Centro de Tecnología y Artes Visuales.

Resumen JavaScript

Byron Navarro Salazar

Tabla de contenido

[Definiciones: 4](#_Toc414173161)

[Palabras Reservadas. 4](file:///D:\Tabla%20de%20contenido.docx#_Toc414173162)

[Definiciones Semana 2 5](file:///D:\Tabla%20de%20contenido.docx#_Toc414173163)

[Definiciones Semana 3 6](file:///D:\Tabla%20de%20contenido.docx#_Toc414173164)

[Definiciones Semana 4 7](file:///D:\Tabla%20de%20contenido.docx#_Toc414173165)

[JSON 9](file:///D:\Tabla%20de%20contenido.docx#_Toc414173166)

[Apendice A Good Parts 11](#_Toc414173167)

[Unicode: 11](file:///D:\Tabla%20de%20contenido.docx#_Toc414173168)

[Typeof: 11](file:///D:\Tabla%20de%20contenido.docx#_Toc414173169)

[JavaScript Timing Events 12](file:///D:\Tabla%20de%20contenido.docx#_Toc414173170)

[JavaScript Errores – Throw y tratar de atrapar 12](file:///D:\Tabla%20de%20contenido.docx#_Toc414173171)

[Funciones 12](file:///D:\Tabla%20de%20contenido.docx#_Toc414173172)

[Colección 15](#_Toc414173173)

[Array.concat (material) 15](#_Toc414173174)

[Array.join (separador) 15](#_Toc414173175)

[Array.pop () 15](#_Toc414173176)

[Array.push (material ...) 15](#_Toc414173177)

[Array.reverse () 15](#_Toc414173178)

[Array.shift () 15](#_Toc414173179)

[Array.slice (inicio, fin) 16](#_Toc414173180)

[array.sort (comparefn) 16](#_Toc414173181)

[Array.splice (inicio, deleteCount, material ...) 16](#_Toc414173182)

[Array.unshift (material ...) 16](#_Toc414173183)

[Función 16](#_Toc414173184)

[Function.apply (thisArg, argArray) 16](#_Toc414173185)

[Número 16](#_Toc414173186)

[Number.toExponential (fractionDigits) 16](#_Toc414173187)

[Number.toFixed (fractionDigits) 16](#_Toc414173188)

[Number.toPrecision (precisión) 16](#_Toc414173189)

[Objeto 17](#_Toc414173190)

[Object.hasOwnProperty (nombre) 17](#_Toc414173191)

[RegExp 17](#_Toc414173192)

[regexp.exec (cadena) 17](#_Toc414173193)

[Cadena 17](#_Toc414173194)

[String.charAt (pos) 17](#_Toc414173195)

[String.charCodeAt (pos) 17](#_Toc414173196)

[String.Concat (cadena ...) 17](#_Toc414173197)

[String.indexOf (searchstring, posición) 17](#_Toc414173198)

[string.localeCompare (que) 18](#_Toc414173199)

[string.replace (searchValue, replaceValue) 18](#_Toc414173200)

[string.search (regexp) 18](#_Toc414173201)

[string.slice (inicio, fin) 18](#_Toc414173202)

[string.split (separador, límite) 18](#_Toc414173203)

[string.toLocaleLowerCase () 18](#_Toc414173204)

[string.toLocaleUpperCase () 18](#_Toc414173205)

[string.toLowerCase () 19](#_Toc414173206)

[string.toUpperCase () 19](#_Toc414173207)

[String.fromCharCode (char ...) 19](#_Toc414173208)

[Capítulo 9 19](#_Toc414173209)

[“Al escribir un estilo claro, tus programas serán fácil de leer” 19](#_Toc414173210)

[Capítulo 10 21](#_Toc414173211)

[“Sería bueno si los productos y lenguajes de programación fueran diseñados para tener solo partes buenas” 21](#_Toc414173212)

# Definiciones:

|  |  |
| --- | --- |
| **Palabra** | **Función** |
| eval | Ejecuta o evalúa una discusión (evalúa la expresión). |
| join | Une a los elementos de una matriz en una cadena, y devuelve la cadena. Se separa con coma(,). |
| toLowerCase | Convierte una cadena a minúsculas. |
| toUpperCase | Para convertir una cadena a mayúsculas. |
| Prompt | Muestra un cuadro de diálogo que solicita al visitante para la entrada. Se utiliza a menudo,  si desea que el usuario introduzca un valor antes de entrar en una página. |
| Alert | Muestra un cuadro de alerta con un mensaje especificado y un botón Aceptar.  Se utiliza a menudo si usted quiere asegurarse de que la información llega hasta el usuario. |
| Break | La sentencia break "salta" de un bucle. |

# Palabras Reservadas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| break | predeterminados | función | return | var |
| case | eliminar | if | switch | void |
| catch | do | in | this | while |
| const | else | instanceof | throw | con |
| continue | finally | let | try |  |
| debugger | for | new | typeof |  |

# Definiciones Semana 2

Motor V8:

V8 está diseñado para ser utilizado tanto en un navegador (sobre todo en Chrome y Chromium navegadores) y como un motor de alto rendimiento independiente que se puede integrar en los proyectos independientes, por ejemplo del lado del servidor JavaScript en Node.js , o cliente lado JavaScript en .NET / Mono usando V8.NET.

Scripting Language:

Es un lenguaje de programación que soporta scripts, programas escritos para un especial entorno de tiempo de ejecución que puede interpretar (en lugar de compilar ) y automatizar la ejecución de tareas que, alternativamente, podría ser ejecutado de una en una por un ser humano operador.

Onload:

El proceso de carga se produce cuando un objeto se ha cargado.

onload es la más utilizada en el elemento <body> para ejecutar un script una vez a la página web se ha cargado completamente todo el contenido (incluyendo imágenes, archivos de comandos, archivos CSS, etc.).

El proceso de carga se puede utilizar para comprobar el tipo de navegador y versión del navegador del visitante, y cargar la versión correcta de la página web en base a la información.

Document Ready:

Una página no puede ser manipulado de forma segura hasta que el documento es "listo". jQuery detecta este estado de preparación para usted. Código incluye dentro de $( document ).ready() sólo se ejecutará una vez que la página Document Object Model (DOM) está listo para el código JavaScript a ejecutar.

Onunload:

Ocurre una vez al perfil ha descargado (o la ventana del navegador se ha cerrado).

onunload se produce cuando el usuario se desplaza fuera de la página (haciendo clic en un enlace, envía un formulario, cierre la ventana del navegador, etc.).

|  |  |
| --- | --- |
| **Palabra** | **Función** |
| Typeof | Se utiliza para encontrar el tipo de una variable de JavaSccript |
| Undefined | Indica que no se le ha asignado valor a una variable |
| Null | Se supone que es algo que no existe, osea, «nulo», en JS es un objeto. |
| Confirm | Muestra un cuadro de diálogo con un mensaje determinado, junto con un bien y  un botón Cancelar. |
| Open | Abre una ventana en el navegador. |

# Definiciones Semana 3

|  |  |
| --- | --- |
| **Palabra** | **Función** |
| Delete | Retira una propiedad de un objeto. |
| String | Los índices de cadenas están basados ​​en cero: El primer carácter está en la posición 0,  el segundo en 1, y así sucesivamente. Es una cadena. |
| Boolean | Booleano JavaScript representa uno de dos valores: **verdadero** o **falso.** |
| Number | Convierte el argumento objeto a un número que representa el valor del objeto.  Si el valor no se puede convertir a un número legal, se devuelve NaN. |
| Object | Objetos son variables también. Pero los objetos pueden contener muchos valores.  Este código asigna muchos valores (Fiat, 500, blanco) a un llamado coche por ejemplo. |
| Function | Una función de JavaScript es un bloque de código diseñado para realizar una tarea en particular.  Una función de JavaScript se ejecuta cuando "algo" lo invoca (llama). |
| Date | El objeto Date se utiliza para trabajar con fechas y horas.  Fecha objetos son creados con new Date (). |
| Constructor | Metodo en el que se utiliza una función que está dentro de una clase que nos permite hacer el llamado. |

Programacion Orientada a Objetos:

Es un paradigma de programación que utiliza la abstracción para crear modelos basados ​​en el mundo real. Utiliza diversas técnicas de paradigmas previamente establecidas, incluyendo la modularidad, polimorfismo y encapsulamiento. Hoy en día, muchos lenguajes de programación soportan programación orientada a objetos (POO).

Puede considerarse como el diseño de software a través de una conjunto de objetos que cooperan, a diferencia de un punto de vista tradicional en el que un programa puede considerarse como un conjunto de funciones, o simplemente como una lista de instrucciones para la computadora.

# Definiciones Semana 4

String:

El indexOf() método devuelve el índice, dentro del objeto String que realiza la llamada, de la primera ocurrencia del valor especificado, comenzando la búsqueda desde indice Busqueda; 0 -1 si no se encuentra dicho valor.

Hoisting:

Es el comportamiento predeterminado de JavaScript de mover todas las declaraciones de la parte superior del ámbito actual (a la parte superior de la secuencia de comandos actual o la función actual).

Strict JS:

Define que el código JavaScript debe ser ejecutado en el "modo estricto".  
La directiva "use strict" es nuevo en JavaScript 1.8.5 (ECMAScript versión 5).

No es una declaración, sino una expresión literal, ignorado por las versiones anteriores de JavaScript.

El propósito de "use strict" es para indicar que el código debe ser ejecutado en el "modo estricto".

Con el modo estricto, no se puede, por ejemplo, utilizar variables no declaradas.

JavaScript Common Mistakes:

**Incomprensión Flota**

Todos los números en JavaScript se almacenan como 64-bits de números de punto flotante(flotadores).

**Esperando Libremente Comparación:**

En comparación regular, tipo de datos no importa.

**Accidentalmente Usando el operador de asignación:**

Programas JavaScript puede generar resultados inesperados si un programador utiliza accidentalmente un operador de asignación (=), en lugar de un operador de comparación (==) en una sentencia if.

Esto si instrucción devuelve falso (como se esperaba) porque x no es igual a 10:

**Reglas de objetos:**

Reglas generales para las definiciones de objeto:

* Coloque el soporte de la apertura en la misma línea que el nombre del objeto.
* Utilice dos puntos más un espacio entre cada propiedad y su valor.
* Use comillas valores de cadena, no en torno a valores numéricos.
* No añada una coma después de la última pareja propiedad-valor.
* Coloque el soporte de cierre, en una nueva línea, sin espacios iniciales.
* Siempre termine una definición de objeto con un punto y coma.

JavaScript Style Guide and Coding Conventions:

**Convenciones de codificación de JavaScript:**

Las convenciones de codificación son las directrices de estilo de programación . Por lo general se refieren a:

* Reglas de nomenclatura y de declaración de variables y funciones.
* Reglas para el uso de espacios en blanco, la sangría, y los comentarios.
* Programación de las prácticas y los principios

Las convenciones de codificación de calidad segura :

* Mejora la legibilidad del código
* Hacer el mantenimiento del código más fácil

Las convenciones de codificación se pueden documentar las reglas para los equipos a seguir, o simplemente ser su práctica de codificación individual.

**Nombres de variables:**

En W3schools utilizamos camelCase de nombres de identificadores (variables y funciones).

Todos los nombres comienzan con una letra .

**Reglas Declaración**

Reglas generales para las declaraciones simples:

* Siempre termine declaración simple con un punto y coma.

Normas generales para la (compuestos) declaraciones complejas:

* Ponga el soporte de abertura en el extremo de la primera línea.
* Utilice un espacio antes del corchete de apertura.
* Ponga el soporte de cierre en una nueva línea, sin espacios iniciales.
* No finalice declaración complejo con un punto y coma.

# JSON



|  |  |
| --- | --- |
|  | \* JSON utiliza la sintaxis de JavaScript, pero el formato JSON es sólo texto.  El texto puede ser leído y utilizado como un formato de datos por cualquier lenguaje de programación. |

Es cómo acelerar su código JavaScript.

**Reducir la actividad en Loops:**

Loops se utilizan a menudo en la programación.

Cada declaración dentro de un bucle se ejecutará para cada iteración del bucle.

Búsqueda de declaraciones o asignaciones que se pueden colocar fuera del bucle.

**Reducir DOM Acceso:**

Acceso al DOM HTML es muy lento, en comparación con otras sentencias de JavaScript.

Si usted espera para acceder a un elemento DOM varias veces, acceder a él una vez, y lo utilizan como una variable local.

JavaScript Performance:

# Apendice A Good Parts

Es una variable que es visible en todos los ámbitos.

Puede complicar el programa debido a que puede cambiar.

Hace que sea difícil de usar los subprogramas.

Tienen un montòn de idiomas.

Formas de declararlas:

Var foo = valor; 🡪 Fuera de cualquier función.

Window.foo = valor; 🡪 Para añadir una propiedad.

Foo =valor; 🡪 Una variable sin ser declarada.

Scope:

La sintaxis de JavaScript viene de C. Como C, un bloque (un conjunto de estados envueltos entre llaves) crea un ámbito. JavaScript utiliza la sintaxis del bloque, pero no proporciona ámbito de bloque: una variable declarada en un bloque es visible en todas partes en la función que contiene el bloque. Esto resulta ser una mala práctica en JavaScript, ya que no tiene ámbito de bloque.

Semicolon Insertion:

Que intenta corregir los programas defectuosos por punto y coma. Esto puede enmascarar los errores más graves.

Palabras Reservadas:

La mayoría de estas palabras no se usan en el lenguaje. No pueden ser utilizados para las variables o parámetros de nombres. Cuando las palabras reservadas se utilizan como claves en objetos literales, deben ser citados.

# Unicode:

Ha crecido para tener una capacidad de más de 1 millón de caracteres. Unicode considera el par a ser un único carácter.

# Typeof:

Devuelve una cadena que identifica el tipo de su operando.

+ Simbolo:

El operador + puede agregar o concatenar.

Floating Point:

Esta norma es muy adecuado para muchas aplicaciones, pero viola la mayoría de las cosas que ha aprendido acerca de los números en la escuela media.

NaN:

Es una cantidad especial definido por IEEE 754. Representa no es un número, a pesar de que:

typeof "número" NaN === // true

El valor puede ser producido por el intento de convertir una cadena en un número cuando la cadena no está en la forma de un número.

# JavaScript Timing Events

Ejecuta un código en intervalos de tiempo, se le llama “Eventos de Tiempo”.

Utiliza dos métodos llamados:

* SetInterval(): Ejecuta la función que se realiza una y otra vez.
* SetTimeout(): Ejecuta la función una vez y espera un número específico de milisegundos.

# JavaScript Errores – Throw y tratar de atrapar

JavaScript será normal cuando aparezca algún error, para, y genera un mensaje, osea, lanza un error.

🡪Intento: Probar un bloque de código para los errores.

🡪Captura: Maneja el error.

🡪Tiro: Crea errores personalizados.

🡪Últmo: Ejecuta código, después de tratar atraparlo.

# Funciones

Una función encierra un conjunto de sentencias. Las funciones son la unidad modular fundamental de JavaScript. Se utilizan para la reutilización de código, ocultación de información, y la composición. Las funciones se utilizan para especificar el comportamiento de los objetos.

Objetos de función

Las funciones en JavaScript son objetos. Eestán vinculados a Object.prototype. También se crea con dos propiedades adicionales ocultas: contexto de la función y el código que implementa el comportamiento de la función.

Función Literal

La función tiene cuatro partes. La primera parte es la función de palabra reservada. La segunda el nombre de la función, si una función no se le da un nombre, como se muestra en el ejemplo anterior, se dice ser anónimo. La tercera parte es el conjunto de parámetros de la función, envueltos en paréntesis. La cuarta parte es un conjunto de sentencias envueltos entre llaves. El objeto de función creada por un literal de función contiene un enlace a ese contexto exterior. Esto se denomina cierre.

Invocación

Invocar una función suspende la ejecución de la función actual, pasando de control y parámetros para la nueva función. Además de los parámetros declarados, cada función recibe dos parámetros adicionales: este y argumentos.

Alcance

Alcance en un lenguaje de programación controla la visibilidad y tiempos de vida de las variables y parámetros. Las variables definidas en un bloque pueden ser liberados cuando finalice la ejecución del bloque. Esta es una buena cosa.

Las funciones internas tienen acceso a los parámetros y variables de las funciones que están definidas dentro (con la excepción de esto y argumentos).

Las devoluciones de llamada

Supongamos que hay una secuencia que comienza con la interacción del usuario, haciendo una petición del servidor, y finalmente mostrar la respuesta del servidor.

Módulo

Podemos utilizar las funciones y cierre para hacer módulos. Un módulo es una función u objeto que presenta una interfaz pero que esconde su estado y la aplicación. Mediante el uso de funciones para producir módulos, podemos eliminar casi por completo el uso de variables globales, mitigando de esta manera una de las peores características de JavaScript.

Cascada

Es el estado de un objeto a devolver nada. Si tenemos esos métodos devuelven esto en vez de definir, podemos permitir a las cascadas. En una cascada, podemos llamar a muchos métodos en el mismo objeto en secuencia en una sola sentencia.

Memoization

Las funciones se pueden usar objetos para recordar los resultados de las operaciones anteriores, por lo que es posible evitar trabajo innecesario. Esta optimización se llama memoization.

El patrón de invocación de método

Un método puede usar esto para acceder al objeto para que pueda recuperar los valores del objeto o modificar el objeto.

El Patrón Función Invocación

Si el método define una variable y le asigna el valor de esto, la función interna tendrá acceso a este a través de esa variable.

El Patrón Constructor Invocación

Es lo peor de ambos mundos. Si una función se invoca con el nuevo prefijo, a continuación, un nuevo objeto se crea con un vínculo oculto por el importe de miembro prototipo de la función, y esto estará vinculado a ese nuevo objeto. El nuevo prefijo también cambia el comportamiento de la instrucción de retorno. Funciones que están destinados a ser utilizados con el nuevo prefijo se llaman constructores.

El Aplicar Patrón Invocación

El método de aplicación nos permite construir una serie de argumentos a utilizar para invocar una función. También nos permite elegir el valor de este. El método de aplicación tiene dos parámetros. El primero es el valor que se debe obligado a ello. El segundo es una serie de parámetros.

Argumentos

Un parámetro bono que está disponible para funciones cuando se invocan es el conjunto argumentos.. Esto hace que sea posible escribir funciones que toman un número indeterminado de parámetros.

Return

Cuando se invoca una función, se inicia la ejecución de la primera sentencia, y termina cuando llega al que cierra el cuerpo de la función. Eso hace que la función para devolver el control a la parte del programa que invoca la función.

Excepciones

Las excepciones son inusuales percances (pero no completamente inesperados) que interfieren con el flujo normal de un programa.

Aumentar Tipos

Mediante el aumento de Function.prototype, podemos hacer un método disponible para todas las funciones. Al aumentar Function.prototype con un método método, ya no tenemos que escribir el nombre de la propiedad prototipo. Esa parte de la fealdad ahora se puede esconder.

La recursividad

Es una técnica de programación de gran alcance en el que un problema se divide en un conjunto de subproblemas similares, cada uno resuelto con una solución trivial.

Capítulo 8

Methods

# Colección

## Array.concat (material)

Produce una nueva matriz que contiene una copia superficial de esta matriz con los elementos añadidos al final.

## Array.join (separador)

Hace una cadena de una matriz. Para ello, hacer una cadena de cada uno de los elementos de la matriz y, a continuación, concatenar todos juntos con un separador entre ellos.

## Array.pop ()

 Los métodos pop y empuje hacen un trabajo conjunto como una pila. El método pop elimina y devuelve el último elemento de esta matriz. Si la matriz está vacía, devuelve indefinido.

## Array.push (material ...)

Añade elementos al final de una matriz. A diferencia del método concat, que modifica la matriz y añade elementos de matriz entera.

## Array.reverse ()

El método modifica la matriz inversa invirtiendo el orden de los elementos.

## Array.shift ()

 El método de cambio elimina el primer elemento de una matriz y devuelve. Si la matriz está vacía, devuelve indefinido.

## Array.slice (inicio, fin)

 El método slice hace una copia superficial de una porción de una matriz. El primer elemento copiado será array [Inicio]. Se detendrá antes de copiar array [final]. El parámetro final es opcional y el valor predeterminado es Array.length.

## array.sort (comparefn)

 El método sort ordena el contenido de una matriz en su lugar.

## Array.splice (inicio, deleteCount, material ...)

El empalme elimina elementos de una matriz, reemplazándolos con nuevos elementos. El parámetro de inicio es el número de una posición dentro de la matriz. El parámetro deleteCount es el número de elementos a eliminar a partir de esa posición.

## Array.unshift (material ...)

El método unshift es como el método push excepto que empuja a los elementos al frente de esta matriz en lugar de al final.

# Función

## Function.apply (thisArg, argArray)

El método de aplicación invoca una función, pasando el objeto que se une a esto y un conjunto opcional de argumentos.

# Número

## Number.toExponential (fractionDigits)

El método toExponential convierte este número en una cadena en forma exponencial. El parámetro fractionDigits opcional controla el número de decimales.

## Number.toFixed (fractionDigits)

 El método toFixed convierte este número en una cadena en forma decimal. El parámetro fractionDigits opcional controla el número de decimales.

## Number.toPrecision (precisión)

 El método toPrecision convierte este número en una cadena en forma decimal.

# Objeto

## Object.hasOwnProperty (nombre)

El método hasOwnProperty devuelve true si el objeto contiene una propiedad que tiene el nombre. No se examina la cadena de prototipo.

# RegExp

## regexp.exec (cadena)

El método exec es la más potente (y más lenta) de los métodos que utilizan expresiones regulares. Si coincide con éxito la expresión regular y la cadena, devuelve una matriz. El elemento 0 de la matriz contendrá la subcadena que hacía juego con la expresión regular.

# Cadena

## String.charAt (pos)

 El método charAt devuelve el carácter en la posición pos en esta cadena. Ifpos es menor que cero o mayor que o igual a String.length, devuelve la cadena vacía. JavaScript no tiene un tipo de carácter.

## String.charCodeAt (pos)

El método charCodeAt es el mismo que charAt excepto que en lugar de devolver una cadena, se devuelve una representación entera del valor de punto de código del carácter en la posición pos de esa cadena.

## String.Concat (cadena ...)

El método concat hace una nueva cadena concatenando otras cadenas juntas.

## String.indexOf (searchstring, posición)

El método indexOf busca una searchstring dentro de una cadena. Si se encuentra, devuelve la posición del primer carácter emparejado; de lo contrario, devuelve -1.

## string.localeCompare (que)

 El método localCompare compara dos cadenas. No se especifican las reglas de cómo se comparan las cuerdas. Si esta cadena es menor que la cadena, el resultado es negativo.

## string.replace (searchValue, replaceValue)

El método replace hace una búsqueda y reemplazar operación en esta cadena, la producción de una nueva cadena.

## string.search (regexp)

El método de búsqueda es como el método indexOf, excepto que toma un objeto de expresión regular en lugar de una cadena. Devuelve la posición del primer carácter del primer partido, si lo hay, o -1 si la búsqueda falla.

## string.slice (inicio, fin)

El método slice hace una nueva cadena copiando una parte de otra cadena. Si el parámetro de inicio es negativo, se añade string.length a ella. El parámetro final es opcional, y su valor por defecto es string.length.

## string.split (separador, límite)

El método split crea una matriz de cadenas mediante el fraccionamiento de esta cadena en trozos. El parámetro de límite opcional puede limitar el número de piezas que se dividen.

## string.toLocaleLowerCase ()

El método toLocaleLowerCase produce una nueva cadena que se hizo mediante la conversión de esta cadena a minúsculas utilizando las reglas para la configuración regional. Esto es principalmente para el beneficio de turco porque en ese idioma "yo" se convierte en I, no 'i'.

## string.toLocaleUpperCase ()

 El método toLocaleUpperCase produce una nueva cadena que se hizo mediante la conversión de esta cadena a mayúsculas utilizando las reglas para la configuración regional. Esto es principalmente para el beneficio de Turquía, ya que en ese idioma 'i' se convierte en '', no 'I'.

## string.toLowerCase ()

El método toLowerCase produce una nueva cadena que se hizo mediante la conversión de esta cadena a minúsculas.

## string.toUpperCase ()

El método toUpperCase produce una nueva cadena que se hizo mediante la conversión de esta cadena a mayúsculas.

## String.fromCharCode (char ...)

La función String.fromCharCode produce una cadena a partir de una serie de números.

# Capítulo 9

### “Al escribir un estilo claro, tus programas serán fácil de leer”

Los programas son las cosas más complejas que los seres humanos hacen, están compuestos por funciones, sentencias, un número de piezas y expresiones y que deben estar libres de error.

Se espera que los softwares sean modificados en el transcurso del tiempo.

Los buenos programas deben tener una presentación clara, “Sí un programa se expresa bien, entonces tenemos mejor probabilidad de ser capaz de entenderlo, de modo que podremos modificarlo y repararlo con éxito.

Para codificar con JavaScript tenemos que tener mucha disciplina, el tiempo de compilación da aseguramiento de la calidad de nuestros programas.

JavaScript tiene características débiles o problemáticas que pueden modificar el hecho de que hagamos buenos programas, debemos evitarlo, debemos evitar también las características que son útiles pero peligrosas, así se evita gran clase de errores.

Sí un el programa es capaz de comunicar con claridad su estructura y características, es menos probable que se rompa cuando se modifica en el futuro.

Resulta que las cuestiones de estilo en la programación por la misma razón es que importe en la estructura.

Si somos capaces de leer y entender, seremos capaces de modificarlo y mejorarlo.

Siempre utiliza if y while porque son menos propensos a errores.

Yo utilicé el estilo K&R porque evita un error horrible de diseño.

Utilizar comentarios en el programa para dejar clara la programación.

JavaScript proporciona soporte para programas grandes.

Las variables globales se hacen cada vez problemáticas.

Yo uso una variable global para contener una aplicación o una biblioteca.

# Capítulo 10

### “Sería bueno si los productos y lenguajes de programación fueran diseñados para tener solo partes buenas”

Habla del programa que el quería hacer con JavaScript, pero con un subconjunto simplificado. Las características que tomó en cuenta fueron:

* Funciones como objetos de primera clase.
* Objetos dinámicos con herencia de prototipos.
* Literales de objetos y matriz.

Contenía lo mejor de las partes buenas, notó que las palabras reservadas son innecesarias. El programador elige una palabra, para más facilidad. Resulta que los diseños que solo trabajan son mucho más difíciles de producir que los diseños que se ensamblan largas listas de características.

Las características tienen un costo de documentación por eso queremos obtener las características del núcleo por es donde creamos la mayor parte del valor.

Hacemos frente a la complejidad del diseño de características impulsando por la búsqueda y la pervivencia de las buenas partes.