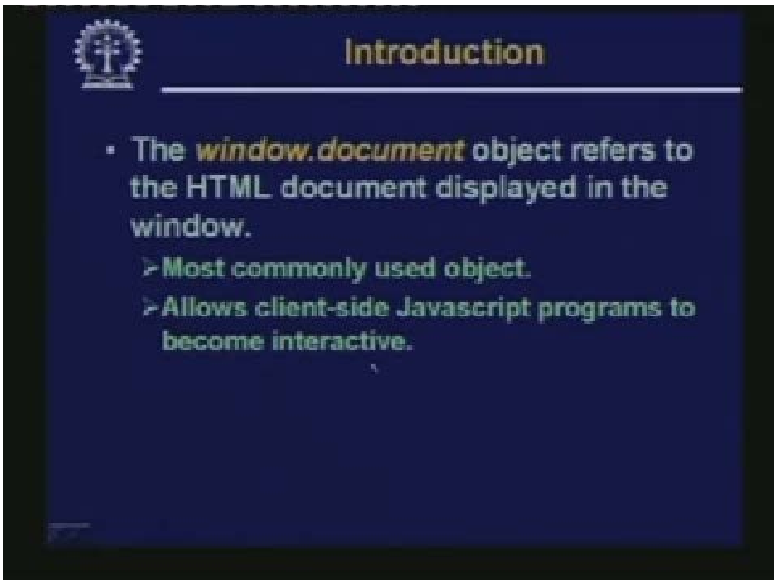
(Refer Slide Time: 47:43)



“document” Object

Next let us come to the document object. Now the document object is also a very important object. Just like window because as I had said the document object indicates the document being displayed on the present window or the frame and there are number of attributes and methods you need to associate with this particular object in order to operate or manipulate the document. Let us see.

(Refer Slide Time: 48:14)

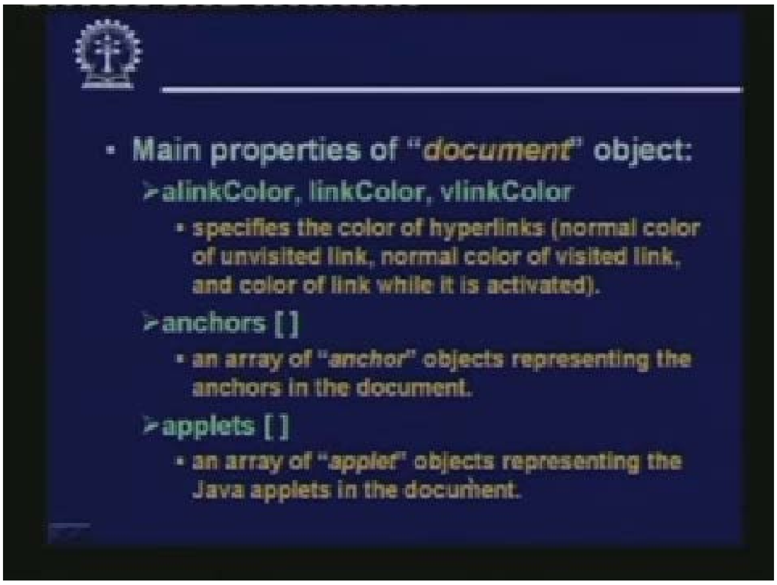


Introduction

- The *window.document* object refers to the HTML document displayed in the window.
  - Most commonly used object.
  - Allows client-side Javascript programs to become interactive.

So this is what I have said window document is displayed, the HTML document being displayed. This is most commonly used along with window and it is due this window document object and the method and attributes available under it. This allows the client-server Javascripts help in designing interactive web pages. Since you are able to manipulate the document it helps you in making the pages interactive in number of different ways.

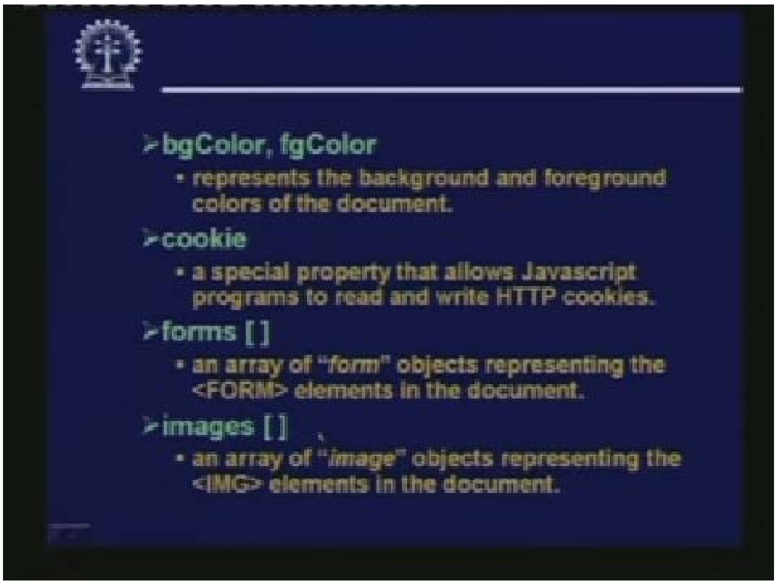
(Refer Slide Time: 48:53)



- Main properties of "*document*" object:
  - alinkColor, linkColor, vlinkColor
    - specifies the color of hyperlinks (normal color of unvisited link, normal color of visited link, and color of link while it is activated).
  - anchors [ ]
    - an array of "*anchor*" objects representing the anchors in the document.
  - applets [ ]
    - an array of "*applet*" objects representing the Java applets in the document.

Let us see some of the properties and methods. Some of the ‘main properties of the document object are alinkColor, linkColor, vlinkColor. These are the colors of the hyperlinks. The first one indicates the normal color of an unvisited link which has not yet been clicked. linkColor represents the color of a link after it has been clicked and visited and vlinkColor while you are pressing or clicking a link, the color of link temporarily changes to some other color that is vlinkColor. Second set of properties are anchors. Now you know within a document you can provide some anchors and you can have hyperlinks made to those anchors in the same document. By clicking on the hyperlinks you can straight away to a certain section of a document. Now a document can have ten anchors defined, they will be available as an array of ten elements and this array will be called anchors. So anchors actually represent an array of anchor objects which are character strings representing the defined anchors in the document. Similarly if you have applets defined in your HTML page then the applets property will contain the array of all the applet objects that have been defined.

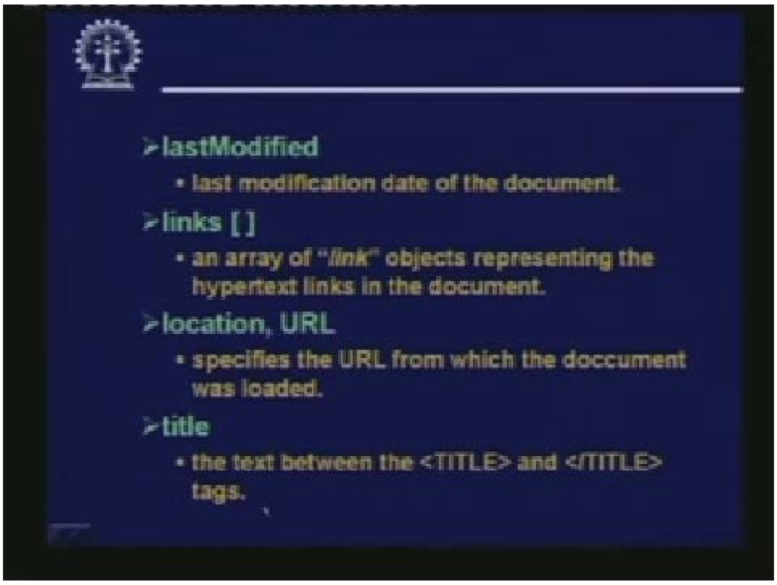
(Refer Slide Time: 50:33)



- bgColor, fgColor
  - represents the background and foreground colors of the document.
- cookie
  - a special property that allows Javascript programs to read and write HTTP cookies.
- forms [ ]
  - an array of "*form*" objects representing the <FORM> elements in the document.
- images [ ]
  - an array of "*image*" objects representing the <IMG> elements in the document.

bgColor, fgColor represents the background and foreground color which you can change as per your need. Cookie we have already seen how we can use it. Forms again if there are forms in the document they can be accessed as an array of form objects. Similar is the case for images if there are several IMG tagged elements representing images in the document. Each of them will indicate a particular element of this image array. So all these things are available for access in a document.images object.

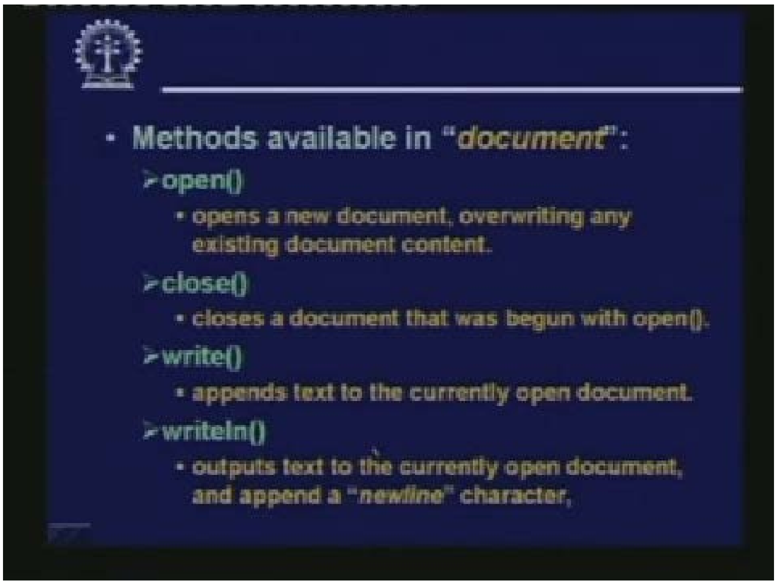
(Refer Slide Time: 51:19)



- lastModified
  - last modification date of the document.
- links [ ]
  - an array of "*link*" objects representing the hypertext links in the document.
- location, URL
  - specifies the URL from which the doccument was loaded.
- title
  - the text between the <TITLE> and </TITLE> tags.

lastModified, we had seen this example. This represents the last modification data of the document link. If there are hyperlinks in the page again they are available as array of links location or URL. Both you can use they represent the URL from where the document presently being displayed had been loaded. Title is an attribute which the text which stored between the begin TITLE and end TITLE tags of the document being displayed.

(Refer Slide Time: 52:05)



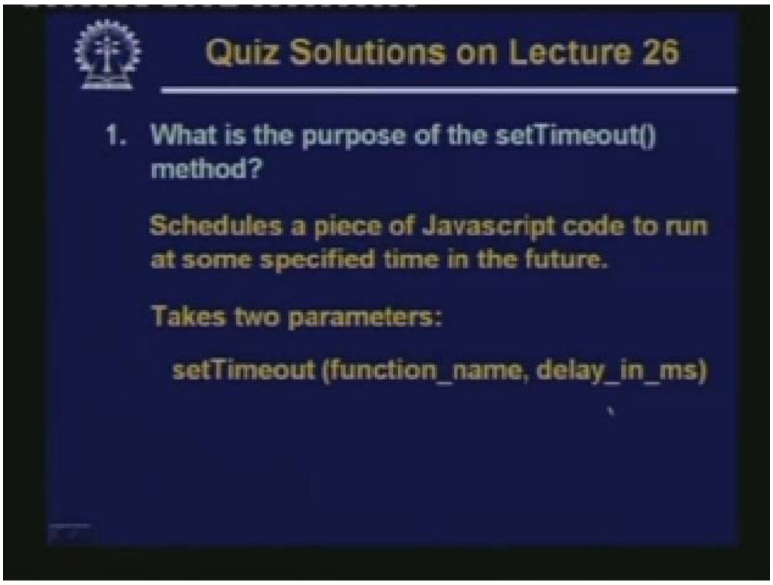
- Methods available in "*document*":
  - open()
    - opens a new document, overwriting any existing document content.
  - close()
    - closes a document that was begun with open().
  - write()
    - appends text to the currently open document.
  - writeln()
    - outputs text to the currently open document, and append a "*newline*" character,

Now some of methods which are available on the document are, see document means when you have a document being displayed on a screen, we have said earlier that this document is something which is embedded inside a window document in basically an HTML file. So when we talk about the methods applicable under this we should be remembering that actually what is being contained in this object is nothing but HTML document. So the first method available is open which opens a new HTML document inside the present window.

This open is different from an open window. In window open we are opening a new window. But here the window is the same we are opening a new HTML document under that. Similarly you can close a document, you can call write or writeln; write means you just write the string specified in the current location of the document.

Writeln means after writing the string you append a newline character automatically. So in this by using write or writeln; whatever you are outputting that gets appended or merged to the surrounding HTML document. Now with this we come to the end of our discussion on Javascript. If you recall again over the last three lectures we had talked about the basic features of the Java language including the object-oriented features, object-oriented paradigm, concepts of object and properties. Then through examples we had tried to see how we can use these properties to our benefit. How we can utilize these concepts to develop more attractive and interactive web pages. Now let us see some solutions to the problems of our previous lecture. So in our lecture number 26, the first question was:

(Refer Slide Time: 54:31)



Quiz Solutions on Lecture 26

1. What is the purpose of the setTimeout() method?

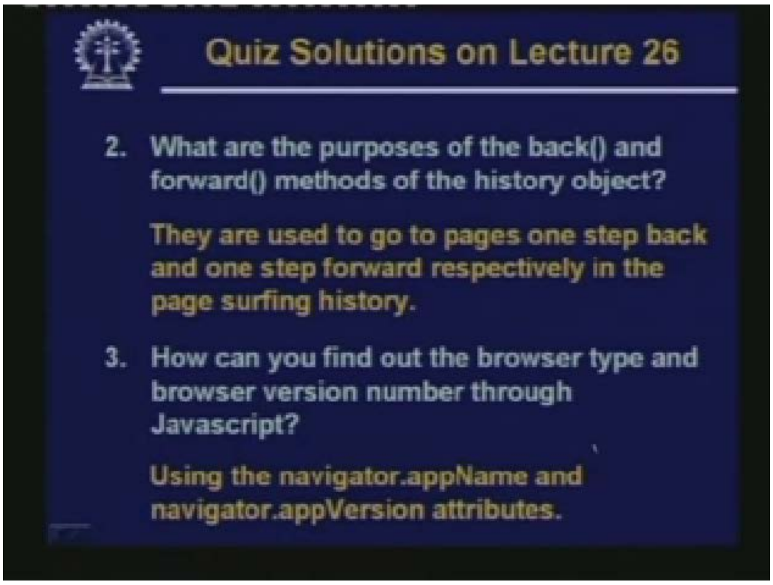
   Schedules a piece of Javascript code to run at some specified time in the future.

   Takes two parameters:

   setTimeout (function_name, delay_in_ms)

What is the purpose of the setTimeout method?
Here the setTimeout I had said the setTimeout method basically is used to schedule a piece of Javascript code to execute and run at some specified time in the future. Now here you must specify two parameters to this function or method. The first one is which Javascript method to invoke and second parameter will indicate how much after how much delay it is to be invoked and this delay has to be specified in milliseconds.

(Refer Slide Time: 55:15)



Quiz Solutions on Lecture 26

2. What are the purposes of the back() and forward() methods of the history object?

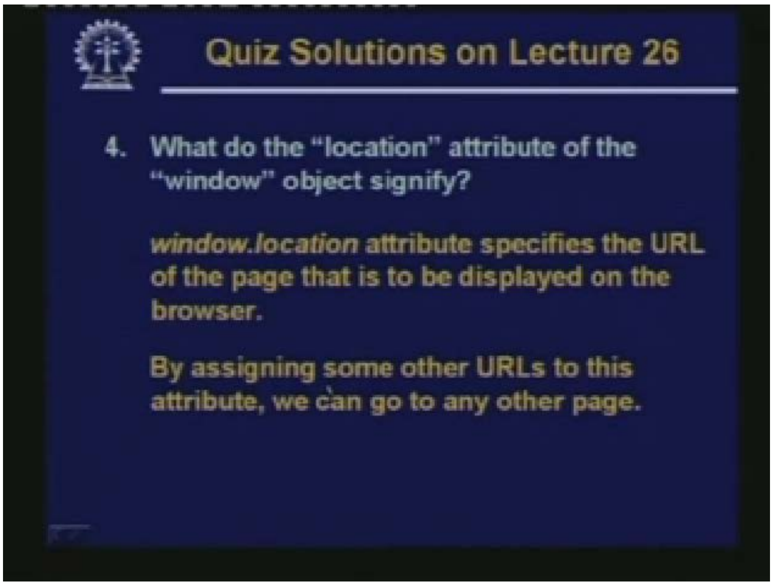
   They are used to go to pages one step back and one step forward respectively in the page surfing history.

3. How can you find out the browser type and browser version number through Javascript?

   Using the navigator.appName and navigator.appVersion attributes.

Second. What are purposes of the back and forward methods of the history object?
This we have explained several number of times. There are basically used to move backward and forward in the history list. This allows you to browser across the pages you have visited in the recent past or in the future we again come back from there. So you can move forward and backward in that list.
How can you find out the browser type and browser version number?
This again we had mentioned using the navigator appName and navigator appVerison attributes, we can use these.

(Refer Slide Time: 56:00)



Quiz Solutions on Lecture 26

4. What do the "location" attribute of the "window" object signify?

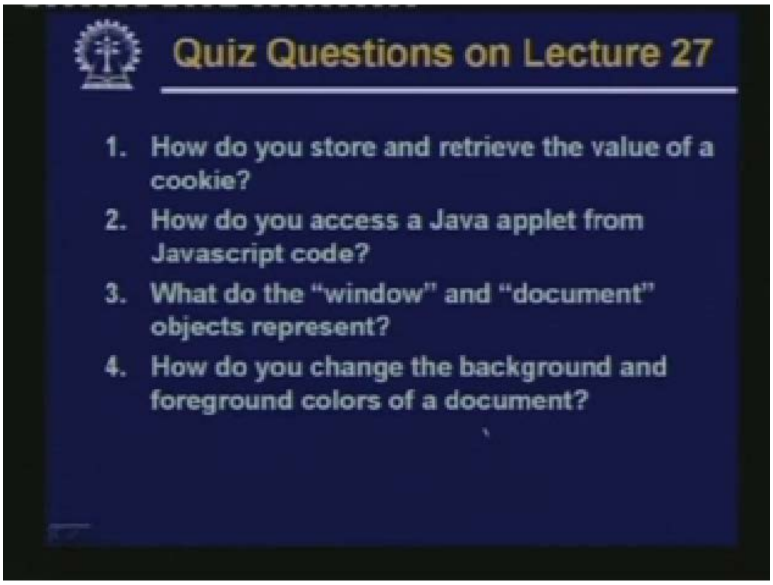
   window.location attribute specifies the URL of the page that is to be displayed on the browser.

   By assigning some other URLs to this attribute, we can go to any other page.

What do the location attribute of the window object signify?
Location actually represents the URL of the page that is presented or displayed on the browser. As we had seen through examples, if we assign some new URL to this location

attribute, the page being displayed also gets changed. So now some questions on today's lecture.

(Refer Slide Time: 56:35)



Quiz Questions on Lecture 27

1. How do you store and retrieve the value of a cookie?
2. How do you access a Java applet from Javascript code?
3. What do the "window" and "document" objects represent?
4. How do you change the background and foreground colors of a document?

How do you store and retrieve the value of a cookie?
This we have seen through examples today.
How do you access a Java applet from Javascript code?
Here we had seen that the first part is easier the reverse is not so easy and is a non-standard thing. So Java applet from Javascript code is straight forward.
What do the window and document objects represent is Javascript?
How do you change the background and foreground colors of a document?
Now so with this we end today's lecture. In the next lecture we shall be starting our discussion on Java and Java applets. Thank you.