

# Laboratorio #3: JavaScript y JSON

## 1. Objetivos

Reforzar el conocimiento de los estudiantes en el uso del lenguaje de programación JavaScript y formato de texto ligero para el intercambio de datos JSON.

## 2. Estructura

Este laboratorio está dividido en tres partes: la primera parte consistirá en una descripción teórica del lenguaje de programación JavaScript y algunas de sus funcionalidades, la segunda parte consistirá en una actividad ejemplo práctica para los estudiantes, y una última parte que consta de una actividad de evaluación que el estudiante debe realizar.

## 3. Antecedentes de JavaScript

Diseñado y especificado como ECMA-262[1], ECMAScript o JavaScript, es un lenguaje de programación de scripting interpretado, basado en objetos, débilmente tipado que puede realizar cómputo y manipular objetos computacionales en un ambiente anfitrión.

JavaScript es el lenguaje de mayor uso en el Web, y provee mecanismos para dinamizar documentos HTML en navegadores y ejecutar cómputo del lado servidor como parte de una arquitectura cliente/servidor HTTP. Así como el lenguaje de marcado HTML representa la estructura y el lenguaje CSS la presentación, JavaScript representa comportamiento en el documento HTML.

El lenguaje JavaScript puede funcionar tanto como un lenguaje procedural como un lenguaje orientado a objeto. Los objetos son creados programáticamente en JavaScript, al añadir métodos y atributos a objetos vacíos en tiempo de ejecución, a diferencia de las definiciones de clase sintácticas comunes en lenguajes compilados como C++ y Java. Una vez un objeto ha sido construido puede ser usado como un prototipo para crear objetos similares.

El estándar ECMA establece que ECMAScript es un lenguaje de programación orientado a objeto para realizar computación y manipular objetos computacionales en un ambiente anfitrión. ECMAScript no procura ser computacionalmente auto suficiente, sin embargo, se espera que el ambiente de ejecución de un programa ECMAScript provea los objetos y otras facilidades específicas al ambiente.

[1] http://www.ecma-international.org/publications/standards/Ecma-262.htm

ECMAScript fue diseñado originalmente como lenguaje de scripting en el ambiente Web, sin embargo, ECMAScript puede proveer capacidades de scripting para una variedad de ambientes anfitriones. JavaScript tiene una característica particular respecto a los lenguajes clásicos y es que no provee una forma de realizar entrada/salida de datos. Esto es debido a que el estándar ECMAScript plantea que el entorno en el cual se ejecute, debe proveer los mecanismos de entrada/salida.

Cada navegador tiene su intérprete ECMAScript/JavaScript, por lo que puede haber diferencias de desempeño de JavaScript en diferentes navegadores. El ambiente anfitrión que proveen los navegadores web incluyen, entre otros, objetos que representan ventanas, menús, pop-ups, cajas de diálogo, áreas de texto, enlaces, frames, historial, cookies y entrada/salida.

El navegador web como ambiente anfitrión también proveer mecanismos para relacionar el lenguaje de scripting con eventos del DOM como cambio de foco, carga de documentos e imágenes, selecciones, envío de formularios y acciones del ratón y/o teclado.

El desarrollador del lado del cliente debe tener en cuenta que el usuario puede deshabilitar el intérprete JavaScript en la configuración del navegador, lo cual afectará el comportamiento del documento HTML.

Entre las características más relevantes de Javascript se encuentran:

1. **Imperativo y estructurado**: soporta muchas de las características de los lenguajes estructurados como C, con la excepción del ámbito de las variables, dado que JavaScript utiliza ámbito a nivel de funciones.

2. **Dinámico:**
    a. **Tipado dinámico:** como en la mayoría de los lenguajes de scripting el tipo está asociado al valor en lugar de la variable, lo cual hace sumamente versátil el manejo de los elementos.
    b. **Orientado a objetos:** está basado en objetos mediante el esquema de prototipos. Los mismos son arreglos asociativos con la posibilidad de escribirse con el estándar común de los lenguajes orientados a objetos clásicos (`x.attr`) bajo la forma de azúcar sintáctica. JavaScript tiene definidos un cierto grupo de objetos como `function` o `date`.
    c. **Evaluación en tiempo de ejecución:** utiliza la función `eval` para evaluar oraciones provistas en strings como parámetros.

3. **Funcional / Funciones de primera clase**: las funciones en JavaScript son de primera clase, es decir, son objetos en sí mismos. Por tanto, ellas mismas tienen métodos. Una

función anidada es una definida dentro de otra función y es creada cada vez que la función es invocada. Además, cada función creada forma una clausura léxica: el ámbito léxico de la función externa, incluidas cualquier constantes, variables locales y argumentos, se vuelven parte del estado interno de cada función interna.

4. **Basado en prototipos**: JavaScript usa prototipos en vez de clases para el uso de herencia. Es posible llegar a emular muchas de las características que proporcionan las clases en lenguajes orientados a objetos tradicionales por medio de prototipos.
    a. **Funciones como constructores de objetos:** las funciones también se comportan como constructores. Prefijar una llamada a la función con la palabra clave `new` crea una nueva instancia de un prototipo, que heredan propiedades y métodos del constructor.

5. **Funciones variádicas**: un número indefinido de parámetros pueden ser pasados a la función. La función puede acceder a ellos a través de los parámetros o también a través del objeto *local arguments*.

6. **Expresiones regulares**: JavaScript también soporta expresiones regulares, que proporcionan una sintaxis concisa y poderosa para la manipulación de texto que es más sofisticado que las funciones incorporadas a los objetos de tipo string.

# 4. Sintáxis Básica de JavaScript

1. **Declaración de variables**

```
var x; //Definición sin valor
var y = 2; //Definición y asignación de valor 2
```

2. **Declaración de funciones (generales)**

```
function sumar(a,b){
      return a+b;
}
```

3. **Función recursiva**

```
function factorial(n){
      if ( n === 0 ){
            return 1;
      }
      return n*factorial(n-1);
}
```

4. **Función anónima**

```
var sumar = function (a,b){
      return a+b;
}
```

5. **Objeto (forma literal)**

```
var persona = {
      nombre: "Brendan Eich",
      cedula: 111111,
      edad: 52,
      envejecer: function() { this.edad = this.edad + 1; }
};
```

6. **Objeto (forma dinámica)**

```
var persona = {};
persona.nombre: "Brendan Eich";
persona.cedula: 111111;
persona.edad: 52;
persona.envejecer = function() { this.edad = this.edad + 1; };
```

7. **Herencia (prototipo)**

```
var persona = {
      nombre: "Brendan Eich",
      cedula: 111111,
      edad: 52,
      envejecer: function() { this.edad = this.edad + 1; }
};
var estudiante = Object.create(persona);
estudiante.carnet = 123456;
```

# 5. JavaScript y el DOM HTML

JavaScript, en el anfitrión navegador web contemporáneo, tiende a tener una fuerte interacción con la API DOM HTML, lo que permite acceder y manipular los documentos HTML y, por consiguiente, su comportamiento. Algunos métodos que se definen son:

```
document.getElementById(“nombre_id”); //Obtiene el elemento con el nombre de id especificado.

document.getElementsByClassName(“nombre_clase”); //Obtiene los elementos que tengan en el atributo class el nombre de clase indicado y retorna un arreglo con los elementos especificados.

document.getElementsByName(“nombre”); //Obtiene los elementos con el valor nombre indicado en el atributo name y retorna un arreglo con los elementos especificados.

document.getElementsByTagName("nombre_etiqueta"); //Obtiene los elementos del tipo indicado y los retorna en un arreglo.

element.innerHTML = "<html_en_string>"; //Permite colocar html o texto como hijo del elemento.

element.style.styleProperty = "<valor equivalente del CSS>"; //Permite asignar a un elemento cierto valor de una propiedad CSS.

element.attribute = "valor"; //Permite cambiar o asignar el valor del atributo de un elemento.
```

# 6. JavaScript y los Eventos del DOM HTML

El lenguaje JavaScript se encarga del comportamiento de los documentos HTML, y en la API DOM se maneja el concepto de eventos. El desarrollador tiene a su disposición JavaScript para escuchar eventos del DOM HTML y generar una respuesta que puede involucrar la modificación del DOM.

Los eventos en el árbol DOM involucran, en su gran mayoría, una acción por parte del usuario, y comprenderlos, y conocerlos queda de parte de la comprensión del desarrollador. En DOM

HTML, actualmente existen tres niveles de manejo y asignación de los eventos descritos en los siguientes ejemplos:

1. **Nivel 0**

```
<elemento onEvent = "código Javascript">
```

2. **Nivel 1**

```
element.onEvent = <objeto_función>;
```

3. **Nivel 2**

```
element.addEventListener("event", <objeto_función>);
```

## 7. Inclusión de JavaScript en un documento HTML

Para agregar código JavaScript en un documento HTML se proveen tres diferentes formas:

1. **Archivo Externo**

```
<script type = "text/javascript" src = "ruta/al/archivo.js" />
```

2. **Código explícito dentro del HTML**

```
<script type = "text/javascript" >
      //código JavaScript
</script>
```

3. **Código en un Event Handler de Nivel 0**

```
<element onEvent= "código JavaScript">
```

## 8. Antecedentes de JSON

JSON es un formato ligero para el intercambio de datos que está basado en un subconjunto de la notación literal de objetos JavaScript (estándar ECMA-262). Tiene forma de texto plano, es de simple escritura, lectura y generación. Los archivos de JSON usan la extensión .json y son del tipo MIME "application/json". Este formato es usado principalmente para serializar objetos JavaScript y transferir datos entre los servidores y las aplicaciones web. Para más información sobre el formato consulte (AQUÍ).

# 9. Tipos de datos en JSON

JSON está constituido por dos estructuras:

- Una **colección de pares nombre/valor**: en varios lenguajes esto se conoce como objeto, registro, estructura, diccionario, tabla hash, lista de claves o arreglo asociativo
- Una l**ista ordenada de valores**: en la mayoría de lenguajes esto se implementa como vectores, arreglos, listas o secuencias

Los tipos de datos disponibles con JSON son:

- **Números**: Se permiten números negativos y opcionalmente pueden contener parte fraccional separada por puntos. Ejemplo: 123.456
- **Cadenas**: Representan secuencias de cero o más caracteres. Se ponen entre doble comilla y se permiten cadenas de escape. Ejemplo: "Hola"
- **Booleanos**: Representan valores booleanos y pueden tener dos valores: true y false
- **null**: Representan el valor nulo.
- **Array**: Representa una lista ordenada de cero o más valores los cuales pueden ser de cualquier tipo. Los valores se separan por comas y el vector se mete entre corchetes. Ejemplo: `["juan","pedro","jacinto"]`
- **Objetos**: Son colecciones no ordenadas de pares de la forma <nombre>:<valor> separados por comas y puestas entre llaves. El nombre tiene que ser una cadena y entre ellas. El valor puede ser de cualquier tipo. Ejemplo:

```
{      "departamento":8,
       "nombredepto":"Ventas",
       "director": "juan rodriguez",
       "Empleados":[
              {"nombre":"Pedro","apellido":"Fernandez"},
              {"nombre":"Jacinto","apellido":"Benavente"} ]
}
```

# 10. Utilizar JSON

Con el objeto JSON nativo de JavaScript podemeos manipular estos objetos, a través de sus métodos para la creación y manipulación de objetos JSON. El objeto nativo incluye dos métodos principales:

- JSON.parse(): analiza una cadena JSON, y construye un objeto de JavaScript
- JSON.stringify(): acepta un objeto de JavaScript y devuelve su equivalente JSON

Por ejemplo:

```
var objetoJson = JSON.parse('{"clave":"valor"}');

var objetoJavascript = {name : "Robert", lastName : "Nyman”}
JSONString = JSON.stringify(objetoJavascript);
```

# 11. Actividad de Ejemplo

A continuación, deberá descargar la actividad de ejemplo llamada “Lab 3”, desde el Portal de Asignaturas ubicado en la sección “Otros”. Analice y comente con su preparador el código descargado como ejemplo práctico de JavaScript y JSON.

# 12. Evaluación

Como actividad de evaluación, se requiere que el estudiante realice un documento HTML, el cual debe poseer una imagen y la definición de los *event listeners* necesarios, para que cuando el ratón se ubique sobre la imagen, ésta cambie por otra segunda imagen, y cuando el ratón cambie de posición fuera de la imagen, la imagen regrese a ser la primera mostrada.

***“No hay secretos para el éxito. Éste se alcanza preparándose, trabajando arduamente y aprendiendo del fracaso.” - Colin Powello.***