Laboratorio #3

JavaScript y JSON

1. 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 la 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 el comportamiento en el documento HTML.

El lenguaje JavaScript puede funcionar tanto como un lenguaje procedural como un lenguaje orientado a objetos. 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 objetos 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.

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 provee 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.

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

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. **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.

3. **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.

4. **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.

5. **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.

6. **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.

7. **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*.

8. **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.

Sintaxis Básica de JavaScript

Declaración de variables:

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

Declaración de funciones (generales):

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

Función recursiva :

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

Función anónima:

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

Objeto (forma literal):

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

Objeto (forma dinámica):

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

Herencia (prototipo):

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

JavaScript y HTML

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

Archivo Externo :

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

Código explícito dentro del HTML:

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

Código en un Event Handler:

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

2. JavaScript y el DOM

JavaScript tiende a tener una fuerte interacción con la API DOM, 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.
```

JavaScript y los eventos del DOM

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:

Opción 1

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

Opción 2

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

Opción 3

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

3. 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: https://es.wikipedia.org/wiki/JSON.

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 lista 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.
- **Arreglos**: 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**: 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"} ]
}
```

Utilizar JSON

Con el objeto JSON nativo de JavaScript podemos 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:

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

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

4. Ejercicios

a) Descargar la actividad de ejemplo llamada “Lab 3”. Analice y comente con su preparador el código descargado como ejemplo práctico de JavaScript y JSON.

b) Tomar las páginas web que se vienen trabajando en los retos y agregar dinamismo usando JavaScript y JSON, de manera de separar la estructura (HTML), de los datos y agregarle algo de comportamiento con un buscador. Para eso:

1) Eliminar de tu proyecto todos los directorios dummy. Hacer commit con el mensaje “sin dummies”
2) Bajarse el archivo reto3.zip del aula virtual y descomprimir este archivo en la raíz de tu proyecto. Luego de descomprimir vas a ver que hay otros directorios:
3) Varios directorios con nombres de CI que tienen fotos e información de varios estudiantes.
4) Un directorio “conf” con 3 archivos JSON llamados configXX.json. Estos archivos tienen los textos del HTML que tienen que ver con la interfaz del usuario y podrían ser traducidos a otro idioma en caso de usar el mismo HTML para otro país. A estos datos los llamaremos datos de configuración. Los datos de configuración son textos que se muestran en la pantalla pero que no pertenecen al perfil del estudiante. Un ejemplo sería: “Mi color favorito es:” .
5) Un directorio “datos” con un archivo JSON que tiene la lista de todos los estudiantes.
6) Entrar al directorio con tu CI y sacar de la paginas perfil.html los datos y crear un archivo JSON con tu información. Este archivo debe alojarse en el directorio con tu CI y debe llamarse perfil.json y debe tener la misma estructura que en los otros directorios, pero con la información de tu perfil. En ese archivo encontrarás campos

para: nombre, descripción, color, libro, música, video_juego, lenguajes, email. Además, campos nuevos: CI, género (femenino, masculino), fecha_nacimiento.

7) Entrar al directorio “datos” y agregar al principio del archivo “index.json” la información de acceso a tu perfil. Hacer commit con el mensaje “mis datos de perfil”

8) Mueve tu archivo “perfil.html” del directorio con tu CI al directorio raíz, y elimina todos tus datos. Elimina todos los datos del archivo “index.html” también.

9) Editar el index.html y agregar un enlace que llame al JSON "conf/configES.json" en la etiqueta <head>.

10) Usar JavaScript para manipular los datos que están en los archivos JSON y agregar de forma automática la información de configuración en el HTML. El JavaScript debe quedar en el archivo externo "js/index.js", agregue en la etiqueta <head> el enlace a este archivo y asegúrese que el método de inicio sea luego que cargue todo el HTML. Al tener todo listo, haga commit con el mensaje “configuración desde javascript”

11) Agregue código JavaScript para que se muestre la lista de estudiantes según el JSON "datos/index.json", en vez de los dummies. La idea es que el index.html se vea como antes pero usando datos obtenidos de los archivos JSON. Al tener todo listo, haga commit con el mensaje “listado de estudiantes dinámico usando javascript”.

12) Observe que los archivos JSON entregados empiezan con algo de código JavaScript: el nombre de una variable, asignación y llave ("const config = {" o "const perfil = {" o "const perfiles ="). Como JavaScript NO permite abrir archivos directos desde la computadora personal por problemas de seguridad, y sólo se puede hacer importando desde un módulo (ES6+) o con llamadas en segundo plano desde un servidor web, tenemos que hacer un pequeño truco para que este reto funcione en cualquier navegador. El código JavaScript incluído en los JSON es una asignación del objeto JSON a una variable de JavaScript, NO es parte de un archivo JSON como tal, pero se necesita para que funcione este ejercicio.

13) Edite el archivo perfil.html para que muestre la información de un perfil indicado desde un atributo del URL. Al tener todo listo, haga commit con el mensaje “perfil con información dinámica”

14) Permitir cambiar de lenguaje de la interfaz mediante un atributo del URL tanto en index.html como en perfil.html usando JavaScript. Al tener todo listo, haga commit con el mensaje “cambio de lenguaje según atributo”

15) Agregue código JavaScript para hacer búsquedas sobre el listado de estudiantes. Lograr que cuando se escriba algo en el campo de texto del formulario de búsqueda vaya filtrando los alumnos del listado con aquellos estudiantes que tengan ese texto en su nombre. Si no consigue nada debe aparecer un mensaje "No hay alumnos que tengan en su nombre: [query]", donde [query] es el texto que se busca. Tomar en

cuenta que este texto, al ser de interfaz y sujeto a traducción, debe ser guardado en "conf/configXX.json". Si no hay datos, la página se debe ver como se muestra en la imagen de abajo. Al tener todo listo, haga commit con el mensaje “búsqueda de estudiantes”

Al terminar, colocar en el reto del Aula Virtual el url del repositorio de GitHub que uso para hacer este ejercicio b y una imagen del historial de Git Graph que muestra su Visual Code.