# 3 - Más Acerca de las Funciones

# Funciones

```
function hola(msg) {
    alert('Hola ' + msg);
}

hola('Mundo');
```

# Funciones como variables

```
var hola = function (msg) {
    alert('Hola ' + msg);
}

hola('Mundo');
```

# Funciones como variables

```
var hola = function (msg) {
    alert('Hola ' + msg);
}

hola('Mundo');
```

- La función seguirá las mismas reglas que cualquier otra variable en Javascript.

# Funciones como variables

```
var hola = function (msg) {
    alert('Hola ' + msg);
}

hola('Mundo');
```

- La función seguirá las mismas reglas que cualquier otra variable en Javascript.
- Tendrá el mismo alcance y puede ser pasado a otras funciones.

# Pasando Funciones a Funciones

```
var mostrar = function(que) {
    que('funcion mostrar');
}

var hola = function (quien) {
    alert('Hola '+quien);
}

mostrar(hola) // "Hola funcion mostrar"
```

# Pasando Funciones a Funciones

- Una Función puede pasarse como cualquier otra variable
- Cualquier cosa que pueda contener un valor puede contener una función.

# Pasando Funciones

```
function hacerAlgo() {
    alert('diste click!');
}

document.onclick=hacerAlgo;
```

# Pasando Funciones

```
function presionaBoton() {
    alert('presionaste boton!');
}

function hazAlgo() {
    return presionaBoton;
}

document.onclick = hazAlgo();
```

# Pasando Funciones

```
var prueba = function () {
    return "Esto es un String";
}

var pruebaCopia = prueba;
var pruebaStr = prueba();
var probando = pruebaCopia();
```

# Funciones Anónimas

```
document.onmousedown = function()
 { alert("Presionaste el boton"); }
```

# Autoinvocación de Funciones

```
var x = (function () {return(5+6)})();
document.writeln(typeof(x)+'<BR>');
document.writeln(x);
```

# Autoinvocación de Funciones

```
var x = (function () {return(5+6)})();
document.writeln(typeof(x)+'<BR>');
document.writeln(x);
```

# Autoinvocación de Funciones

```
function alerta(msg) {
    alert(msg);
}

alerta( (function(animal1,animal2) { return animal1
+ ' ama ' + animal2; } ) ('gato', 'perro') );
```

# Autoinvocación de Funciones

```
function mostrar(msg) {
   document.writeln(msg+'<br>');
}

function crearCadena(num1, num2) {
   return num1 + ' + ' + num2 + ' = ' + (num1+num2);
}

for (i=0; i<10; i++) {
  mostrar( (crearCadena) (i, i+5) );
}
```

# Autoinvocación de Funciones

```
function mostrar(msg) {
   document.writeln(msg+'<br>');
}

function crearCadena(num1, num2) {
   return num1 + ' + ' + num2 + ' = ' + (num1+num2);
}

for (i=0; i<10; i++) {
  mostrar( crearCadena(i, i+5) );
}
```

# Apuntador de las Funciones

```
var original = function () {
   alert('Hola Mundo');
}

var copia = original;

var original = function () {
   alert('adios mundo');
}

copia();
```

# Visibilidad (*scope*) de una Función

```
var global = function () {
   alert('Hola Mundo');
}

var contenedora = function() {

  var subFuncion = function() {
    alert("Soy Local");
    global();
  }

  global();
  subFuncion();
}

contenedora();
subFuncion();
```

# Visibilidad (*scope*) de una Función

```
var global = function () {
   alert('Hola Mundo');
}

var contenedora = function() {

  subFuncion = function() {
    alert("Soy Local");
    global();
  }

  global();
  subFuncion();
}

contenedora();
subFuncion();
```

# Propiedades y Métodos de las Funciones

- **Propiedades:**
  - arguments -- un Array que contiene los argumentos pasados a la función
    - arguments.length
    - arguments.callee – Apuntador a la función
  - length – El número de argumentos que la función espera
  - Constructor
  - prototype

# Propiedades y Métodos de las Funciones

- Métodos:
  - apply – Permite pasar argumentos más fácilmente
  - call – Llama una función dentro de un contexto diferente.
  - toSource – Regresa la fuente de la función como String.
  - toString
  - valueOf

# Objetos

# Instanciar un Objeto

```
obj = new Object;
obj = new Object();

obj = new Array;

obj = new miObjeto;
obj = new miObjeto();
```

# Objetos Simples

- El constructor orientado a objetos más simple en Javascript es *Object*.

# Objetos Simples

- El constructor orientado a objetos más simple en Javascript es Object.
- En Javascript los objetos están implementados como una colección de propiedades con nombre.

# Objetos Simples

- El constructor orientado a objetos más simple en Javascript es *Object*.
- En Javascript los objetos están implementados como una colección de propiedades con nombre.
- Javascript permite la creación de cualquier número de propiedades en un objeto en cualquier momento.

# Objetos Simples

- El constructor orientado a objetos más simple en Javascript es Object.
- En Javascript los objetos están implementados como una colección de propiedades con nombre.
- Javascript permite la creación de cualquier número de propiedades en un objeto en cualquier momento.
- NO tienen que estar predefinidas en el constructor o en la declaración del objeto.

# Objetos Simples

```
obj = new Object;
obj.x = 1;
obj.y = 2;
```

# Función Constructora

```
function Foo() {
    this.x = 1;
    this.y = 2;
}

obj = new Foo;
```

# Función Constructora

```
function Foo(x, y) {
    this.x = x;
    this.y = y;
}

var obj1 = new Foo(5, 3);
var obj2 = new Foo(1, 2);
```

# Función Constructora

```
function Foo(x, y) {
    this.x = x || 1;
    this.y = y || 2;
}
var obj1 = new Foo();
var obj2 = new Foo(5);
var obj3 = new Foo(3, 4);
```

# Función Constructora

```
function Foo(x, y) {
    this.x = x || 1;
    this.y = y || 2;
    this.suma = function() {
        return this.x + this.y;
    }
}

var obj = new Foo(6);
alert(obj.suma()); // 8
```

# Función Constructora

```
function Foo(x, y) {
    this.x = x || 1;
    this.y = y || 2;
    this.suma = suma;
}

function suma() {
    return this.x + this.y;
}

var obj = new Foo(6);
alert(obj.suma()); // 8
```

# Función Constructora

```
var Foo = function(x, y) {
    this.x = x || 1;
    this.y = y || 2;
    this.suma = function() {
        return this.x + this.y;
    }
}

var obj = new Foo(6);
alert(obj.suma()); // 8
```

# Objetos

```
function Foo() {
    this.x = 6;
}

var obj1 = new Foo;
var obj2 = new Foo;

obj1.y = 3;

document.writeln(obj1.x);
document.writeln(obj2.x);
document.writeln(obj1.y);
document.writeln(obj2.y);
```

# Objetos

```
function Foo() {
    this.x = 6;
}

var obj1 = new Foo;
var obj2 = new Foo;

obj1.y = 3;

document.writeln(obj1.x); // 6
document.writeln(obj2.x);
document.writeln(obj1.y);
document.writeln(obj2.y);
```

# Objetos

```
function Foo() {
    this.x = 6;
}

var obj1 = new Foo;
var obj2 = new Foo;

obj1.y = 3;

document.writeln(obj1.x); // 6
document.writeln(obj2.x); // 6
document.writeln(obj1.y);
document.writeln(obj2.y);
```

# Objetos

```
function Foo() {
    this.x = 6;
}

var obj1 = new Foo;
var obj2 = new Foo;

obj1.y = 3;

document.writeln(obj1.x); // 6
document.writeln(obj2.x); // 6
document.writeln(obj1.y); // 3
document.writeln(obj2.y);
```

# Objetos

```
function Foo() {
    this.x = 6;
}

var obj1 = new Foo;
var obj2 = new Foo;

obj1.y = 3;

document.writeln(obj1.x); // 6
document.writeln(obj2.x); // 6
document.writeln(obj1.y); // 3
document.writeln(obj2.y); // undefined
```

# La Propiedad *prototype*

- Cada objeto puede heredar propiedades de otro objeto llamado *prototype*
- Solo puede ser alcanzada a través del nombre del constructor.

# La Propiedad *prototype*

```
function Foo(x) { this.x = x || 5; }

var o1 = new Foo;
var o2 = new Foo(4);

Foo.prototype.y = 6;

document.writeln(o1.x); // 5
document.writeln(o2.x); // 4
document.writeln(o1.y); // 6
document.writeln(o2.y); // 6
```

# La Propiedad *prototype*

```
function Foo(x, y) {
  this.x = x || 5;
  this.y = y || 3;
}

var o1 = new Foo;
var o2 = new Foo(4, 3);

Foo.prototype.suma = function() {
    return this.x + this.y
};

document.writeln(o1.suma()); // 2
document.writeln(o2.suma()); // 1
```

# Herencia

```
function A() {
  this.x = 3;
}
B.prototype = new A;
B.constructor = B;

function B() {
  this.y = 4;
}
var b = new B;

alert(b instanceof B); // true
alert(b instanceof A); // true
```

# Uso de los Objetos más utilizados en Javascript

# Objeto Math - Métodos

- abs(x) - Devuelve valor absoluto de x
- acos(x) – Arco-coseno de x
- asin(x) – Arco-seno de x
- atan(x) – Arco-tangente de x
- atan2(x, y) – Devuelve el ángulo de un punto (x,y)
- ceil(x) – Valor de x redondeado al entero superior
- cos(x) – coseno de x

# Objeto Math - Métodos

- exp(x) – Devuelve el valor de E elevado a x
- floor(x) – Devuelve x redondeado al entero inferior
- log(x) – Logaritmo natural de x
- max(x, y) – Devuelve el mayor de x o y
- min(x, y) – Devuelve el menor de x o y
- pow(x, y) – x elevado a y potencia
- random() - número aleatorio entre 0 y 1

# Objeto Math - Métodos

- round(x) – x redondeado al entero más cercano
- sin(x) – seno de x
- sqrt(x) – raíz cuadrada de x
- tan(x) – tangente de x

# Objeto Math - Constantes

- E – Valor de la constate de Euler
- LN2 – Logaritmo natural de 2
- LN10 – Logaritmo natural de 10
- LOG2E – Logaritmo base-2 de E
- LOG10E – Logaritmo base-10 de E
- PI – Devuelve PI
- SQRT1_2 – Raíz cuadrada de ½
- SQRT2 – Raíz cuadrada de 2

# Jerarquía de los objetos de una Documento HTML

- Window
  - History
  - Location
  - document `<body>...</body>`
    - anchor `<a name="...">...</a>`
    - applet `<applet>...</applet>`
    - area `<map>...</map>`
    - form `<form>...</form>`
    - image `<img src="..." />`
    - link `<a href="...">..</a>`
    - plugin `<embed src="..." />`
  - frame <frame>
  - navigator

# Objeto history

- Se encarga de almacenar una lista con los sitios por los que se ha estado navegando.
- Se usa principalmente para movernos hacia  atrás y adelante en dicha lista
- Uso: history.propiedad

# Objeto history - Propiedades

- current – contiene URL completa de la entrada actual del historial
- next – cadena del siguiente URL
- length – Entero que contiene el número de entradas del historial
- previous – cadena de la URL anterior

# Objeto history - Métodos

- back() - Vuelve a cargar la URL anterior
- forward() - Vuelve a cargar la URL siguiente
- go(posicion) – Carga el URL especificado por posicion, si posicion es entero indica la posición relativa dentro del historial, si posicion es una cadena indica la URL que esta en el historial

# Objeto location

- Contiene la URL actual así como algunos datos de interés respecto a esta URL.
- Su finalidad principal es, por una parte, modificar el objeto location para cambiar a una nueva URL, y extraer los componentes de dicha URL de forma separada.

- protocolo://maquina_host[:puerto]/camino_al_recurso

# Objeto location - Propiedades

- hash – Cadena que contiene el nombre del enlace, dentro de la URL.
- host - Cadena que contiene el nombre del servidor y el número del puerto, dentro de la URL.
- hostname - Cadena que contiene el nombre de dominio del servidor (o la dirección IP)
- href - Cadena que contiene la URL completa.

# Objeto location - Propiedades

- pathname - Cadena que contiene el camino al recurso.
- port - Cadena que contiene el número de puerto del servidor.
- protocol - Cadena que contiene el protocolo utilizado (incluyendo los dos puntos)
- search - Cadena que contiene la información pasada en una llamada a un script

# Objeto location - Métodos

- reload() - Vuelve a cargar la URL especificada en la propiedad href del objeto location.

- replace(cadenaURL) - Reemplaza el historial actual mientras carga la URL especificada en cadenaURL.

# Objeto screen - Propiedades

- availHeight – Regresa la altura de la pantalla (excluyendo la barra de tareas)
- availWidth - Regresa el ancho de la pantalla (excluyendo la barra de tareas)
- colorDepth – Regresa la profundidad de bits de la paleta de colores de la pantalla
- height – La altura de la pantalla
- width – La anchura de la pantalla

# Objeto navigator - Propiedades

- Este objeto nos da la información relativa al navegador que esté utilizando el usuario

- appCodeName - Nombre del código del cliente.
- appName – Nombre del cliente

# Objeto navigator - Propiedades

- appVersion - Información sobre la versión del cliente.
- language - información sobre el idioma de la versión del cliente
- mimeTypes - Array que contiene todos los tipos MIME soportados por el cliente.

# Objeto navigator - Propiedades

- platform - plataforma sobre la que se está ejecutando el programa cliente
- plugins - Array que contiene todos los plug-ins soportados por el cliente
- UserAgent - abecera completa del agente enviada en una petición HTTP. Contiene la información de las propiedades appCodeName y appVersion.

# Objeto navigator - Métodos

- javaEnabled() - Devuelve true si el cliente permite la utilización de Java, en caso contrario, devuelve false

# Objeto document

- Cada documento HTML cargado en el navegador se convierte en un objeto document
- Provee acceso a todos los elementos en una página HTML donde se ejecuta el script
- El objeto document también forma parte del objeto window y puede ser accesado a través de window.document

# Objeto document - Propiedades

- cookie – Establece o regresa todas las cookies asociadas al documento actual
- domain – Regresa el nombre del dominio
- lastModified – Fecha y hora de la última modificación del documento

# Objeto document - Propiedades

- title – Título del documento actual
- URL – URL del documento actual

# Objeto document - Métodos

- getElementById(id) – Regresa una referencia del primer objeto con id especificado
- getElementsByName(name) – Regresa una colección de objetos en el nombre especificado por name
- getElementsByTagName(tag) – Regresa una colección de objetos con tag especificado.

# Objeto document - Métodos

- write(cadena) – Escribe la cadena en el documento
- writeln(cadena) – Idéntico al write() solo que agrega un caracter de salto del línea al final

# Objeto document - Colecciones

- anchors[] - Referencia a todas los objetos anclas en un document
- forms[] - Referencia a todos los objetos de un formulario
- images[] - Referencia a todos los objetos de imagen
- links[] - Referencia a todas las areas y links del documento

# Objeto window

- Representa una ventana abierta en el navegador.
- El objeto window apunta  a la ventana actual
- Si un documento contiene frames, el navegador crea un objeto window para el documento HTML y un objeto window adicional para cada frame

# Objeto window - Colecciones

- frames[] - Contiene una referencia a todos los frames en la ventana

# Objeto window – Propiedades

- closed - Regresa si una ventana ha sido o no cerrada
- document – Referencia al objeto document
- history – Referencia al objeto history
- length – Número de frames en la ventana
- location – Referencia al objeto location

# Objeto window – Propiedades

- name – Establece o regresa el nombre de la ventana
- opener – Referencia a la ventana que creo la ventana actual
- parent – Referencia al objeto window padre o que contiene el frameset
- self – Referencia a la ventaja actual

# Objeto window – Propiedades

- status – Establece texto en la barra de estado
- top – Regresa la ventana en el nivel más superior

# Objeto window – Métodos

- alert(mensaje) – Muestra mensaje de alerta
- blur() - Elimina el foco de la ventana
- clearInterval(id) – Elimina el intervalo id establecido por setInterval()
- clearTimeout(name) - Elimina el intervalo name establecido por setTimeout()
- close() - Cierra la ventana actual

# Objeto window – Métodos

- confirm(mensaje) – Despliega un cuadro de diálogo con el mensaje y los botones 'Aceptar' y 'Cancelar'
- focus() - Establece el foco a la ventana actual
- moveBy(x, y) – Mueve la ventana el numero de pixeles especificados
- moveTo(x, y) – Mueve la ventana a las coordenadas especificadas
- open(URL, nombre, opciones) – Abre una ventana

# Objeto window – Métodos

- prompt(mensaje) – Muestra una caja de usuario que sugiere al usuario ingresar datos

- setInterval(expresion, tiempo) - Evalua la expresión especificada después de que hayan pasado el número de milisegundos especificados en tiempo. Devuelve un valor que puede ser usado como identificativo por clearInterval()

# Objeto window – Métodos

- setTimeout(expresion, tiempo) - Evalua la expresión especificada después de que hayan pasado el número de milisegundos especificados en tiempo. Devuelve un valor que puede ser usado como identificativo por clearTimeout()

# Abriendo Ventanas

- URL – Opcional. URL de la página a abrir. Si no se especifíca se abre una about:blank
- nombre – Opcional. Especifica el atributo target o el nombre de la ventana. Soporta:
  - _blank
  - _parent
  - _self
  - _top
  - nombre

# Abriendo Ventanas

- opciones – Opcional. Lista de elementos separados por coma
  - directories = yes|no|1|0
  - fullscreen = yes|no|1|0
  - height = pixeles
  - left = pixeles
  - location = yes|no|1|0
  - menubar = yes|no|1|0
  - resizable = yes|no|1|0
  - scrollbars = yes|no|1|0
  - status = yes|no|1|0
  - titlebar = yes|no|1|0
  - toolbar = yes|no|1|0
  - top = pixeles
  - width = pixeles

# Abriendo Ventanas

```
ventana = window.open("", "", "width=800,
 height=300, location=no, directories=no,
 menubar=no, resizable=no");
```

# Gracias por su atención.

Luis Felipe Gil.

gil.luisfelipe@gmail.com

@lamaslg