Informe de Laboratorio 11

Tema: JavaScript

Estudiante	Escuela	Asignatura
Luis Gustavo Sequeiros Condori lsequeiros@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web I Semestre: II

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 6 Enero 2024	Al 10 Enero 2024

1. Objetivos

- Conocer el lenguaje JavaScript

2. Temas a Tratar:

- JavaScript

3. JavaScript

3.1. ¿Qué puede hacer?

- Cambiar el contenido HTML
- Cambiar los atributos HTML
- Cambiar los estilos de los elementos HTML
- Esconder y mostrar elementos HTML

3.2. ¿Dónde está?

- Puede ser insertado en la etiqueta ¡script¿, pueden estar situados en body o head.
- Situar los scripts debajo de los elementos mejora la visualización, la interpretación del script hace más lenta la página.
- También puede estar situado en archivos externos, con`<script src="file.js"></script>`
- Se comportará de acuerdo a donde esté ubicado la etiqueta ¡script¿
- Los scripts externos no pueden contener etiquetas ¡script¿
- Con scripts externos, el programa es fácil de modificar y de leer
- El almacenamiento en caché puede acelerar la carga de la página

3.3. La salida

- Puede escribir en un elemento HTML, en la salida HTML, en una ventana de alerta o en la consola del navegador
- innerHTML: Cambia el contenido de la etiqueta. `document.getElementById("demo").innerHTML = 5 + 6;`
- document.write() puede ser utilizado con fines de testeo. Cuando se usa document.write() luego de haber cargado la página, esta borrará su contenido y se escribirá lo ingresado (sobreescribir)
- window.alert(), se utiliza para mostrar datos. Se puede skipear el objeto window, ya que tiene alcance global. Eso significa que todas las variables y métodos pertenecen al objeto window.
- console.log() para propósitos de debugeo, muestra la salida en la consola del navegador
- window.print(), imprime (abre una ventana para imprimir en impresora) el contenido de la ventana actual.

3.4. Sentencias

- Las sentencias de JavaScript son ejecutadas por el navegador
- Se componen de valores, operadores, expresiones, claves y comentarios
- Se ejecutan una a una en el orden en el que están escritos
- Al final de una sentencia se debe colocar punto y coma (semicolon), no es obligatorio pero si muy recomendable
- Una buena práctica es que la línea no tenga más de 80 caracteres, asimismo, el meor lugar para romper una línea es después de un opereador
- Las sentencias pueden ser agrupadas en bloques delimitados por llaves.
- A menudo las sentencias inician con una palabra clave para indicar la acción a realizar

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

Figura 1: Palabras clave en JS

3.5. Sintaxis

- Hay dos tipos de valores: fijos(literales) y variables
- Literales: Lo números son escritos con o sin decimales, Strings con comillas dobles o simples
- Variables: Almacenan valores. Las claves var, let y const declaran variables.
- El signo = asigna valores
- Operadores: + - /
- Expresiones: Una combinación de variables, valores y operadores, con los que se calcula un valor. El cálculo es llamado Evaluación
- Comentarios: Luego de // o entre /* */
- Identificadores: Empiezan con letras,$o _, no con números. Distinguen entre mayúsculas y minúsculas.
- Utiliza Unicode

3.6. Variables

- Se pueden declara de 4 formas: Automáticamente, var, let y const (se considera buena práctica declarar las variables antes de usarlas)
- var fue usada desde 1995 a 2015 (solo debe ser usada en código escrito para navegadores antiguos)
- let y const fueron añadidas en 2015
- una variable declarada con let puede ser cambiada, con const no.
- const si el valor no debe cambiarse, si el tipo no debe cambiarse (arreglos y objetos)
- usar let solo si no se puede usar const
- En una declaración, la variable no tiene valor (undefined)
- Es una buena práctica declarar todas las variables al inicio del código
- Se puede declarar en una línea separando por coma
- Si se vuelve a declarar una variable declarada con var, esta no perderá su valor. No se puede redeclarar una varible declarada con let y const
- Si en una operación aritmética se encuentra un String, los números siguientes se comportarán como String.
- Es una convención utilizar _ al inicio del nombre de una variable .^es^condida"

3.7. Let y const

- Estas dos le brinda a las variables un ámbito, las variables declaradas con var tienen un ámbito global
- Ojo que si se puede redeclararüna variable let, per en un ámbito diferente (incluso si es un ámbito hijo y el padre tiene declarada ya la variable)
- Las variables const deben ser asignadas al momento de ser declaradas. Definen una referencia constante, no un valor constante.
- Usa const cuando declares arreglos, objetos, funciones y regEXP, su valor puede cambiar que la referencia no

3.8. Aritmética

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Figura 2: Operadores de asignación

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Figura 3: Operadores de comparación

- Los strings se comparan alfabeticamente
- Cuando se unsa + en Strings es un operador de concatenación
- Los operadores lógicos son los mismos que en Java
- typeof: devuelve tipo de variable, instanceof: true si es una instancia de un tipo de objeto
- El operador ** es de potenciación

3.9. Tipos de Datos

- Los tipos son dinámicos, la misma variable puede ser usada para contener diferentes tipos de datos
- BigInt() puede ser usado para representar valores más grandes que 64 bit (todas las variables tienen este almacenamiento)

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

Figura 4: Operadores bit a bit

3.10. Funciones

- Bloque de código designado a una tarea en particular
- `function name(parameter1, parameter2, parameter3)`
- En la función , los parámetros se comportan como variables locales
- Se invoca cuando un evento ocurre, invocado por código JS o autoinvocado.
- return detiene la función y retorna una valor
- Al colocar el nombre de la función sin paréntesis, se la trata como objeto
- Las funciones tiene su propio ámbito

3.11. Objetos

- Los objetos también son variables, pero contienen más valores
- `const car = {type:"Fiat", model:"500", color:"white"};`
- Los espacios y saltos de línea no son importantes aquí
- Para acceder a una propiedad del objeto `car.type` o `car["type"]`
- Los objetos pueden tener métodos, los cuales se tratan como propiedades en estos `fullName : function(){ return`
- this tiene diferentes significados, en la tabla de abajo se encuentra su explicación
- En una definición de función, this se refiere al propietario de la función (objeto que tiene la función)
- `objectName.methodName()`
- No es bueno declarar literales como objetos.

3.12. Strings

- Una cadena de plantilla permite comillas dobles y simples dentro de ella `let text = ‘He’s often called "Johnny'`
- Para el largo de un String se utiliza la propiedad length (como arrayas en Java)
- Backslash permite .[es]caparçaracteres
- OJO:Comparar dos objetos en JS siempre retorna false
- charAt() devuelve el carácter de la posición especificada, charCodeAt() devuelve el código del carácter en UTF-16 (un entero)
- at(), es lo mismo que chatAt(), funciona desde 2022, admite posiciones negativas
- También se puede acceder como si fuera un arreglo, es impredecible, si no encuentra caracter retorna indefinido, charAt retorna un string vacío. Es solo de lectura, no se puede modificar contenido como en un arreglo.
- slice(start, end), extrae una parte del string y lo retorna en un nuevo string, si solo se ingresa un parámetro, entonces extrae desde ahí hasta el final del string. Si el parámetro es negativo, cuenta desde atrás
- Con substring() pasa algo similar, pero los valores negatios se tratan con 0.
- substr(start, length), si el inicio es negativo se cuenta desde atrás
- toUpperCase() y toLowerCase(), para mayúsculas y minúsculas.
- Los Strings son inmutables
- trim() remueve los espacios en blanco a los costados del string. trimStart() los remueve solo del comienzo, trimEnd() del final. (2019)
- repeat(n), repite n veces el string
- replace(pattern, rep) replaza la primera coincidencia. Si se desea que no se diferencia entre mayúsculas y minúsculas /MICROSOFT/i. Para reemplazar todas las coincidencias se puede usar expresiones regulares /Microsoft/g
- replaceAll() funiona con expresiones regulares (2021)
- split() devuelve un arreglo a partir de un separador
- indexOf() posición de la primera ocurrencia, -1 si no encuentra nada
- lastIndexOf() la última ocurrencia
- Ambos admiten un segundo parámetro, que es la psición de donde empezará a buscar
- search() busca un string o una expresión regular, no tiene un segundo parámetro, indexOf() no acepa expresiones regulares
- match() retorna un array que contiene los resultados de las coincidencias. matchAll() devuelve un iterador (2020)
- includes() retorna true si contiene un valor específico, Acepta un segundo parámetro de posición
- startsWith() y endsWith(), empieza o termina con una cadena
- Tempate Strings: Entre comillas invertidas, aceptan multilíneas, acepta la interpolación con ${...}

3.13. Números

- Los números grandes se pueden expresar con notación científica
- Todos los números son almacenados como reales flotantes precisos en 64 bits, el número se almacena en los bits del 0 al 51, el exponente del 52 al 62 y el signo en el bit 63
- Los números enteros tiene una precisión de hasta 15 dígitos
- La aritmétca de la coma flotante no es siempre precisa
- JS siempre trata de hacer las operaciones correctas (+,-, /, *) cuando se trata de strings numéricos. Sin embargo, + es un operador de concatenación para strings, por lo suqe no funciona correctamente
- NaN es una palabra reservada que indica que el número no es algo admitido. isNan() nos ayuda a determinar este valor
- Cualquier operación matemática con NaN será otro NaN
- Infinity es el valor que JS devolverá s calcula un número fuera del mayor posible
- Hexadecimales, Js interpretanúmeros hexadecimales si están precedidos por 0xFF (nunca escribir un número con 0 a la izquierda ya que algunas versiones lo interpretan como octales)
- BigInt es el segundo tipo numérico de dato en JS
- Number.MAXi_SAFE_INTEGER y Number.MIN_SAFE_INTEGER, Number.EPSILON, Number.MAX_VALUE, Number.MIN_VALUE, Number.POSITIVE_INFINITY, Number.NEGATIVE_INFINITY, Number.NaN
- isInteger() para saber si es un número enteros
- toString(), retorna un número como String
- toExponential(), retorna un número escrito en su notación exponencial
- toFixed(), retorna un número escrito con un número de decimales
- toPrecision(), retorna un número con un laro especial
- valueOf(), retorna un número como número
- De variable a número, Number(), parseFloat() parseInt()

3.14. Arreglos

- `const array_name = \[item1, item2, ...\];`
- Espacios y saltos de línea no importan en la declaración del arreglo
- El arreglo se pueden inicializar como objeto, pero para más simplicidad es mejor hacerlo como arriba
- Para convertir un arreglo a String simplemente hay que utilizar el método toString()
- atributo length para saber su largo, sort() ordena el arreglo
- Se puede añadir otro elemento utilizando push(), o también utilizando la propiedad length

- No se puede crear un objeto array (con new) con un solo elemento, digamos new Array(3), crea un array de tres elementos indefinidos
- Array.isArray() compureba si el elemento es un arreglo o no
- at() permite ingresar números negativos para contar las posiciones desde atrás
- join() devuelve los elementos del arreglo en String pero divididos pors el parámetro del método
- pop() guarda el último elemento y lo quita del arreglo
- shift() remueve y guarda primer elemento del arreglo
- unshift() añade un nuevo elemento al comienzo
- keyword delete borra un elemento, pero deja un espacio en indefinido
- concat() concatena dos o más arreglos
- splice(start, end) para eliminar elementos sin dejar agujeros en el arreglo
- toSpliced() hace lo mismo pero crea una copia del arreglo para no modificar el original
- slice() elimina elementos sin modificar el arreglo original

3.15. Fecha

```
new Date()
new Date(date string)

new Date(year,month)
new Date(year,month,day)
new Date(year,month,day,hours)
new Date(year,month,day,hours,minutes)
new Date(year,month,day,hours,minutes,seconds)
new Date(year,month,day,hours,minutes,seconds,ms)

new Date(milliseconds)
```

Figura 5: Constructores Date

Method	Description
getFullYear()	Get **year** as a four digit number (yyyy)
getMonth()	Get **month** as a number (0-11)
getDate()	Get **day** as a number (1-31)
getDay()	Get **weekday** as a number (0-6)
getHours()	Get **hour** (0-23)
getMinutes()	Get **minute** (0-59)
getSeconds()	Get **second** (0-59)
getMilliseconds()	Get **millisecond** (0-999)
getTime()	Get **time** (milliseconds since January 1, 1970)

Figura 6: Métodos de get de Date

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

Figura 7: Métodos de set de Date

- Los objetos de fecha son estáticos, no están corriendo
- toDateString(), toUTCString(), toISOString()

3.16. Math

Math.round(x)	Returns x rounded to its nearest integer
Math.ceil(x)	Returns x rounded up to its nearest integer
Math.floor(x)	Returns x rounded down to its nearest integer
Math.trunc(x)	Returns the integer part of x (new in ES6)

Figura 8: Algunos métodos de Math

- Es un objeto estático, por ello no tiene constructor
- Math.random() devuelve un número aleatorio entre 0 y 1

3.17. If,else, switch

- Funcionan tal como en Java
- El default del switch no necesariamente tiene que ser el último Bloque
- switch utiliza la comparación estricta === (valor y tipo)

3.18. Bucles

- For normal funciona como en Java
- For in es utilizado para iterar en las propiedades de un objeto `for (key in object)`, también puede servir en arreglos `for (variable in array)`
- For of es para objetos iterables `for (variable of iterable)` arreglos, strings, mapas nodelists, etc
- while y do while son lo mismo que en Java
- Break puede ayudarnos a salir del Bucles

- continue interrumpe una iteración de acuerdo a un caso específico, continúa con la siguiente iteración
- Un label es un conjunto de snetencias con un nombre label: {sentencias} , break y continue se pueden utilizar aquí break label; contnue label;

3.19. Mapas

Method	Description
new Map()	Creates a new Map
set()	Sets the value for a key in a Map
get()	Gets the value for a key in a Map
delete()	Removes a Map element specified by the key
has()	Returns true if a key exists in a Map
forEach()	Calls a function for each key/value pair in a Map
entries()	Returns an iterator with the [key, value] pairs in a Map
Property	**Description**
size	Returns the number of elements in a Map

Figura 9: Algunos métodos de Mapas

3.20. Expresiones regulares

- Modificadores: i, case sensitive; g, encontrar todas las coincidencias; m, con diferentes líneas; d, coincidencia de inicio y final

abc , [0-9], (x—y) como en perl

- \d para dígito, \s espacio en blanco , \b match al inicio o final de la palabra
- \uxxxx encontrar un caracter unicode especial
- *, +; ? como el Perl
- el objeto expresión regular `const pattern = /e/; pattern.test("The best things in life are free!");` retorma true si lo contiene, en esste caso si es true

3.21. Errores

- try catch como en Java, el parámetro de catch es err
- El objeto error de JS tiene dos propiedades, name y message
- COn throw puedes crear un error personalizado, para manejarlo luego con try catch
- Finally permite ejecutar código después del try catch, independientemente del resultado

Error Name	Description
EvalError	An error has occurred in the eval() function
RangeError	A number "out of range" has occurred
ReferenceError	An illegal reference has occurred
SyntaxError	A syntax error has occurred
TypeError	A type error has occurred
URIError	An error in encodeURI() has occurred

Figura 10: Seis errores de JS

3.22. Fetch

Listing 1: Código JavaScript que utiliza la api fetch para extaer una archivo

```
function show(){
  if(document.getElementById("keyword").value == ''){alert("Ingrese palabras clave en el campo");
      ↪ return;}
  document.getElementsByTagName("table").item(0).style.display = "block";
  fetch('data_utf8.csv')
    .then(response => {
      if(!response.ok){
        throw new Error('No se concretó la solicitud');
      }
      const t = response.text();
      console.log(t);
      return t;
    })
    .then(dat => {
      const data = String(dat);
      table(data);
    })
    .catch(err => console.error("Error", err));
}
function table(data){
  const table = document.getElementsByTagName("table").item(0);
  const tbody = table.getElementsByTagName("tbody").item(0);
  tbody.innerHTML = '';
  const kind = document.getElementById("kind").value;
  const keyword = document.getElementById("keyword").value;
  const lines = data.split("\n");
  for(const line of lines){
    const mydata = line.split("|");
    const params = find(line, mydata);
    const value = params.get(kind);
    if(!(typeof value === 'undefined') && value.includes(keyword.toUpperCase())){
      const row = document.createElement("tr");
      tbody.appendChild(row);
      const totable = [mydata[1], mydata[2], mydata[3],
        mydata[4], mydata[7], mydata[8],
        mydata[10], mydata[11], mydata[12],
        mydata[15], mydata[16], mydata[17],
        mydata[18]];
      for(const unis of totable){
        const el = document.createElement("td");
        el.innerHTML = unis;
        row.appendChild(el);
      }
    }
  }
}
```

```
46 function find(line,mydata){
47   const uni = new Map([
48     ["name", mydata[1]],
49     ["period", mydata[4]],
50     ["localRegion", mydata[10]],
51     ["studyProgram", mydata[16]]
52   ]);
53   return uni;
54 }
```

3.23. Recursos

- Repositorio de GitHub para commits `https://github.com/gusCreator/pweb-course/tree/main/lab10`.