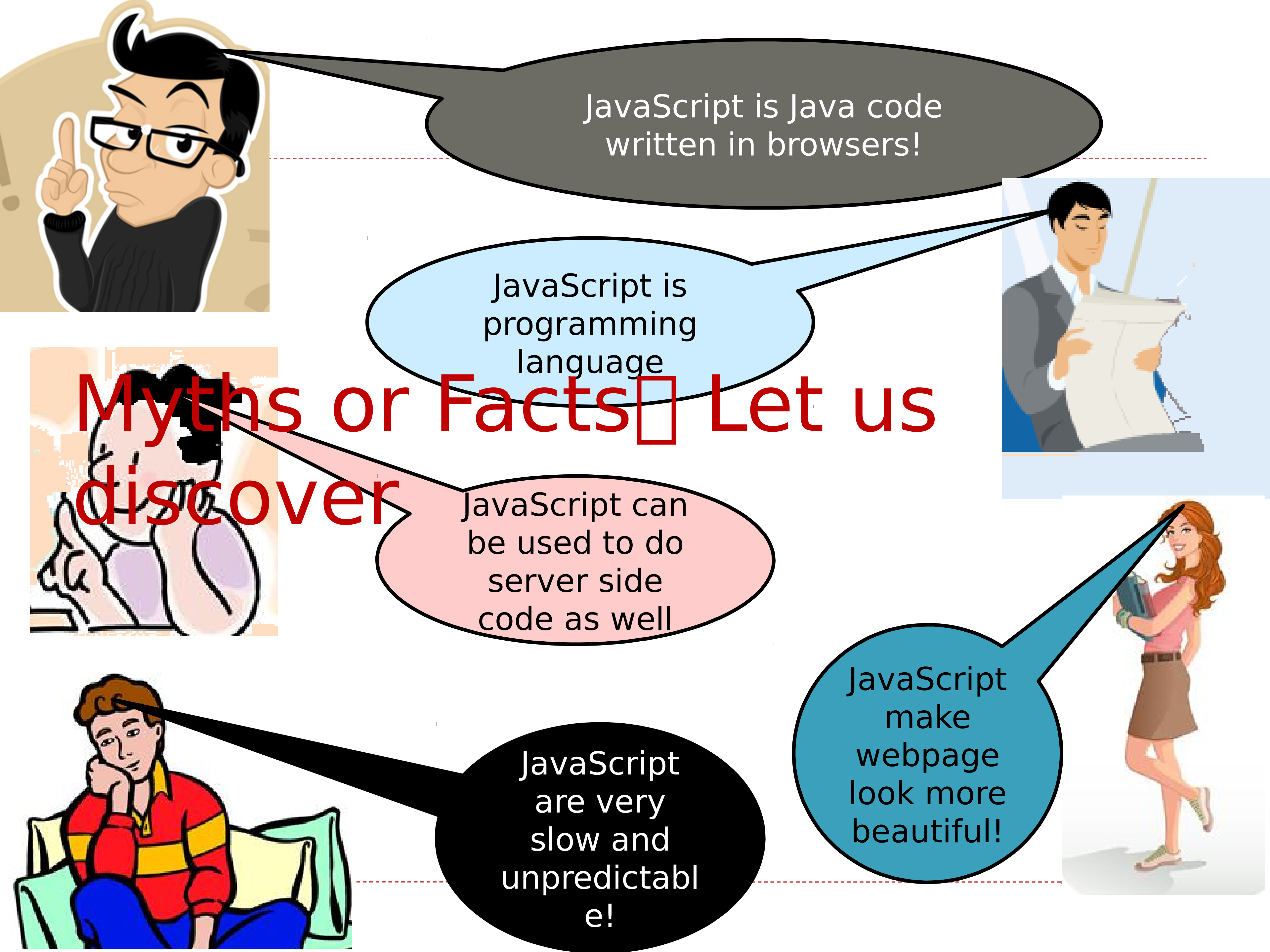
Classical JavaScript

Rajeev Gupta
Trainer & Consultant

JavaScript language fundamentals

Myths or Facts Let us discover

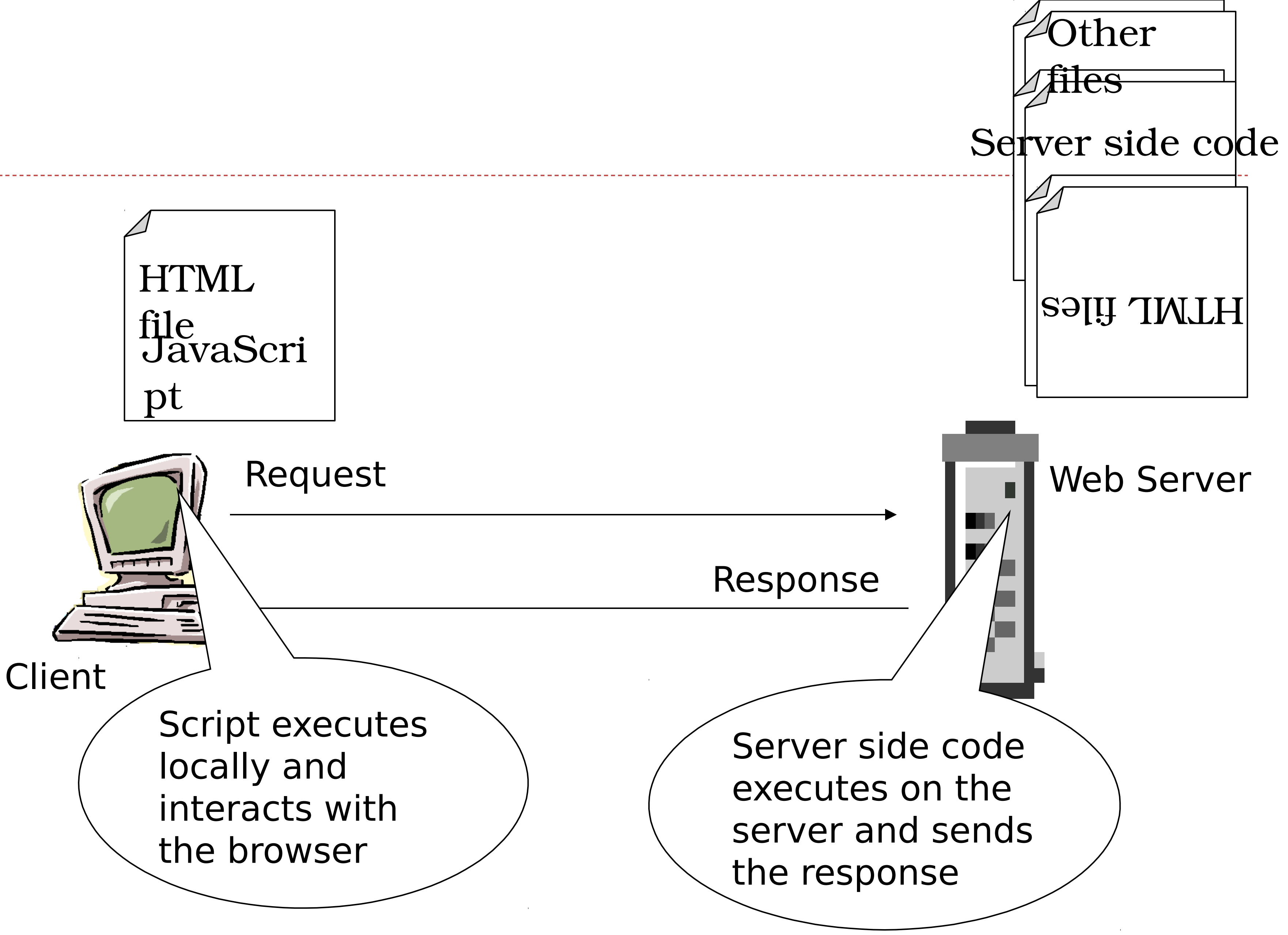

JavaScript is Java code written in browsers!
JavaScript is programming language
JavaScript can be used to do server side code as well
JavaScript make webpage look more beautiful!
JavaScript are very slow and unpredictabl e!

JavaScript

- Is a scripting language developed by Netscape, and later standardized by W3C.
- It was originally called LiveScript.
- JavaScript on web page works with HTML and CSS to create a DHTML page.
- **Script are line of code that does not execute stand-alone. They run on browser (client-side) or application server (server-side) or on top of some other application.**
- They are interpreted at runtime and are not compiled.

JavaScript in a Web Page

- To create more interactive pages- client side validations etc.
- To generate html dynamically.
- Event handling
- To enhance browser capabilities by giving it a better look – printing on status bar etc.
- Interaction with embedded components like applets and active x controls.


Other files
Server side code
HTML file
JavaScript
HTML files
Request
Web Server
Response
Client
Script executes locally and interacts with the browser
Server side code executes on the server and sends the response

- So JavaScripts on client side executes faster than server-side code.

Language Features

- Syntax similar to C++ and Java
- Case sensitive
- Loosely typed
- Interpreted
- Platform independent
- Object-based language
- Semicolon, as separator for multiple statements in the same line.

Object-based language

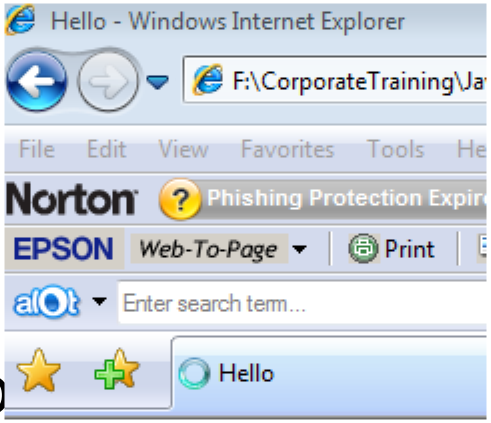
- Also called prototype-based object oriented language
- Object-based language
  - JavaScript objects are associative arrays
  - functions as object constructors and methods
  - prototypes instead of classes for inheritance.

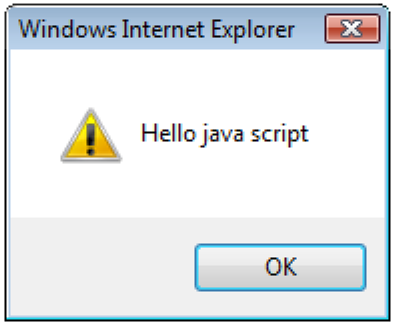
More on this later

Simple Scripts in HTML

```
<HTML><HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
First java script code<br>
<SCRIPT type="text/javascript” >
//Java script single line comment
alert(“Hello java script”);
/* java script script
multi-line comment */
</SCRIPT>
</BODY>
</HTML>
```

pop


Hello - Windows Internet Explorer
F:\CorporateTraining\Ja
File Edit View Favorites Tools He
Norton
Phishing Protection Expir
EPSON Web-To-Page Print
alOt
Enter search term...
Hello



Hello java script

External Script

- Scripts can also be written in a separate file and can be referenced in a HTML file.

```
<HTML><HEAD><BODY>
<SCRIPT type="text/javascript”
SRC=“jsfile.js”>
</SCRIPT>
</BODY>
</HTML>
```

jsfile.js

```
alert(“Hello”);
```

Path can be

1. Absolute path
   `http:://server1/get.jsp`
2. Root relative:
   `/scripts/jsfile.js`
3. Document relative
   `../scripts/jsfile.js`

Why would you want to write JavaScript in another file?

Placeholders for scripts within HTML

- Inside head
  - only declarations. Declarations could for variables or functions.
  - No standalone executable statements must appear
- Inside the body
  - any statements can appear
  - standalone executable statements inside the body are interpreted in place where it appears.
- Along with the event handler
  - Script expression can be written as a value when an event like button click happens.

Primitive data type supported

- Six primitive data type supported :
  - String
  - number
  - boolean
  - null
  - undefined

Variables and Data types

- Variable names must begin with a letter, under-score or $, subsequent characters can be a letter, under-score or $ or number.
- They can be assigned with proper value and used where ever appropriate . They are called data stores.
- To declare a variable:
  - `var x;`
  - `var x=1;` declare and initialize
- Variables declaration is not compulsory in JavaScript. (If you don't declare a variable explicitly, JavaScript will declare it implicitly for you.)

Example

```
<HTML><HEAD>
<SCRIPT type="text/javascript”>
$x=false;
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT type="text/javascript”>
alert($x==0);
</SCRIPT>
</BODY>
</HTML>
```

What do you think the alert box will display?

Find what alert box will display if $x is uninitialized?

Operators

- Arithmetic:
  `+   -   *   /   %   +=   -=   *=   /=   %=  ++  --`
- Logical:
  `&  |   !  &&   ||`
- Relational:
  `>  >=  <  <=  ==  != === !==`
- String concatenation: `+`
- Bit wise:
  `>>  <<  >>>  >>=  <<=  >>>=`
- Ternary: `?:`
- Conversion functions:
  `parseInt()` and **`parseFloat()`**
  To convert string to **`int`** and **`float`** respectively

=== and !==

- **`alert("34"==34);`**
  - Returns true
- **`alert("34"===34);`**
  - Returns false
- **`=== and !==`** are used for stricter comparisons based on types.
- Note that **`alert(34.0===34)`** returns true
- **`alert(true===1);`** returns false

Control statements

- Same as in C++ or Java
  - `if else`
  - `for`
  - `for..in`
  - `while`
  - `do .. while`
  - `switch`

Getting input from user

- **`String prompt(question,defaultanswer)`**
- **`Example: prompt(“what is your name”,””);`**
- A prompt box pops up. The user can enter some text and click either "OK" or "Cancel" to proceed after entering text.
- If the user clicks "OK" the box returns the input value.
- If the user clicks "Cancel" the box returns **`null`**.

Function

- Like other programming languages, in JavaScript also we can define a reusable code-block which will execute when ever it is called.
- A function can be defined inside **<head>** , **<body>** or in a external file and can be called from anywhere after it has been read.

- Syntax

```
function functionname(var1,var2,...,varX)
{
//some code
}
```

Example

```
<html><head>
<script type="text/javascript”>
<!--
function display(x){
alert(x);}
-->
</script>
</head>
<body>
<script>
<!--
display(“hello”);
-->
</script>
</body></html>
```

`display()`can be defined anywhere even after the function call

Calling a function

- The above function can be called as
  - **`display();`**
  - **`display(“hello”);`**
  - Or **`display`** with any number of arguments

```
function display(x){
if(x==null)
x=“Greetings”;
alert(x) ;
}
```

- You can also pass values to a function that does not take any arguments!

```
<body>
<script type="text/javascript”>
display("hello");
display();
function display(x)
{
if(x==null)
x="Greetings";
alert(x) ;
}
function display(){
alert("Greet") ;
}
</script>
</body>
```

No overloading possible. If overloaded functions are provided, only the last defined function is considered.

Prints **Greet** for both the calls

Local and Global variables

- All the variables that are not explicitly declared are global.
- Local variables are created using **var** inside the function

```
<html><head>
<script>
total=0;            →
                          → Global variable
function sum(){
y=20;               →
var x=10;           →  Local variable
total=x+y;
}
function display(){
sum();
alert(total);
alert(y);
alert(x);           →  Error.
}
```

```
</script></head><body>
<script>
display();
</script>
</body></html>
```

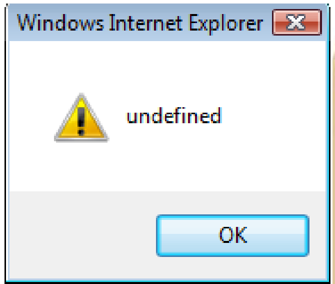
Object based programming in JavaScript

Creating objects in JavaScript

- Using built-in data type called **`Object`**
- Any number of properties can be added to an object at any time.

```
<SCRIPT type="text/JavaScript">
obj = new Object;
obj.x = 1;
obj.y = 2;
alert(obj.x + " "+ obj.y);
</SCRIPT>
```

What will **`alert(obj.k);`** display?



undefined

Class

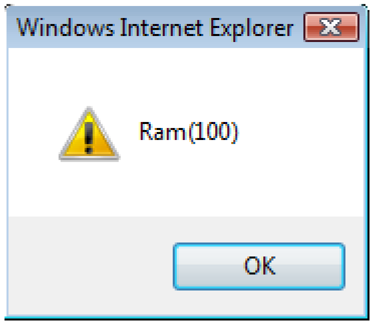
- A class is defined using function.
- When a function is called with the **new** operator, the function is considered as the constructor for that class!

```
<SCRIPT type="text/JavaScript">
function Person(name,eid){
this.name=name;
this.eid =eid;
}

p=new Person("Ram",100);
alert(p.name+ "("+ p.eid+")");
</SCRIPT>
```

constructor

attributes



Ram(100)

Methods

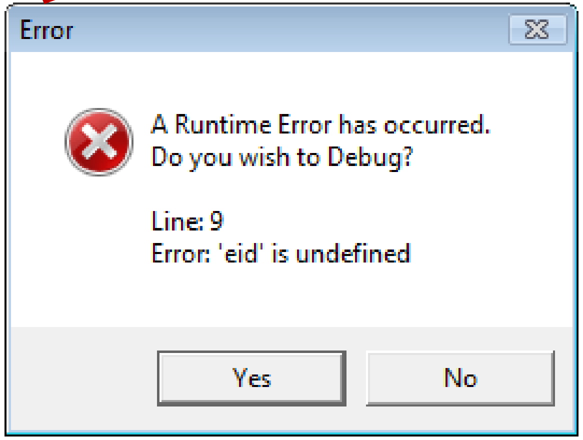
- To define a method **`classname>.prototype.<methodName>= function`**
- syntax is used.

```
<SCRIPT type="text/JavaScript">
        function Person(name,eid)
        {
        this.name=name;
        this.eid=eid;
        }
        Person.prototype.display = function(){
        alert(this.name+ "("+ this.eid+")");
        }
        Person.prototype.change = function(name){
        this.name=name;
        }
        p=new Person("Ram",100);
        p.display();
        p.change("Ramakrishna");
        p.display();</SCRIPT>
```

This is used to distinguish between local variable and member variables in C++ and Java. Is it not the same here?

Yes that is true in JavaScript too. But here, once you have defined member variable using `this`, you must always access it using `this` (even if there are no clashing local variables.

```
<HTML>
<BODY>
<SCRIPT type="text/JavaScript">
function Person(){
this.name="ss";
this.eid=22;
}
Person.prototype.display = function()
{
alert(name+ "("+ eid+")");
}
p=new Person("Ram",100);
p.display();
</SCRIPT>
</BODY></HTML>
```



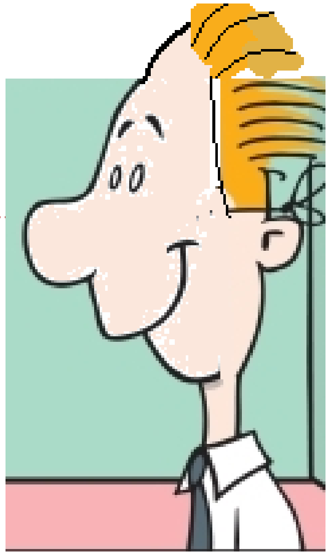
A Runtime Error has occurred.
Do you wish to Debug?

Line: 9
Error: 'eid' is undefined

Correct way is this:

```
alert(this.name+ "("+ this.eid+")");
```

What about access specifiers? Can I have private member



A local variable of the function is considered as private member.

But local variable is accessible only where function is defined. What if the member function wants to access the private member?

For this we need to make the method as **a Privileged Method**

Arrays

- Declaration: arrays are objects in JavaScript

```
    a= new Array(3);
    a[0]=1;
    a[1]=2;
    a[2]=3;
```

- A single array can hold any kind of data

```
    junk= new Array(3);
    junk[0]=“Sunday”;
    junk[1]=0 ;
    junk[2]=true;
```

- Initialized array

```
week=new Array(“sun”,”mon”,”tue”,”wed”,”thu”,”fri”,”sat”);
    alert(week[0]); □sun
    alert(week); □sun,mon,tue,wed,thu,fri,sat
```

- Array length:

```
    a.length□ 3
```

Adding elements to an array

- Ad using subscript. Array size is incremented dynamically

```
a= new Array();
a[4]=4;
a.length is 5
```

- **push()**to automatically adds elements to the end of the array

```
week=new Array(“sun”,”mon”,”tue”,”wed);
week.push('thu');
week.push('fri','sat');
alert(week); sun,mon,tue,wed,thu,fri,sat
```

- **unshift** to automatically adds elements to the beginning of the array

```
nums=new Array(3,4,5,6);
nums.unshift(0,1,2);
alert(nums);
```

Removing elements to an array

- **`pop()`** to remove element from the end of the array

```
week=new Array("sun","mon","tue","wed");
week.pop();
alert(week);sun,mon,tue
```

- **`shift()`** to remove element from the beginning of the array

```
nums=new Array(3,4,5,6);
nums.shift()
alert(nums); 4,5,6
```

splice()

- Allows adding/removing from any index position
- **splice(indexposition, numberOfItemsToDelete,[item(s) to be added]**
- **Example 1:**

```
nums=new Array(3,4,5,6);
nums.splice(2,1,10);
alert(nums); 3,4,10,6
```

- **Example 2:**

```
nums=new Array(3,4,5,6);
nums.splice(2,1);
alert(nums); 3,4,6
```

- **Example 3:**

```
nums=new Array(3,4,5,6);
nums.splice(1,2,11,12,13);
alert(nums); 3,11,12,13,6
```

for..in statement

```
<html><body>
<script type="text/javascript">
var x;
flowers=new Array(“rose”,”lilly”,”lotus”);
for (x in flowers)  {
  alert(flowers[x]);
  }
</script>
</body>
</html>
```

Accessing arguments of a function

- Any number of arguments can be passed to a JavaScript function.
- To access arguments inside the function, **`arguments`** member can be used.
- Functions in JavaScript is actually an object.

```
<script>
function sum(){
total=0
for(j=0;j<sum.arguments.length;j++){
total+=sum.arguments[j];}
alert(total);
}
</script>
```

Or just
`arguments`

DOM

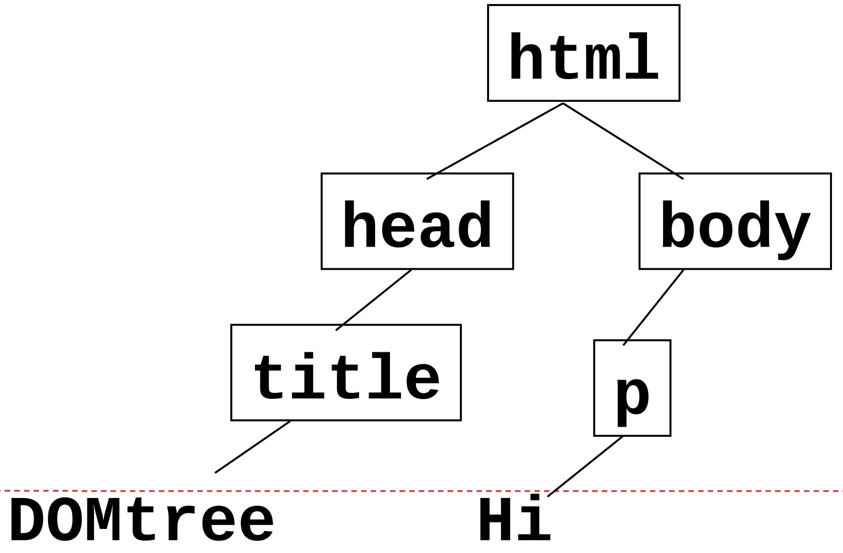
DOM

- W3C definition
  - The Document Object Model (DOM) is an application programming interface (*API*) for valid *HTML* and well-formed *XML* documents.
- It is a way in which elements of HTML, XHTML and XML can be parsed, accessed and modified.
- JavaScript provides API for DOM using which we can access HTML elements.

Nodes

- Every element/tag is considered as a NODE.
- A DOM tree can be built using the nodes.
- Example:

```
<html>
    <head>
   <title>DOMtree</title>
 </head>
    <body>
  <p> Hi</p>
  </body>
</html>
```



Traversing the DOM tree

- Getting an element:
  - **`document.getElementsByTagName(‘tag’)[index]`**
  - Example: to get to the **p** tag

`document.getElementsByTagName(‘p’)[0]`

- On any node following properties can be used to traverse
  - **`firstChild`**
  - **`lastChild`**
  - **`childNodes[index]`**
- To get to the first node (that is **`<html>`**)
  - **`document.firstChild`**
- To get the value of the node: **`nodeValue`**
- To get the name of the node: **`nodeName`**

We write **`document.getElementsByTagName.`** Is **`document`** an object in JavaScript?

That is right. **`document`** is a predefined object available to JavaScript which indicates the current document.

Getting an element by tag name

```
<html>
    <head>
        <title>DOMtree</title>
    </head>
    <body>
        <p> Hi</p>
    </body>
</html>
```

Ways to get to **<p>** node:

- **document.childNodes[0].childNodes[1].childNodes[0];**
- **document.getElementByTagName(‘p’)[0];**
- **document.body.childNodes[0];**

Example to print Hi

```
<html>
   <head>
        <title>DOMtree</title>
   </head>

<body>
   <p>Hi</p>

   <script>

   function f(){
        var x = document.getElementsByTagName(‘p')[0].firstChild;
        alert("Says "+ x.nodeValue);
            }
   f();
   </script>
</body>
</html>
```

Case insensitive

Tags must be available before the call.

`<p>Hi</p>` must appears before the call

getElementById

- To create a section or division in HTML **div** tag is used.
- **div** tag can be associated with an **id**.
- **document.getElementById('name')** can be used to get to the section referred to by the **div.**
- (It can also be used on any html element which is associated with id like **<input>** tag etc.)
- To change the content of **div**, **innerHTML** propery comes very handy.

```
<html>
   <head><title>ID</title></head>
   <body>
   Changing the below section:
   <div id="change">
   <p> This section is going to change</p>
   </div>
<script>
function f(){
var name=prompt("enter your name","");
var epara=document.getElementById('change');
epara.innerHTML ="<p><b>Hello, "+
name+"</b></p>";
epara.setAttribute('align',"center");
}
f();
</script>
</body></html>
```

Predefined JavaScript classes

String

- Creating strings
  - **`var str=“abc”;`**
  - or **`var str= new String(“abc”);`**
- **`String`** class has several members like to get length of the string, search for a pattern etc.
- Regular expression can be used with string to find if a pattern matches.

String members

Methods	Examples	Results
length	“hi”.length	2
toLowerCase()	“Hi”.toLowerCase()	hi
toUpperCase()	“hi”.toUpperCase()	HI
indexOf(searchText [,startposition])	“hello”.indexOf(“e”,0) or "hello".indexOf("e")	1
lastIndexOf(searchstring [,endpos])	"hello".lastIndexOf("l","hello".length) or "hello".lastIndexOf("l")	3
substring(startpos, [endpos])	“hello”.substring(1,3)	el
substr(start [,length])	“hello”.substr(1,3)	ell
charAt(indexPos)	“hello”.charAt(4)	o
slice(startpos, [endpos])	"hello".slice(3) or "hello".slice(3,5)	lo

0

Length of string

Methods	Examples	Results
charCodeAt()	‘A’.charCodeAt()	65
fromCharCode(n1,n2,..,nX)	String.fromCharCode(72,69,76,76,79)	HELLO
match(regexp)	"hello".match(/ll/)	ll
replace(regexp/substr, newstring)	"hello".replace(/ell/,"ipp")	hippo
search(regexp)	"hello".search(/ll/)	2
split(separator [, limit])	"hello".split(“") "red:green:blue".split(":") "red:green:blue".split(":", 2)	h,e,l,l,o red,green,blue red,green

How can you increment characters?

More on regular expression coming up

Regular Expression

- A regular expression (abbreviated to "regex”) is a set of pattern matching rules encoded in a string according to certain syntax rules.
- The syntax is complex but very powerful and allows lots of useful pattern matching than say simple wildcards *.

Creating regular expression

- **`RegExp`** class can be used to create regular expression strings
- Regular expression can also be created by putting them between `/ /`
  - `var x=/ll/; or`
  - `var reg=new RegExp("ll");`

```
alert("hello".match(reg));
```

Both return **`ll`**

Patterns

- **`\d`** is to match any digit
- **`\s`** is to match any whitespace character
- **`\w`** is to match any word character (letters, digits, or "_" (underscore))
- **`.`** Means any character
- **`[]`**: If we need a match to be any one of the characters among a list. Range such as a-z can also be specified here
- **`{}`**: character{n} where n is an integer
- **`[^]`** Typing a caret after the opening square bracket will negate the pattern.
- Quantifiers:
  - **`*`** : Zero or more occurrences
  - **`?`** : Zero or one occurrence
  - **`+`**: One or more occurrences

Examples

1. Octal
   1. `"07679".match(/0[0-7]/)` → 07
   2. `"07679".match(/0[0-7]+/)` → 0767
2. Protocol
   1. `"1.2.3.6.7".match(/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/)` →1.2.3.6
   2. `"1.2.3.".match(/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/)` → null
3. Number between 1 to 999
   1. `"04867".match(/[1-9]\d{0,2}/;`→486
   2. `“4807".match(/[0-9]{0,2}/);`→480

More Examples

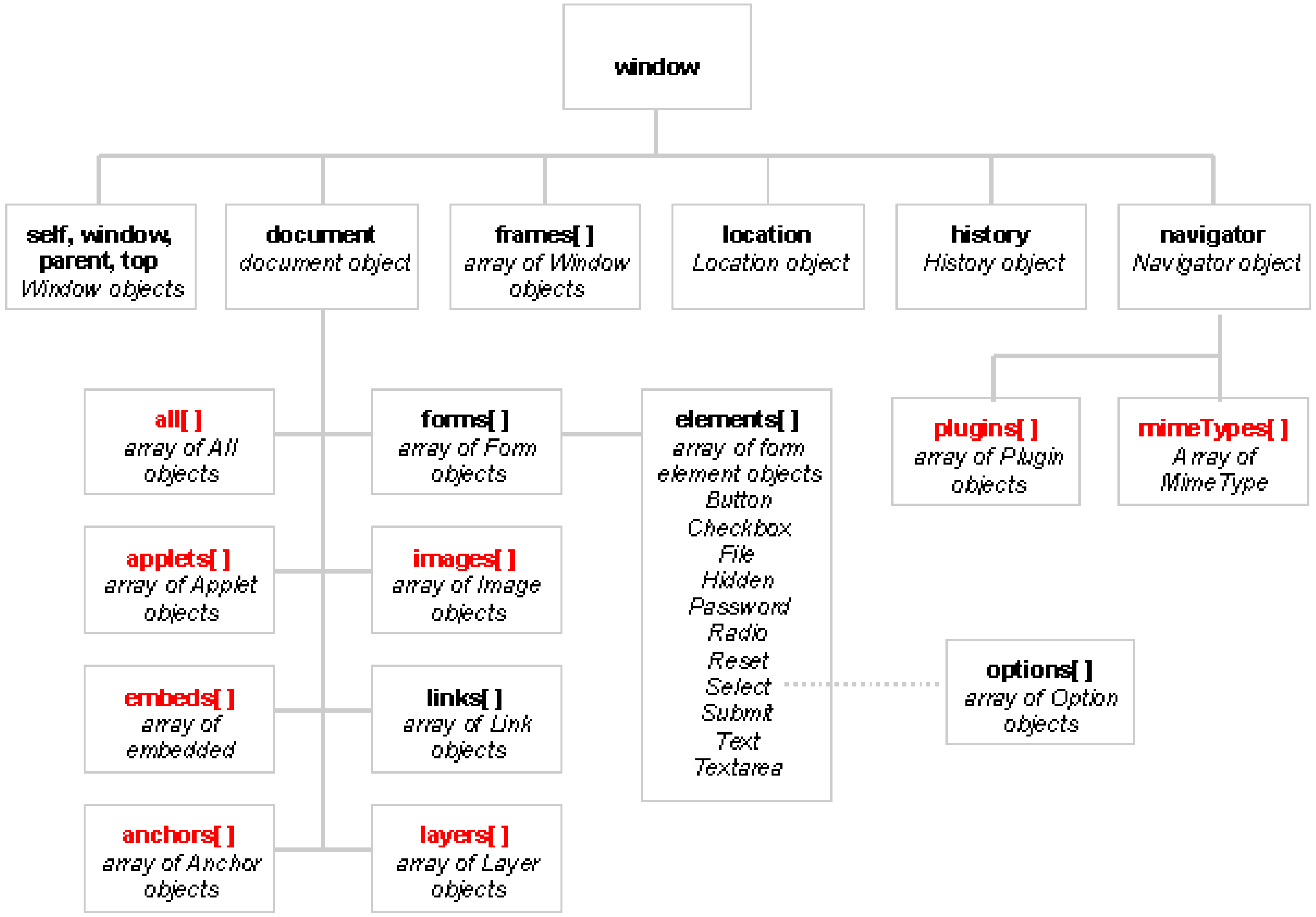
- Match an integer

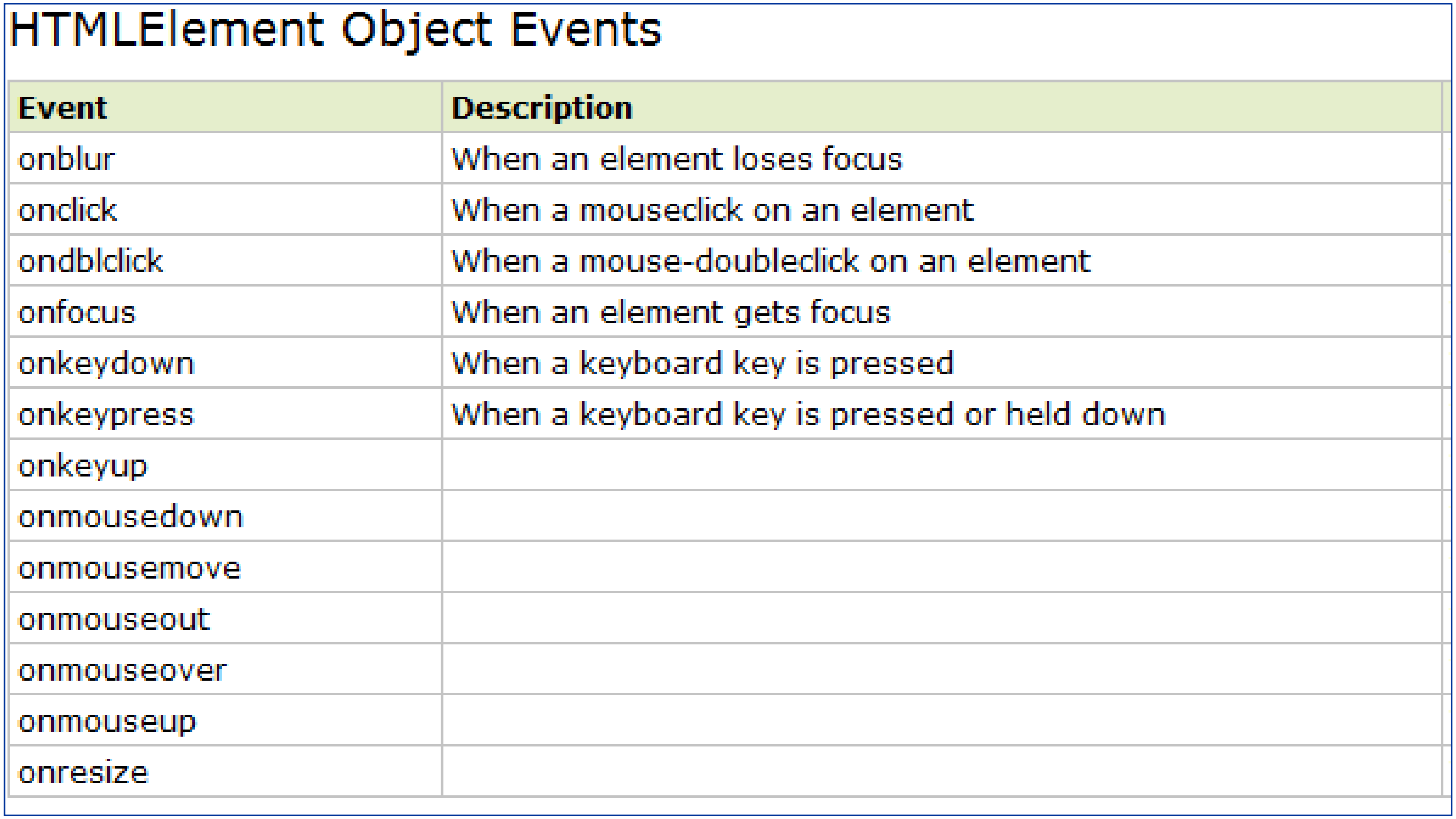
```
var _x= prompt("enter a no","1");
if(_x.match(/[+-]? \d+/)==_x)
alert("ok");
else alert("not ok");
```

- Verifying validity of JavaScript variable name:

```
var _x= prompt("enter var","");
if(_x.match(/[a-zA-Z$_][a-zA-Z$_\d]*/)==_x)
alert("ok");
else
alert("not ok");
```

JavaScript Object Hierarchy


window
self, window, parent, top
Window objects
document
document object
frames[ ]
array of Window objects
location
Location object
history
History object
navigator
Navigator object
all[ ]
array of All objects
forms[ ]
array of Form objects
elements[ ]
array of form element objects
Button
Checkbox
File
Hidden
Password
Radio
Reset
Select
Submit
Text
Textarea
plugins[ ]
array of Plugin objects
mimeTypes[ ]
Array of MimeType
applets[ ]
array of Applet objects
images[ ]
array of Image objects
options[ ]
array of Option objects
embeds[ ]
array of embedded
links[ ]
array of Link objects
anchors[ ]
array of Anchor objects
layers[ ]
array of Layer objects



HTMLElement Object Events

Event	Description
onblur	When an element loses focus
onclick	When a mouseclick on an element
ondblclick	When a mouse-doubleclick on an element
onfocus	When an element gets focus
onkeydown	When a keyboard key is pressed
onkeypress	When a keyboard key is pressed or held down
onkeyup	
onmousedown	
onmousemove	
onmouseout	
onmouseover	
onmouseup	
onresize	

- `window` object is the highest level JavaScript object which corresponds to the web browser window.
- `window` object contains `document` , `location`, `frames` and `history` object.
- By default the `window` object is automatically available to the java script code written for the browser.
- In other words `alert()` implicitly implies `window.alert()` and `prompt()` implicitly implies `window.prompt()`

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

function communicate()
{
alert("Hello");
s=prompt("What is your name", "xyz");
b=confirm("Do you want to see your name displayed in red color");

        if(b)
        {
        var text = document.createTextNode(s);
        document.body.appendChild(text);
        document.body.setAttribute("text","red");
        }
        }
        </script>

        </head>
        <body onUnload="alert('Bye!')">
        <script>
        communicate()
        </script>

        </body></html>
```

- **`Timer setTimeOut(expression, millsecs)`**
  - calls a function after the specified time in milliseconds
- **`clearTimeOut(timerobj)`**
  - clears the timeout that was set using the **`setTimeout`** function

```
<html><head>

<script type="text/javascript">

function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',1000);
}
</script>
</head>

<body onload="startTime()">

<div id="txt"></div>

</body>
</html>
```