

JavaScript

Whats it

- Interpreted lang with OO capabilities
- Syntax, resembles to C
- Most commonly used in browser, client side JavaScript, but actually its general language
- JavaScrips <> Java
- Before the name : LiveScript
- One of WWW Recommendation (please be <http://w3.com>'s lover)

- Was designed to **add interactivity** to HTML pages
- Is a **scripting language** (a scripting language is a lightweight programming language)
- JavaScript code is usually **embedded directly** into HTML pages
- JavaScript is an **interpreted language** (means that scripts execute without preliminary compilation)

What can it does?

- Gives HTML designers a programming tool
- Put **dynamic text** into an HTML page
- React to **events**
- Read and write HTML elements
- Be used to validate input data
- Be used to detect the visitor's browser
- Be used to create cookies

Place

- `<script> ... </script>` in `<head></head>`
or
- `<body> ... </body>`
- Sama seperti css, dapat diambil dari remote file

```
<head>
<script language="JavaScript" type="text/javascript"
        src="http://site1/JavaScript1.js">
</script>
<script language="JavaScript" type="text/javascript"
        src="myJS/myJavaScript2.js">
</script>
```

```
<body>
<script>
var fact=1;
for (i=1;i<10;i++){
fact=fact*i;
document.write(i+"!= "+fact+"<br>");
}
</script>
</body>
```

●

1!= 1
2!= 2
3!= 6
4!= 24
5!= 120
6!= 720
7!= 5040
8!= 40320
9!= 362880

Variables

- is name associated with value
- Create a variable with or without the "var" statement

var strname = some value

strname = some value

- Before use it, declar it.
 - var a; var msg;
 - var a=7; var msg="hi";

- **Scope**, the region of your program in which it is defined
 - **Global**, its defined everywhere in your Js code
 - **Local**, defined only in the body of function
- The lifetime of these variables starts when they are declared, and ends when the page is closed

Identifier

- Naming variables and function
 - Ex: a, \$a, a_b, _a
- Case sensitif
- Also have a lot reserved words
 - Ex: break, case, continue, do, else, if, this, true, null, ...etc

Datatypes

- Numbers
 - Integer: 0, 20, 100000000
 - Hexadecimal: 0xff
 - Octal: 0377
 - Floating: 1.1, .7, 1.7e20, 1.7E-20
- String
 - Literal: "this", 'this'

- Boolean
 - True, false
- Function
 - A piece executable code that defined by Js or predefined by Js
 - Function circle()
- Object
 - Collection of name value, usually referred to properties of object

- `var a = new Object()`
- `image.width`
- `document.write`
- `document.myform.`
- **Array**
 - **A collection of data values**
 - `var a = new array()`
 - `a[0] = 7;a[1]='seven';`

```
var scope="its global";    //global
function check(){
    var scope ="its local";    //local
    function check2(){
        var scope="its local nested";
        //nested local
        document.write(scope);    //print
        nested local
    }
    Check2();
}
check();
```

Primitive Reference

- Primitive has fixed size in variable

```
var a=1.7;  
var b=a;  
a =7;  
alert(b);
```

- Reference has no fixed size

```
var  
a=[1,2,3];  
var b=a;  
a[0]=17;  
alert(b);
```

Expression Operator

- `a = 7;`(make interpreter know the value)
- Arithmetic: `+, -, *, /, %, ++, --`
- Equality: `==, ===, !=, !==`
- Comparison: `<, >, <=, >=`
- String : `+`
- Logical: `and/&&, or/||, not/!`
- ect

Statement

- End with (;)
 - `a=7;`
- Compound of statements
 - `{`
 - `a=7;b=13; c=a*b;`
 - `}`

Control

- If <expression>

(statement)

else

(statement)

-

```
var username= "jono";  
if(username !=null)  
    alert("Hai "+username+" Welcome ");  
else{  
    username=prompt("Welcome, whats Ur name");  
    alert("Thanks, hello "+username);  
}
```

- `switch<expression>{
 (statement) }`
- `While <exp>
 (statement)`

```
switch(i){  
  case 1:  
    break;  
  case 2:  
    break;  
  case 3:  
    break;  
  default: // if all fails  
    break;  
}
```

```
var count=0;  
while (count<7){  
  document.write(count+" ");  
  count++;  
}
```

- do

(statement)

while<exp>

```
var c=0;
do
    document.write(c+"-");
while(c++<10);
```

- for (initialize;control;increment)

(statement)

```
for(i=0,j=1,k=0,f=0;i<10;i++,f=j+k,j=k,k=f){
document.write("Fibonacci("+i+ ")= "+f);
document.write("<br>");
}
```

```
var book = {};  
book.title = "Data Mining"; //set property of object book  
book.chapter1 = new Object();  
book.Chapter1.title = "Introduction of Data Mining";  
book.Chapter1.pages = 17;  
book.Chapter2 = {title: "Associate Rule Algorithm", pages: 27};  
  
alert("Outline: "+book.title+" "+  
"Chapter1 "+book.Chapter1.title+" "+  
"Chapter2 "+book.Chapter2.title);
```

- All objects in Js inherit from class object
- Every object has constructor property that refers to the constructor function
- ```
var dt = new Date();
dt.constructor == Date;
```

# Function

- a block in Js code that is defined once but may be invoked or execute any number of times
- Have **parameter or arguments** → local var, specified when the function is invoked, usually to compute a return value that become the value of function
- When invoked **on an object, function is called method**
- Familiar with concept procedure/ subroutine

- A JavaScript function contains some code that will be executed only by an event or by a call to that function
  - To keep the browser from executing a script as soon as the page is loaded, you can write your script as a function
- You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).
- Functions can be defined either <head> or <body> section
  - As a convention, they are typically defined in the <head> section

```
function BoxVolume(l,w,h){
 var V;
 V=l*w*h;
 document.write(V);
}
BoxVolume(3,3,3);
```

```
function distance(x1,x2,y1,y2){
 var dx=x2-x1;
 var dy=y2-y1;
 return Math.sqrt((dx*dx)+(dy*dy));
}
```

```
function factorial(x){
 if (x<=1)
 return 1;
 return x*factorial(x-1);
}
document.write(factorial(7));
```

## Return, turn to exist parameter

- Nested

```
function hypo(a,b){
 function square(x){return x*x}
 return Math.sqrt(square(a)+square(b));
}
```

- Function literal, unnamed function

```
function square(x){return x*x} // f.statement
var square = function(x){return x*x;}; //f.literal
```

- Function as Data, mean can be assigned to variables

```
function square(x){return x*x;}
var a = square(5); // w'll have 16
var b = square; // b refers to the same function that square does
var c = b(7); // w'll have 49
```

- As property of object

```
var o = new Object;
o.square = function(x){return x*x;} //as object property, f.literal
y = o.square(7); // 49
```

- Function as Method

```
o.meth = f; //define method meth() of object
o.meth(x,x*x);
```

```
var calculator = { //its object
 op1:1,
 op2:7,
 compute:function(){
 this.sum = this.op1+this.op2;
 }
};
calculator.compute(); //1+7
document.write((calculator.sum));
```

- A JavaScript function contains some code that will be **executed only by an event or by a call** to that function
  - > To keep the browser from executing a script as soon as the page is loaded, you can write your script as a function
- May **call a function from anywhere within the page** (or even from other pages if the function is embedded in an external .js file).
- Functions can be defined either <head> or <body> section
  - > As a convention, they are typically defined in the <head> section

# Class, Constructor

- Class of object, individual object as instance of class. Class in Js just like **pseudo class**, not like in Java or C++. Its **constructor or prototype**
- **A function designed to be used with new operator is called a constructor.**
- Constructor **use to initialize a newly created object**, setting any properties that need to be set before the object is used
- Constructor **usually dont have** return value

```
//initiate the constructor
function Rectangle(w,h){
 this.width = w;
 this.height = h;
 //note, no return value
}
```

```
 //new objects
 var rect1 = new Rectangle(1,7);
 var rect2 = new Rectangle(7,17);
```

# Client side Js

## The browser environment

- Window object that serves as global object and global execution context for client-side Js code
- Hierarchy and DOM that forms a part of it
- Event-driven programming model

# Windows as global execution

- Web browser = display HTML
- In Js, document object represent HTML doc
- Window object defines a number of properties and method that allow to manipulare window browser
- Has 2 self\_ref: window and self
- As **global object**, all global var are defined as properties of the window

```
var a = 7;
window.a = 7;//same
```

# Hierarchy and DOM

- Window → self/window/parent/top(various window object), navigator, frames[](array of window object), location, history, **document**, screen
- Document → **forms[]**, anchors[], links[], images[], applets[]
- Forms → **elements[]**
- Element → options[]

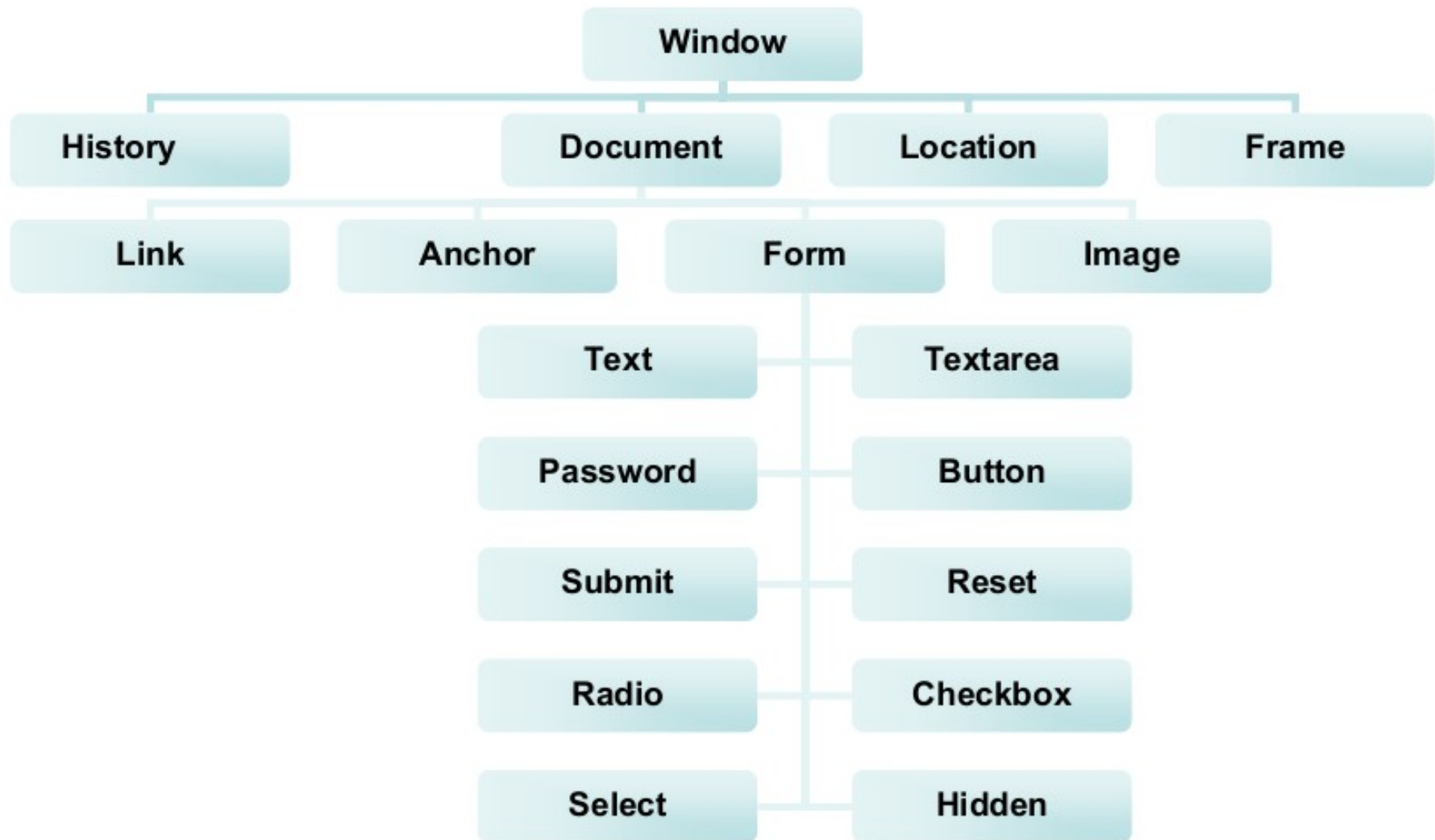
```
window.document.forms[0];
parent.frames[0].document.forms[0].element[3].options[2].text;
```

- The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML documents
- All HTML elements, along with their containing text and attributes, can be accessed through the DOM.
  - The contents can be modified or deleted, and new elements can be created.
- The HTML DOM is platform and language independent
  - It can be used by any programming language like Java, JavaScript, and VBScript

# DOM Object

- Anchor object
- Document object
- Event object
- Form and Form Input object
- Frame, Frameset, and IFrame objects
- Image object
- Location object
- Navigator object
- Option and Select objects
- Screen object
- Table, TableHeader, TableRow, TableData objects
- Window object

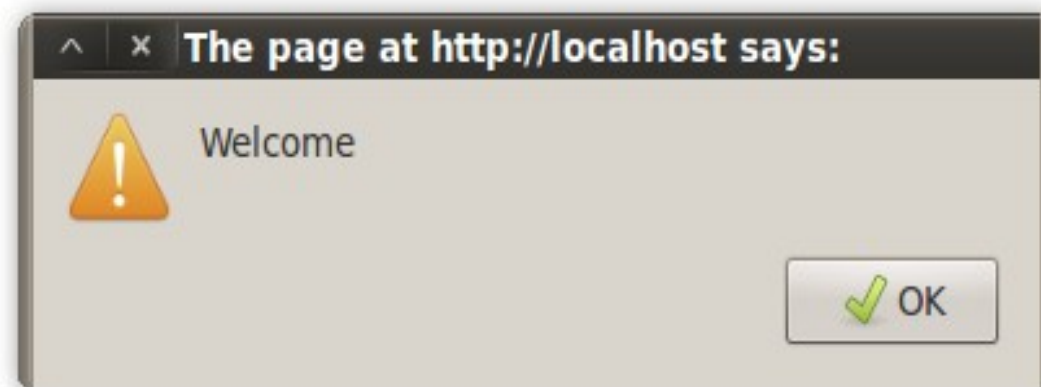
# DOM Hierarchy



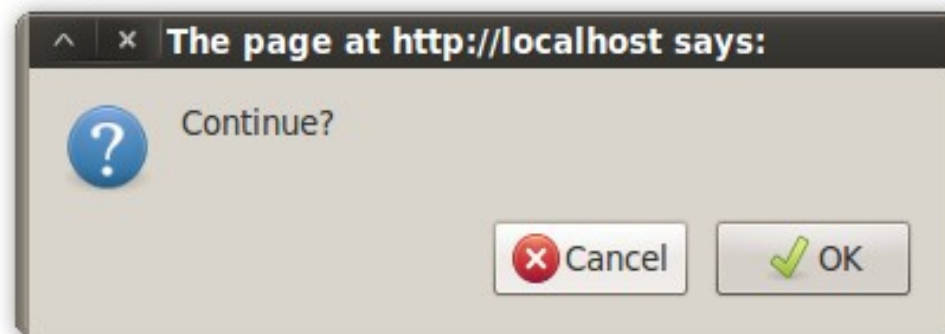
# JavaScript Popup Boxes

- Alert box
  - User will have to click "OK" to proceed
  - `alert("sometext")`
- Confirm box
  - User will have to click either "OK" or "Cancel" to proceed
  - `confirm("sometext")`
- Prompt box
  - User will have to click either "OK" or "Cancel" to proceed after entering an input value
  - `prompt("sometext","defaultvalue")`

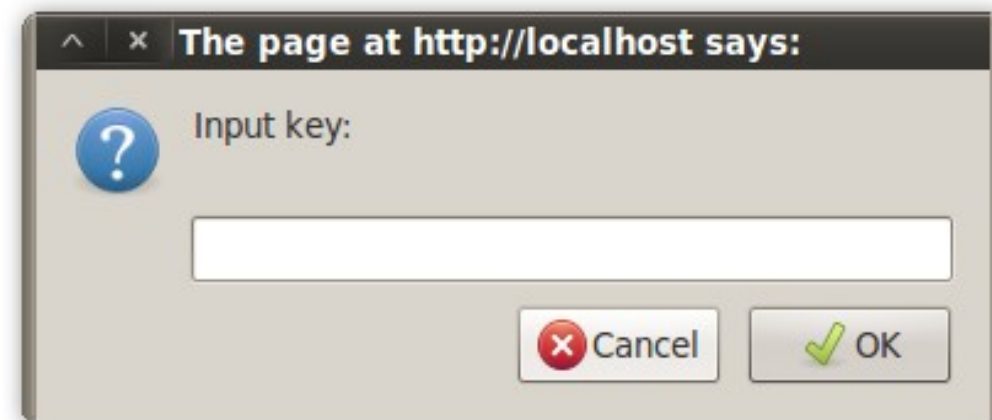
```
<head>
 <script type="text/javascript">
 function display() {
 alert("Welcome")
 }
 </script>
</head>
<body onload="display()">
```



```
<head>
 <script type="text/javascript">
 function display() {
 confirm("Continue?")
 }
 </script>
</head>
<body onload="display()">
```



```
<head>
 <script type="text/javascript">
 function display() {
 prompt("Input key:")
 }
 </script>
</head>
<body onload="display()">
```



# Event Driven

- HTML contain an embedded graphical user interfaces (GUI), so client side Js uses the event driven programming model
- A kind of **interactivity client side Js**, dynamic
- **Event handler**
- Few of them: onclick, onmousedown, onmouseup, etc...

```
<INPUT type="checkbox" name="option" value="choice1" onclick="choice1=this.checked;">
Choice 1
```

- Every element on a web page has certain events which can trigger invocation of event handlers
- Attributes are inserted into HTML tags to define events and event handlers
- Examples of events
  - A mouse click
  - A web page or an image loading
  - Mousing over a hot spot on the web page
  - Selecting an input box in an HTML form
  - Submitting an HTML form
  - A keystroke

# Event

- onabort - Loading of an image is interrupted
- onblur - An element loses focus
- onchange - The content of a field changes
- onclick - Mouse clicks an object
- ondblclick - Mouse double-clicks an object
- onerror - An error occurs when loading a document or an image
- onfocus - An element gets focus
- onkeydown - A keyboard key is pressed

- onkeypress - A keyboard key is pressed or held down
- onkeyup - A keyboard key is released
- onload - A page or an image is finished loading
- onmousedown - A mouse button is pressed
- onmousemove - The mouse is moved
- onmouseout - The mouse is moved off an element
- onmouseover - The mouse is moved over an element
- onmouseup - A mouse button is released

- onreset - The reset button is clicked
- onresize - A window or frame is resized
- onselect - Text is selected
- onsubmit - The submit button is clicked
- onunload - The user exits the page

# onload & onUnload

- The onload and onUnload events are triggered when the user enters or leaves the page
- The onload event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information
- Both the onload and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page
- Its event on "Body" object

# onFocus, onBlur, onChange

- The onFocus, onBlur and onChange events are often used in combination with validation of form fields.
- Example: The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email"
onchange="checkEmail()">;
```

```

<head>
<script type="text/javascript">
function upCase() {
var x=document.getElementById("fname").value
document.getElementById("fname").value=x.toUpperCase()
}
</script>
<style>
.class1{background-color: orange;}
</style>
</head>
<body>
Input name:
<input type="text" id="fname" onblur="upCase()" class="class1">
</body>

```

Input name: DEWI

- The `onSubmit` event is used to validate all form fields before submitting it.
- Example: The `checkForm()` function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be canceled. The function `checkForm()` returns either `true` or `false`. If it returns `true` the form will be submitted, otherwise the submit will be cancelled.

```
<head>
<script type="text/javascript">
function val() {
//
}
</script>
<style>
.class1{background-color: orange;}
</style>
</head>
<body>
<form onsubmit="return val()" name="form1" action="on.html">
Input name:|
<input type="text" id="fname" class="class1">

<button type="submit" name="sub">send</button>
</form>
</body>
```

---

Input name: Sumami

send

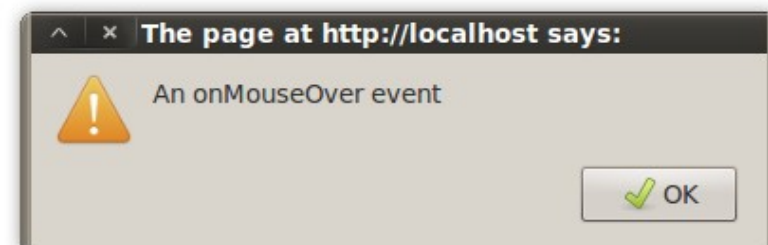
- onMouseOver and onMouseOut are often used to create "animated" buttons.
- Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver
event');return false">
```

```
<head>
<script type="text/javascript">
</script>
 <style>
 .class1{background-color: orange;}
 .class2{background-color: yellow;}
 </style>
</head>
<body>
 <a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver event');return false">
 Click Me!

</body>
```

**Click Me!**



# Object

- 3 different ways
  - Create a direct instance of an object by using built-in constructor for the Object class
  - Create a template (Constructor) first and then create an instance of an object from it
  - Create object instance as Hash Literal

# Creating a Direct Instance of a JavaScript Object

- By invoking the built-in constructor for the Object class
  - `personObj=new Object(); // Initially empty`
- Add properties to it
  - `personObj.firstname="John";`  
`personObj.age=50;`
- Add an anonymous function to the personObj
  - `personObj.Age=function()`  
`{alert("This age is " + this.age);}`  
`//You can call then tellYourage function as following`  
`personObj.Age();`

- Add a pre-defined function

```
function Age(){
 alert("The age is" + this.age);
}
```

```
personObj.Age=Age;
```

- Note that the following two lines of code are doing completely different things

```
// Set property with a function
```

```
personObj.Age=Age;
```

```
// Set property with returned value of the function
```

```
personObj.Age=Age();
```

# Creating a template of a JavaScript Object

- The template defines the structure of a JavaScript object in the form of a function
- You can think of the template as a constructor

```
function Person(firstname,lastname,age,eyecolor) {
 this.firstname=firstname;
 this.lastname=lastname;
 this.age=age;
 this.Age=function(){
 alert("This age is " + this.age);
 }
}
```

- Once you have the template, you can create new instances of the object

```
myFather=new Person("John","Doe",50,"blue");
```

```
myMother=new Person("Sally","Rally",48,"green");
```

- You can add new properties and functions to new objects

```
myFather.newField = "some data";
```

```
myFather.myfunction = function() {
```

```
 alert(this["fullName"] + " is " + this.age);
```

```
}
```

# Creating JavaScript Object as a Hash Literal

- Create personObj JavaScript object

```
var personObj = {
 firstname: "John",
 lastname: "Doe",
 age: 50,
 tellYourage: function () {
 alert("The age is " + this.age);
 }
 tellSomething: function(something) {
 alert(something);
 }
}
```

- personObj.tellYourage();
- personObj.tellSomething("Life is good!");

- Its composite datatypes, aggregate multiple values into single unit
- Unordered collection of properties
- `var none = {} //empty`

```
var dot = {x:1,y:7};
```

```
var Jono = {
 "name":"Jono",
 "age":"17",
 "occupation":"student",
 "phone":"62 55555",
 "email":"jono@jono.com",
};
```

# Some suggestions

- Just like the other languages, Js **has built in function**, along with practice it will be easier to understand and Look Up Again the Compl Ref
- U can put it **anywhere in Ur page**
- **Lack in security**, so U know U wanna use Js where and for what
- **Not full featured in OO**, dont use it for large development project

# Ex

```
<head>
 <style>
 .class1{background-color: orange;}
 </style>
</head>
<body>
<script type="text/javascript">
 todaynow = new Date(); //get day and time
 document.write("<p> " + todaynow + "</p>");
 t = "am";
 h = todaynow.getHours(); //date function
 if (h > 12) {
 h -= 12;
 t = "pm"
 }
 else if (h == 0) {
 h = 12;
 }
 document.write("<p> " + h + ":" +
 todaynow.getMinutes() + ":" +
 todaynow.getSeconds() + " " +
 t + "</p>");
</script>
</body>
```

Tue Mar 22 2011 05:27:23 GMT+0700 (WIT)

5:27:23 am

```
<body>
<script type="text/javascript">
 todaynow = new Date(); // date and time today
 newYear = new Date(2012,0,1); //get defin date at 12 am
 s = Math.round((todaynow-newYear)/1000); // pembulatan
 d = Math.floor(s / 86400); //measure
 s -= d*86400;
 h = Math.floor(s / 3600);
 s -= h*3600;
 m = Math.floor(s / 60);
 s -= m*60;
 document.write(d + " days, " +
 h + " hours, " +
 m + " minutes, and " +
 s + " seconds."); //rest or distance to new ye
</script>
```

-285 days, 5 hours, 32 minutes, and 51 seconds... to 2012 New Year

```
</head>
<script language="JavaScript">
function msg(){
var now = new Date();
var s = now.getSeconds(); // or minutes
var Quote = [//array msg
"Tahukah anda Newton di masa kecilnya dianggap idiot oleh gurunya dia dididik khusus oleh ibunya sendiri",
"Einstein awalnya hanya seorang pekerja rendahan di bidang elektronik yang gemar menulis di majalah sains",
"Ibnu Sina atau dikenal juga dengan nama Avisenna biasa belajar hanya berteman air putih untuk menghilangkan kantuknya"
];
var i = Quote.length;
document.write(Quote[s % i]);
setTimeout("msg()",1000); //to make a simple loop
}
</script>
<body>
<script>msg();</script>
```