**UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA**

**‘Investigación JSON, ECMAScript 6’**

**EXPERIENCIA DE USUARIO**

**Presentado a:**

**Ing. Elmer Danuary Padilla Sorto**

**Presentado Por:**

**Jasser Abdel Ramos Betancourt 11741013**

**Tegucigalpa, M. D. C., 28 de julio del 2020**

**JavaScript Object Notation (JSON)**

JSON, cuyo nombre corresponde a las siglas JavaScript Object Notation o Notación de Objetos de JavaScript, es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas.

JSON es un formato de texto completamente independiente de lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores, entre ellos:

* C
* C++
* C#
* Java
* JavaScript
* Perl
* Python

Dichas propiedades hacen de JSON un formato de intercambio de datos ideal para usar con API REST o AJAX. A menudo se usa en lugar de XML, debido a su estructura ligera y compacta.

**¿Cómo surgió JSON?**

A principios de la década de los 90 surgió el problema de que las máquinas pudieran entenderse entre sí. Entonces utilizaban diferentes sistemas operativos y sus programas estaban escritos en diferentes lenguajes de programación. Una de las soluciones fue crear el estándar XML.

Sin embargo, XML presentaba problemas sobre todo cuando se trataba de trabajar con gran volumen de datos, puesto que el procesamiento se volvía lento. Surgieron entonces intentos para definir formatos que fueran más ligeros y rápidos para el intercambio de información. Uno de ellos fue JSON, promovido y popularizado a principios de los 2000 por Douglas Crockford, un programador conocido como el ‘gurú’ de JavaScript.

Desde entonces JSON se caracteriza por reducir el tamaño de los archivos y el volumen de datos que es necesario transmitir. Por ello fue adquiriendo popularidad hasta convertirse en un estándar.

**Características de JSON:**

* JSON es solo un formato de datos.
* Requiere usar comillas dobles para las cadenas y los nombres de propiedades. Las comillas simples no son válidas.
* Una coma o dos puntos mal ubicados pueden producir que un archivo JSON no funcione.
* Puede tomar la forma de cualquier tipo de datos que sea válido para ser incluido en un JSON, no solo arreglos u objetos. Así, por ejemplo, una cadena o un número único podrían ser objetos JSON válidos.
* A diferencia del código JavaScript, en el que las propiedades del objeto pueden no estar entre comillas, en JSON solo las cadenas entre comillas pueden ser utilizadas como propiedades.

**Ventajas y desventajas de JSON**

**JSON**

**Ventajas**

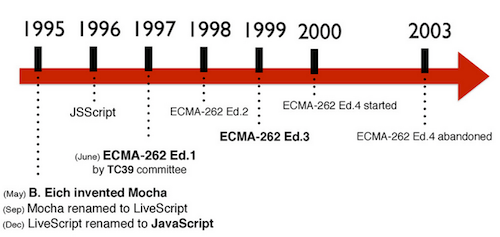
* Es auto descriptivo y fácil de entender.
* Su sencillez le ha permitido posicionarse como alternativa a XML.
* Es más rápido en cualquier navegador.
* Es más fácil de leer que XML.
* Es más ligero (bytes) en las transmisiones.
* Se parsea más rápido.
* Velocidad de procesamiento alta.
* Puede ser entendido de forma nativa por los analizadores de JavaScript.

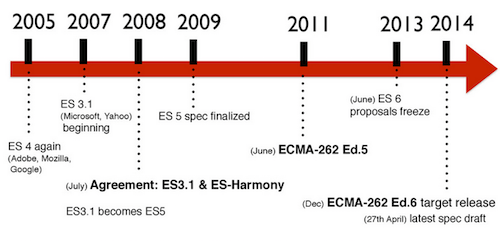
**Desventajas**

* Algunos desarrolladores encuentran su escueta notación algo confusa.
* No cuenta con una característica que posee XML: extensibilidad.
* No soporta grandes cargas, solo datos comunes.
* Para la seguridad requiere de mecanismos externos como expresiones regulares.

**ECMAScript 6**

ECMAScript v6 (Abreviado como ES6 o ES2015) es el estándar que sigue JavaScript desde Junio de 2015. Hasta ese momento la versión de JS que estábamos usando en nuestros navegadores y Node.js, era la v5.





**Principales novedades de ES6**

* **Función Arrow**

Las funciones flecha o arrow funtions son funciones anónimas con una sintaxis más compacta y que aparte de la diferencia en la sintaxis también tienen algunas peculiaridades como que no vinculan su propio this o que no se pueden usar como constructores.

* **Clases**

Ahora JavaScript tendrá clases, muy parecidas las funciones constructoras de objetos que realizábamos en el estándar anterior, pero ahora bajo el paradigma de clases, con todo lo que eso conlleva, como, por ejemplo, herencia. Aunque no deja de ser un azúcar sintáctico (Sugar Syntax) porque en JavaScript no tenemos clases, tenemos prototipos.

* **This**

La variable this muchas veces se vuelve un dolor de cabeza. antiguamente teníamos que cachearlo en otra variable ya que solo hace referencia al contexto en el que nos encontremos. Por ejemplo, en el siguiente código si no hacemos var that = this dentro de la función document.addEventListener, this haría referencia a la función que pasamos por Callback y no podríamos llamar a foo().

* **Let y Const**

Ahora podemos declarar variables con let en lugar de var si no queremos que sean accesibles más allá de un ámbito.

* **Template Strings**

Con ES6 podemos interpolar Strings de una forma más sencilla que como estábamos haciendo hasta ahora.

* **Destructuring**

Tenemos nuevas formas de asignar valores a Arrays y a Objetos.

* **Valores por defecto**

Otra novedad es asignar valores por defecto a las variables que se pasan por parámetros en las funciones. Antes teníamos que comprobar si la variable ya tenía un valor. Ahora con ES6 se la podemos asignar según creemos la función.

* **Módulos**

A esto lo llamo un browserify nativo. Ahora JavaScript se empieza a parecer a lenguajes como Python o Ruby. Llamamos a las funciones desde los propios Scripts, sin tener que importarlos en el HTML, si usamos JavaScript en el navegador.

**Diferencia entre Tecnologías del lado del Cliente y Tecnologías del lado del Server**

**En qué consiste la programación del lado del servidor**

La programación del lado del servidor entra en juego en el desarrollo de páginas web con elementos dinámicos y de aplicaciones web. Esta tecnología de desarrollo web se basa en la utilización de scripts que ejecuta el servidor web, con ayuda del lenguaje de programación adecuado, cuando un cliente solicita el contenido. Una tarea frecuente de los scripts consiste en extraer los datos que se necesitan de una base de datos y de integrarlos en el proyecto web. Aunque el usuario accede al proyecto a través de páginas HTML, el código fuente de los scripts permanece oculto.

En los primeros tiempos de la red de redes, la programación del lado servidor se reducía casi exclusivamente al desarrollo de programas en lenguajes C, Perl y de líneas de comando (CMD, PowerShell). Estas aplicaciones eran ejecutadas e interpretadas por el sistema operativo del servidor y el resultado podía ser transmitido desde el servidor web al navegador que había hecho la petición a través de una interfaz de entrada común (Common Gateway Interface, CGI). Hoy en día, muchos servidores web pueden ejecutar directamente los scripts, por ejemplo, con ayuda de módulos. El lenguaje de programación del lado servidor más utilizado en la actualidad es PHP, publicado en 1995 y muy similar a C y a Perl.

**La programación del lado cliente**

Los lenguajes de programación del lado cliente también se utilizan para realizar proyectos con contenidos dinámicos, pero, a diferencia de los lenguajes del lado servidor, no es el servidor el que ejecuta y procesa los scripts, sino el cliente solicitante. Con esta finalidad, los scripts se incluyen en el documento HTML o XHTML o se escribe en un archivo separado que se enlaza al documento principal.

Cuando un usuario solicita una página web o una aplicación web con un script de este tipo, el servidor web envía el documento HTML y el script al navegador, quien lo ejecuta y presenta el resultado final. Asimismo, los scripts del lado cliente contienen instrucciones concretas para el navegador web al respecto de cómo ha de reaccionar a ciertas acciones llevadas a cabo por el usuario como, por ejemplo, un clic en un botón específico.

**Ajax**

Es una nueva tecnología que utiliza tanto del lado del cliente scripting y secuencias de comandos del lado del servidor en conjunto. Ajax puede llamar a una base de datos sin actualizar la página. Ajax es un ejemplo de un script del lado del cliente que llama a un script del lado del servidor. La llamada es transparente para el usuario desde una nueva llamada al servidor que se realiza en la ventana actual sin actualizar la página como las típicas del lado del servidor.