

Teoria-JavaScript-intermedio.pdf



ShadowedWarrior



Lenguajes de marcas y sistemas de gestión de información



1º Desarrollo de Aplicaciones Multiplataforma



Estudios España



Accede al documento original

antes



**Descarga sin publi
con 1 coin**



Después

WUOLAH





Teoría JavaScript Intermedio

En este documento se verán estos aspectos en JavaScript:

- Objetos
- Arrays y bucles
- DOM
- Eventos
- Expresiones regulares

Objetos

Es un grupo de propiedades / atributos, que pueden ser distintos, de una clase

Objeto persona con atributos:

- nombre: 'Jhon'
- edad: 30
- hablar: function() {...}

Creamos un objeto, de 3 formas:

1era forma

```
var persona1 = {  
  nombre: 'Jhon',  
  edad: 30,  
  saludar: function() {  
    console.log('Hola ' + this.nombre);  
  }  
}
```

2a forma

```
function Persona(nombre, edad) {  
  this.nombre = nombre;  
  this.edad = edad;  
  this.saludar = function() {  
    console.log('Hola ' + this.nombre);  
  }  
}  
  
var persona2 = new Persona('Jhon', 30);
```



Disney
**EL
REY
LEÓN**

TEATRO LOPE
DE VEGA,
MADRID

© Disney

PRODUCIDO POR

Stage
ENTERTAINMENT
SPAIN

WUOLAH

3a forma

```
var persona3 = new Object({
  nombre: 'Jhon',
  edad: 30,
  saludar: function() {
    console.log('Hola ' + this.nombre); }
});
```

Objeto Date

```
var date = new Date();
console.log(date);
```

Esto vuelca la fecha actual en una variable y la mostramos

A raíz de este objeto, podemos sacar cosas concretas con métodos: año, mes, día..

```
var year = date.getFullYear();
var month = date.getMonth() + 1;
/* +1 porque Enero empieza en 0, Febrero en 1, etc. */
var day = date.getDay();
var time = date.getTime();

console.log('Hola, hoy es ' + day + ' del mes ' + month + ' del año ' +
year + ', y la hora es ' + time);
```

Objeto Math

Es una clase con objetos matemáticos: pi, funciones, etc.

Ejemplo de creación de pi:

```
var pi = Math.PI;

console.log(pi);
```

Podemos sacar el número mínimo o máximo de un listado:

```
console.log(Math.min(1,2,9,3,4,5,-3));
console.log(Math.max(1,2,9,3,4,5,-3));
```

Podemos redondear números (round arriba, floor abajo, ceil arriba siempre):

```
console.log(Math.round(4.5));
console.log(Math.floor(4.5));
console.log(Math.ceil(4.5));
```

Podemos generar **números aleatorios** (funciona igual que en **java**, entre **0 y 1**):

```
var aleatorio = Math.random();
```

Podemos crear una función que, pasado un parámetro, devuelva un número redondeado entre 0 y el número introducido y después lo muestre por pantalla:

```
function numeroAleatorio(num) {  
    return Math.round(Math.random() * (num));  
}  
  
var miNumero = numeroAleatorio(10);  
console.log(miNumero);
```

Objeto Array

Básicamente arrays como en Java, y tenemos varias formas para crearlos.

1a forma

```
var person1 = ['Jhon', 30, true];
```

2a forma

```
var person2 = new Array(3);  
person2[0] = 'Juan';  
person2[1] = 25;  
person2[2] = false;
```

3a forma

```
var person3 = new Array('Alba', 18, true);  
var colores = new Array("verde", "rojo", "azul", "amarillo", "blanco");
```

También podemos introducir datos adicionales en el array, con el método **push**

```
colores.push("negro");
```

Ahora, colores tendría un número adicional en el **índice** con el **valor "negro"**



Organiza tu futuro: estudia hoy para destacar mañana.

En Carpe Diem te esperan cursos adaptados a ti.
Una forma fácil y real de avanzar profesionalmente.

¡Quiero formarme!



✓
Acreditados

💰
Económicos

🏠
Desde casa

🎯
Elige tu
curso

Bucles

Vamos a recorrer el array de colores creado en el apartado anterior
La sintaxis de los bucles es muy parecida a la de java, siendo esta:

```
for (var i = 0; i < colores.length; i+=1) {  
    console.log("A mí me gusta el color " + colores[i])  
}
```

Práctica Arrays

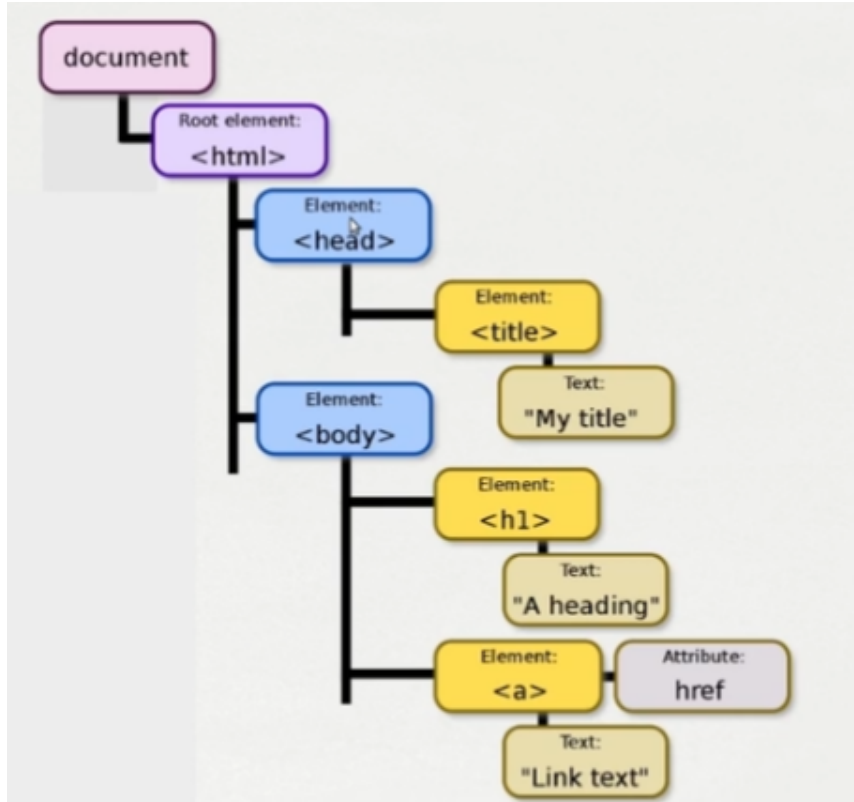
```
/* 1. Mostrar números pares del 0 - 12 */  
/* 2. Sumar números del 0 - 12 */  
var array1 = new Array(13);  
var suma = 0;  
for (var i = 0; i < array1.length; i+=1) {  
    array1[i] = (i);  
    if ((i) % 2 == 0) {  
        console.log((i));  
    }  
    suma = suma + array1[i];  
}  
console.log("El total es " + suma);
```

WUOLAH

DOM - Document Object Model

Nodos de una página **HTML**, a los que accedemos a través de **selectores**:

Cada nodo es un objeto con ciertos atributos y funciones



Selectores

Podemos asignar a una variable en JavaScript un elemento del DOM

```
var parrafo = document.getElementById('parrafo');
```

Igualmente podemos acceder únicamente a su contenido con el método `innerText`

```
console.log(parrafo.innerText);
```

Podemos elegir un elemento según su etiqueta, que nos devuelve un Array

Si el elemento no tiene un id, puedo asignárselo en el atributo **.id**

```
var parrafo = document.getElementsByTagName("p");
```

```
parrafo[0].id = 'prueba_id';
```

Podemos modificar el documento con el atributo **.style**

```
parrafo[0].style.color = 'red';
```

```
parrafo[0].style.backgroundColor = 'blue';
```


Organiza tu futuro: estudia hoy para destacar mañana.



En **Carpe Diem** te esperan cursos adaptados a ti.
Una forma fácil y real de **avanzar profesionalmente**.

✓ **Acreditados**

💰 **Económicos**

🏠 **Desde casa**

🎯 **Elige tu curso**

¡Quiero formarme!



Teléfono: 956 80 33 46
formacioncarpediem.com



Window y Document

Document se refiere a todo el documento en sí, y podemos acceder a una parte u otra dependiendo del selector que utilicemos

Window se refiere al documento y a la ventana en sí del navegador. Sus métodos pueden acceder tanto a la página como a lo que la rodea, como el objeto **window.navigator**, que devuelve información acerca del navegador

Por último, **window.location** devuelve información acerca del documento, de dónde se encuentra alojado, etc. y **window.history** devuelve información de las últimas webs que hemos visitado, con métodos como **back()**, que nos lleva a la última web

Insertar y eliminar nodos

Para insertar nodos usamos el método **.createElement('elemento')** y luego le introducimos contenido con **.textContent**. Todo dentro de una variable
Para introducirlo en el HTML, usamos el método **.appendChild(variable)**
También podemos asignarle un id en cada iteración

```
var todoList = document.querySelector('#to_do_list');
var tareas = ["Estudiar S.I.", "Leer Libro de Napoleon Hill",
"Revisar web L.M.", "Tik Tok"];

for (var i = 0; i < tareas.length; i+=1) {
    var tarea = tareas[i];
    var li = document.createElement('li');
    li.id = 'elem' + i;
    li.textContent = tarea;
    todoList.appendChild(li);
}
```

Podemos eliminar un nodo concreto de esta manera:

Seleccionamos por Id y eliminamos el elemento de la lista

```
var elementoEliminado = document.getElementById('elem2');
todoList.removeChild(elementoEliminado);
```

Podemos hacer que tarde 2 segundos en eliminarse gracias a **setTimeout**

```
setTimeout(function() {
    var elementoEliminado = document.getElementById('elem2');
    todoList.removeChild(elementoEliminado);
}, 2000)
```


NO TODO LO BUENO PASA EL FINDE.
ENTRE SEMANA, PRECIOS DE OTRO REINO.



Eventos

Un evento es algo que ocurre en el sistema y que le informa para que responda de una manera en específico si así lo indicamos.

Ejemplos de eventos:

- **keyDown:** Tecla hundida
- **keyUp:** Dejar de presionar la tecla
- **onclick:** Al hacer click
- **onmousedown:** Pulsar un botón del ratón sobre un elemento
- **onmouseenter:** Puntero entra en el área de un elemento
- **onmousemove:** Puntero se mueve sobre el área de un elemento
- **DOMContentLoaded:** Cuando la página ha cargado
- **Eventos propios**

Para usar eventos, utilizamos:

document.addEventListener("evento", function(evento) {...});

Ejemplo de código usando eventos y botones

```
<body>
  <h1>Eventos</h1>
  <button id="cambiarFondo">Presióname! Cambio el fondo</button>
  <script type="text/javascript">
    var boton = document.getElementById('cambiarFondo');
    function random(numeroMaximo) {
      return Math.round(Math.random() * numeroMaximo);
    }
    boton.addEventListener("click", function(evento) {
      var colorRandom = [
        random(255),
        random(255),
        random(255),
      ];
      document.body.style.backgroundColor =
"rgb("+colorRandom[0]+","+colorRandom[1]+","+colorRandom[2]+") ";
    })
  </script>
</body>
```

*Nota: los eventos en **addEventListener** se ponen sin el "on" delante*



Disney
**EL
REY
LEÓN**
TEATRO LOPE
DE VEGA,
MADRID

© Disney

PRODUCIDO POR
Stage
ENTERTAINMENT
SPAIN

WUOLAH

Ejemplo de evento propio

```
var event = new CustomEvent('cambiaFondo', {'detail': 'Hola Jhon'});
document.dispatchEvent(event); /* Ejecuta el código al cargar body */
document.addEventListener('cambiaFondo', function(evento) {
    console.log(evento.detail); /* Muestra el contenido del evento */
})
```

Ejemplo de To Do List

```
<h1>To Do List</h1>
  <input type="text" id="todoInput">
  <button id="addTask">Agregar tarea</button>
  <ul id="to_do_list">

  </ul>
</script>
var todolist = document.querySelector('#to_do_list');
var tareas = [];
var boton = document.getElementById('addTask');
    boton.addEventListener('click', function(evento) {
        var input = document.getElementById('todoInput').value;
        var li = document.createElement('li');
        li.textContent = input;
        todolist.appendChild(li);
    })
</script>
```

Expresiones regulares

Qué son

- Un objeto más
- Describe patrones en cadenas de texto
- Está en la mayoría de lenguajes de programación
- Simplifica las tareas de procesamiento de cadenas

Uso

- Buscar, leer y reemplazar cadenas de texto

Estructura

/expresión-regular/[parámetros]

Ejemplo

/(aaa)/gi →

g → **expresión global** (todas las cadenas)

i → **no diferencia** de mayúsculas y minúsculas

m → **multi línea**

Métodos

search → **regexObj.search(cadena)** →

Permite saber si dicho patrón está o no en un String.

Devuelve el índice de la primera coincidencia y si no hay devuelve -1

exec → **regexObj.exec(cadena)** →

Realiza una búsqueda del patrón en un String.

Devuelve un array o un null.

text → **regexObj.test(cadena)**

Realiza una búsqueda de una coincidencia entre el patrón y el String.

Devuelve un valor booleano.

match → **regexObj.match(cadena)**

Obtiene todas las coincidencias del patrón en un String

Reglas

Punto “.”

- Representa cualquier carácter

Contra barra “\”

- Se utiliza para que el siguiente carácter coja un significado o deje de tenerlo (como por ejemplo “\n” en Java, que es un salto de línea) (“\.” provocaría que el punto dejara de tener ese significado y fuera literal)

Caracteres especiales:

- | | |
|---|---|
| • \t: Representa un tabulador. | • \d: representa un dígito del 0 al 9. |
| • \r: Representa el "retorno de carro". | • \w: representa cualquier carácter alfanumérico (incluidos los guiones bajos). |
| • \n: Representa la "nueva línea". | • \s: representa un espacio en blanco |

WUOLAH

