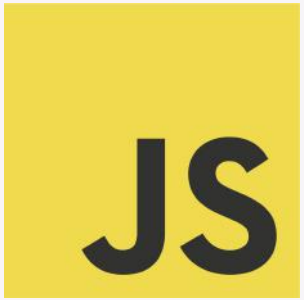
El futuro digital es de todos

MinTIC

Introducción a JavaScript



UNIVERSIDAD SERGIO ARBOLEDA

Mision TIC 2022

¿Qué es JavaScript?

- JavaScript es un lenguaje de programación interpretado, dinámico, débilmente tipado y orientado a objetos.
- Es un lenguaje de código abierto y multiplataforma.
- Fue diseñado para crear aplicaciones web dinámicas.

¿Por qué JavaScript?

- Según Stack Overflow (2020), JavaScript fue durante ese año el lenguaje de programación más comúnmente utilizado (69.7%).

- El uso de JavaScript se ha extendido ampliamente al desarrollo de aplicaciones móviles, aplicaciones de escritorio y videojuegos.

- Debido a la alta demanda, existen millones de vacantes laborales para desarrolladores con conocimientos en este lenguaje de programación y, según el Bureau of Labor Statistics (2020), esta demanda seguirá creciendo en un promedio de 13% anual.

Lenguajes relacionados

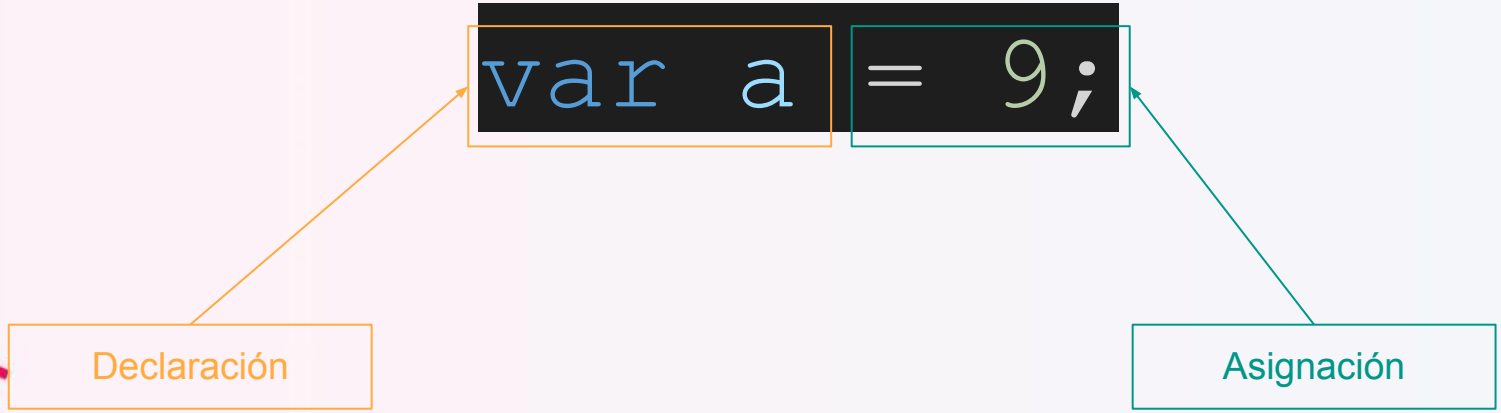
- Dentro del desarrollo web, JavaScript se encuentra estrechamente relacionado con HTML y CSS.

Tipos de datos en JavaScript

Javascript provee al programador 7 tipos de dato que se pueden utilizar en este lenguaje:

1. undefined
2. null
3. boolean
4. string
5. symbol
6. number
7. object

Declaración y asignación de variables en JavaScript



Declaración y asignación de variables en JavaScript

```
1.  var a; //Declaración de una variable.
2.  a = 7; //Asignación de un dato a una variable.
3.  var b = 5; // Declaración y asignación en una única
    línea.
4.  b = a; //Asignación del contenido de una variable en
    otra.
```

Existen tres distintas formas de declarar una variable en este lenguaje, dependiendo de su uso:

1. `var miNombre = "Juan"; //Cuando la variable se pretende utilizar a lo largo del programa (variable global).`
2. `let numero = 8; //Cuando la variable se utilizará de manera local con ámbito de bloque.`
3. `const PI = 3.1416; //Cuando se pretende declarar un dato que permanecerá constante durante la ejecución del programa.`

Gramática y tipos en JavaScript

- JavaScript está fuertemente influenciado por la sintaxis de C, C++ y Java, así como también algunos elementos de Perl y Python.

  Ejemplo, ciclo for:

```
for (let i = 0; i < 5; i++) {
   text += "El número es " + i + "<br>";
 }
```

- Este lenguaje distingue entre mayúsculas y minúsculas (en inglés case-sensitive).

```
variable != VARIABLE != vArIaBlE
```

Gramática y tipos en JavaScript

- JavaScript utiliza el conjunto de caracteres Unicode, por lo tanto, la palabra “antigüedad” puede ser utilizada como una variable

```
var antigüedad = 15;
```

Buenas prácticas específicas para JavaScript

- Si bien la utilización del punto y coma al no es obligatorio para este lenguaje, se recomienda utilizar (;) al final de cada instrucción.

Comentarios en JavaScript

- La sintaxis de los comentarios es la misma que en C++ y Java

```
// Este es un comentario de una sola línea

/* Este es
 * un comentario
 * de varias líneas
 */
```

Operaciones básicas en JavaScript:

- Suma de números:

```
var suma = 2 + 4; // = 6
```

- Resta de números:

```
var resta = 30 - 10; // = 20
```

- Multiplicación de números:

```
var multiplicacion = 4 * 20 // = 80
```

- División de dos números:

```
var division = 20 / 4; // = 5
```

- Operadores de incremento unario:

```
var miVariable = 15;

miVariable++; // = 16
```

- Operadores de decremento unario:

```
var miOtraVariable = 32;

miOtraVariable--; // = 31
```

- Los números decimales en JavaScript son de tipo flotante (floating point).

```
var miDecimal = 7.8;

var miOtroDecimal = 0.0009;
```

- Las operaciones básicas con decimales se efectúan de igual manera que con los enteros.

```
var productoDecimal = 2.5 * 2; // = 5

var cocienteDecimal = 4.4 / 2.0; // = 2.2
```

- Para obtener el residuo de una división, se utiliza el símbolo “%”.

```
var miResiduo;

miResiduo = 11 % 3; // = 2
```

Asignación compuesta en JavaScript

- Los operadores de asignación compuesta en JavaScript combinan el operador de la asignación simple con otro operador binario. Por ejemplo:

```
var a = 8;
var b = 5;
var c = 9;
var d = 20;

a += 10; // = 18
b -= 4;  // = 1
c *= 6;  // = 54
d /= 4;  // = 5
```

Booleanos en JavaScript

- Boolean representa una entidad lógica y puede tener dos valores: true, y false.
- Este tipo se utiliza comúnmente para almacenar valores de sí o no: true significa “sí, correcto, verdadero”, y false significa “no, incorrecto, falso”.

```
//Ejemplo


var verdadero = true;
var falso = false;
```

Operadores lógicos y relacionales en JavaScript

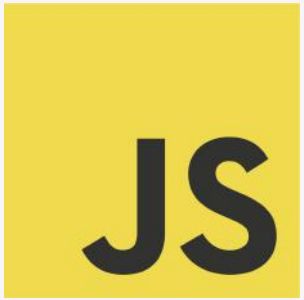
- Los operadores lógicos y relacionales están estrechamente relacionados con los tipos de dato booleanos. Las expresiones donde se utilizan operadores lógicos y relacionales devuelven un valor booleano, es decir, verdadero (true) o falso (false).
- Además de los operadores habituales existen los operadores === que se interpreta como “es estrictamente igual” y !== que se interpreta como “no es estrictamente igual”.

Operadores lógicos y relacionales en JavaScript:

Operadores lógicos y relacionales	Descripción	Ejemplo
==	Es igual	`a == b`
===	Es estrictamente igual	`a === b`
!=	Es distinto	`a != b`
!==	Es estrictamente distinto	`a !== b`
< , <= , > , >=	Menor, menor o igual, mayor, mayor o igual	`a <= b`
&&	Operador and (y)	`a && b`
||	Operador or (o)	`a || b`
!	Operador not (no)	`!a`

El futuro digital es de todos MinTIC

Strings



UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC2022

Declaración de strings en JavaScript

- Tanto comillas simples como dobles son válidas para la asignación de valores a un string en JavaScript.

```
var miNombre = "Juan";
var miApellido = 'Perez';
```

Concatenación de strings en JavaScript

- Existen dos maneras de concatenar un string en Javascript:

1. Utilizando el símbolo “+”.

```
var miPalabra = "Una" + "palabra"; // = "Una palabra"
```

2. Utilizando la asignación compuesta “+=”.

```
miPalabra += "más larga." // = "Una palabra más larga."
```

Concatenación de strings con variables:

- Es posible también concatenar el valor de otra variable dentro de un string de la siguiente manera:

```
var nombre = "Juan";

var miFrase = "Hola, " + nombre + ". Cómo estás?"


// Si se imrimiese en consola, el valor de miFrase sería
// "Hola, Juan. Cómo estás?"
```

Principio de inmutabilidad de los strings:

- Una vez creado un string en JavaScript, este no puede ser modificado. La única manera de modificar su contenido es re-asignandolo.

Métodos útiles con strings

- La propiedad length permite conocer la longitud de un string. Por ejemplo.

```
var palabra = "JavaScript";

var longitud = palabra.length; // = 10
```

Métodos útiles con strings

- También es posible extraer un carácter específico del string utilizando corchetes al final de la variable y especificando su índice. Por ejemplo, para obtener el primer carácter de una palabra, podemos escribir:

```
var nombre = "Juan";
var primeraLetra = nombre[0]; // = "J"
/* Tener en cuenta que JavaScript utiliza el principio de
*indexación basada en cero. Por lo tanto, los índices se
comienzan a contar a partir del número 0. */
```

Métodos útiles con strings.

- Para encontrar el último carácter de una palabra, es posible utilizar tanto el método length como la utilización de corchetes. Por ejemplo,

```
var miPalabra = "Casa";
var ultimaLetra = miPalabra[miPalabra.length-1] // = 'a'
// Se resta uno, debido al principio de indexación
//mencionado anteriormente.
```

Métodos útiles con strings.

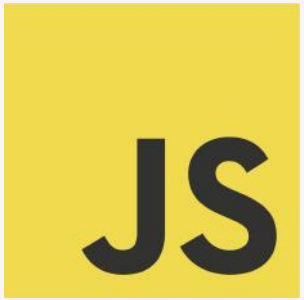
- Muchas veces resulta útil inicializar un string en blanco para después asignar y/o concatenar otros valores. Esto se puede lograr utilizando únicamente las comillas (“”).

```
var stringVacio = "";
```

El futuro digital es de todos

MinTIC

Arreglos



UNIVERSIDAD SERGIO ARBOLEDA

Mision TIC 2022

Arreglos en JavaScript

- Los arreglos en JS permiten almacenar varios datos en una misma estructura.
- Se utilizan los corchetes ([ ]) para indicar el principio y el final del arreglo.
- Cada elemento en el arreglo se separa con una coma (,).
- Los elementos en el arreglo pueden contener cualquier tipo de dato.

Arreglos anidados o multidimensionales en JavaScript

- Es posible anidar uno o varios arreglos dentro de otro arreglo con JavaScript. Por ejemplo:

```
var miArreglo = [ [ "Juan" , 3],[ "Pedro" , 15] ];


//La variable miArreglo en este caso corresponde a un
//arreglo bidimensional
```

Acceso a los datos de un arreglo por medio de índices

- De la misma manera que con los strings, es también posible acceder a los índices de un arreglo mediante el uso de corchetes.

```
//Ejemplo
var miArreglo = [10, 20, 30, 40, 50];
var miDato = miArreglo[1] // = 20
var miOtroDato = miArreglo[0] // = 10
```

Modificar datos de un arreglo con índices:

- En el caso de los arreglos, sí es posible modificar los datos contenidos dentro del mismo mediante el uso de índices.

```
//Ejemplo
var miArreglo = [30,40,50];
miArreglo[1] = 45;
//miArreglo ahora es [30,45,50]
```

Acceso a arreglos multidimensionales con índices:

- Para este caso, resulta necesario no solo acceder al índice del primer arreglo, sino también al índice del arreglo anidado. Por lo tanto, se utilizarían dos juegos de corchetes [ ] [ ]. Por ejemplo:

```
// Ejemplo
// Indice 1          0       1       2
// Indice 2        0 1 2   0 1 2   0 1 2
var miArreglo = [[1,2,3],[4,5,6],[7,8,9]];
var miDato = miArreglo[1][2]; // = 6
```

Manipulación de arreglos con el método push()

- El método push() permite agregar nuevos datos al final de un arreglo

```
var miArreglo = ["María", "José", "Alejandra"];
miArreglo.push("Camilo", "Daniela");

// miArreglo ahora es ["María", "José", "Alejandra",
//"Camilo", "Daniela"];
```

Manipulación de arreglos con el método pop()

- Es posible remover el último elemento de un arreglo con la utilización del método pop() de la siguiente manera:

```
var miArreglo = [1,2,3];
var datoRemovido = miArreglo.pop();
// datoRemovido es igual a 3 y miArreglo ahora es [1,2].
```

Manipulación de arreglos con el método shift()

- Este método permite extraer el primer dato del arreglo.

```
// Ejemplo
var miArreglo = ["Perro", "Gato", "Pájaro"];
var datoRemovido = miArreglo.shift();
// datoRemovido = "Perro" y miArreglo ahora es ["Gato",
// "Pájaro"]
```

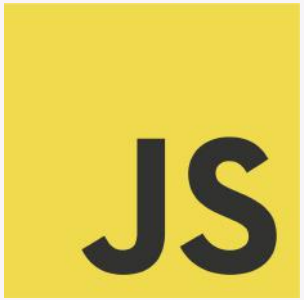
Manipulación de arreglos con el método unshift()

- El método unshift() permite adicionar un nuevo elemento al principio de un arreglo.

```
var miArreglo = ["Colombia", "Argentina", "Uruguay"];
miArreglo.unshift("Ecuador");
// MiArreglo ahora es ["Ecuador", "Colombia",
//"Argentina", "Uruguay"]
```

El futuro digital es de todos MinTIC

Funciones



UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC2022

Funciones en JavaScript:

- Las funciones (function) en JavaScript son bloques de código que componen un determinado comportamiento que puede ser invocado las veces que sea necesario.
- Las funciones permiten escribir código altamente eficiente, ya que puede ser reutilizado innumerables veces.

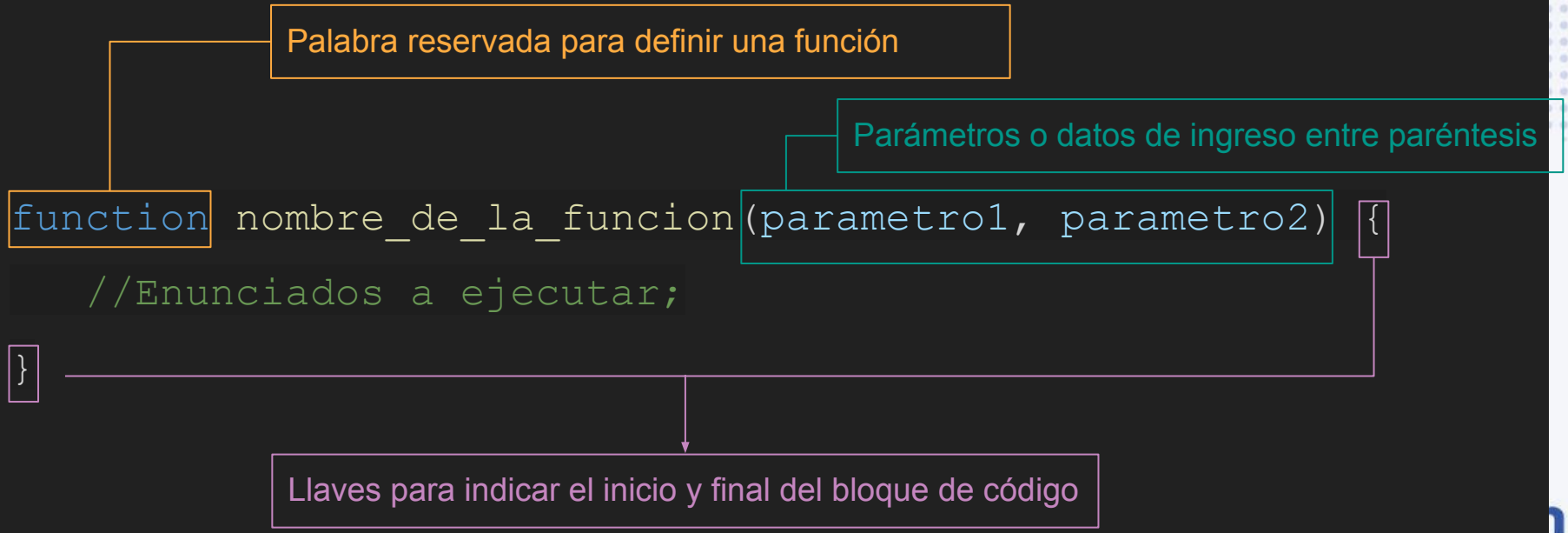
Sintaxis general de una función

- La estructura general de una función en JavaScript presenta este aspecto:

```
function nombre_de_la_funcion(parametro1, parametro2) {
   //Enunciados a ejecutar;
} // Fin de la función.
```

Sintaxis general de una función

- La estructura general de una función en JavaScript presenta este aspecto:



Ejemplos de funciones en JavaScript

- A continuación dos ejemplos de funciones escritas en JavaScript:

```
function suma(num1, num2){
    console.log(num1+num2);
}
function calcularIva(subtotal){
    console.log(subtotal * 1.19);
}
// El método console.log() se utiliza para imprimir un
output por //consola
```

El futuro digital es de todos MinTIC

Retorno de valores en una función:

- Opcionalmente se puede retornar un valor desde la función utilizando la palabra reservada ***return***.

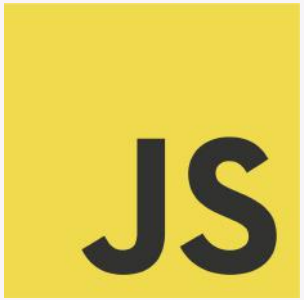
```
//Ejemplo
function multiplicacion(num1, num2){
    return (num1 * num2);
}
var resultado = multiplicacion(5,4); // = 20
```

UNIVERSIDAD SERGIO ARBOLEDA

Mision TIC 2022

El futuro digital es de todos MinTIC

Estructura de control condicional



UNIVERSIDAD SERGIO ARBOLEDA

Mision TIC 2022

Estructura de control “if/else” en Javascript:

- Esta estructura se utiliza comúnmente para que el programa tome decisiones con base en una condición. Su sintaxis en JavaScript es así:

```
if (condicion){
    // Instrucciones que se ejecutarán si se cumple la condición
}
else{
    // Instrucciones que se ejecutarán si NO se cumple la condición
}
```

Estructura de control “if/else” dentro de una función:

- A continuación un ejemplo de la utilización de la estructura de control “if” dentro de una función:

```
function ciertoOFalso(condicion){
    if(condicion){
        return "es cierto.";
    }
    return "es falso.";
}
```

Palabra reservada “else” en una estructura condicional:

- La palabra reservada “else” tiene como propósito dar una serie de instrucciones en caso de que ningún “if” se cumpla anteriormente. Por ejemp

```
function positivoONegativo(numero){

    if (numero > 0){
        return "El número es positivo";
    }
    if(numero == 0){
        return "El número no es positivo ni negativo";
    }
    else{
        return "El número es negativo"
    }
}
```

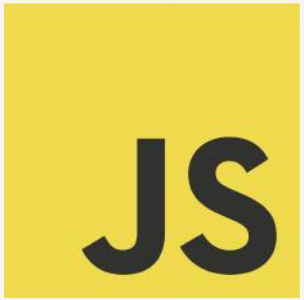
Palabra reservada “else if” en una estructura condicional:

- La palabra reservada “else if” se utiliza para concatenar varias estructuras if, como en el siguiente caso:

```
if (condicion1)
 // instrucción 1;
else if (condicion2)
 // instrucción 2;
else if (condicion3)
//instrucción 3;
//...
else
 // instruccion N;
```

El futuro digital es de todos MinTIC

Estructura de control switch case



UNIVERSIDAD SERGIO ARBOLEDA

Mision TIC 2022

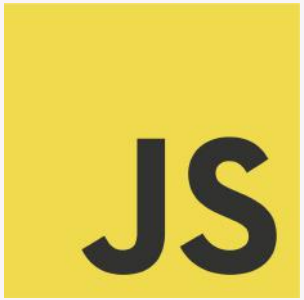
Estructura switch case en JavaScript

- Existen situaciones en las cuales el flujo del programa puede depender de diversas opciones para continuar por un camino u otro. Para esto existe una solución concisa, que se llama switch case, cuya estructura general es:

```
switch(variable){ //palabra reservada 'switch' y la variable como parámetro.
   case 1:
       //instrucciones para el caso 1;
       Break; // palabra reservada break que indica al programa que salga de la estructura.
   case 2:
       //instrucciones para el caso 2;
       break;
   case 3:
       //instrucciones para el caso 3;
       break;
   default:
       //instrucciones para el resto de casos;
       break;
}
```

El futuro digital es de todos MinTIC

Estructura de control While



UNIVERSIDAD SERGIO ARBOLEDA

Mision TIC 2022

Estructura de Control While en JavaScript:

- La estructura de control While tiene el propósito de ejecutar una serie de instrucciones mientras que se cumpla cierta condición. Su estructura general es:

```
while(condicion){ //Mientras que la condición se cumpla
        //Bloque de instrucciones a ejecutar
}
```

Estructura de Control While en JavaScript:

- En este ejemplo, se carga un arreglo mediante la implementación de una estructura While.

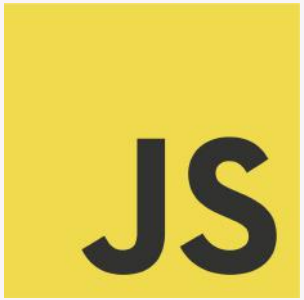
```
var miArreglo = []; // Se inicializa un arreglo vacío
var i = 0;

// Mientras que la variable i sea menor que 5, se ejecutan las
// instrucciones dentro del ciclo;

while(i < 5){
    miArreglo.push[i]; //Se agrega el valor de i a miArreglo.
    i++; //Se incrementa el valor de i por 1.
}
// Ahora miArreglo es [0,1,2,3,4]
```

El futuro digital es de todos MinTIC

Estructura de control Do While



UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC2022

Estructura de control Do While en JavaScript:

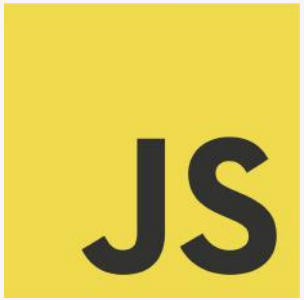
- Esta es una estructura similar al bucle While, con la única diferencia de que en la estructura Do While, el bloque de código se ejecuta al menos una vez antes de verificar la condición. Por ejemplo:

```
var miArreglo = [];
var i = 10;

do{ //Se ejecuta el bloque de código antes de verificar la condición
    miArreglo.push(i);
    i++;
} while(i<15);
```

El futuro digital es de todos MinTIC

Estructura de control For



UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC2022

Estructura de control For en JavaScript

- Esta es tal vez la estructura de control más común en JavaScript. Su estructura básica es:

```
for(let i = 0; i < 5; i++){
    //Bloque de instrucciones
}
```

Estructura de control For en JavaScript

- En este ejemplo se carga un arreglo mediante la ejecución de una estructura for.

```
//Se inicializa un arreglo vacío
var miArreglo = [];
//Se itera con un a estructura for
for(let i = 0; i < 5; i++){
   miArreglo.push(i);
}
// Ahora miArreglo es [0,1,2,3,4
```

Estructura For anidada:

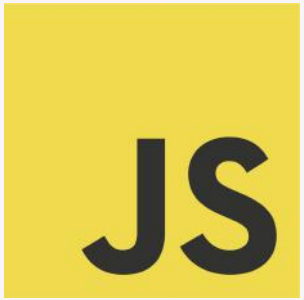
- La anidación de bucles es necesaria para realizar ciertas tareas o procesamientos un poco más complejos que una estructura for simple, o para recorrer arreglos anidados, como en el ejemplo a continuación:

```
//Se inicializa un arreglo bidimensional.
var arreglo = [[1,2,3],[4,5,6],[7,8,9]];

//Se recorre tanto el arreglo principal como aquellos anidados
for (let i = 0; i<arreglo.length; i++ ){
    for(let j = 0; j<arreglo[i].length; j++){
        console.log(arreglo[i][j]);
    }
}
```

El futuro digital es de todos MinTIC

Objetos



UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC2022

El futuro digital es de todos MinTIC

Objetos en JavaScript:

- Como se mencionó anteriormente, JavaScript es un lenguaje diseñado en un paradigma orientado a objetos.
- Un objeto simplemente es una colección de propiedades, cada una de ellas asociada a un valor. Su propósito es crear estructuras comparables con los objetos de la vida real.
- Las propiedades de un objeto básicamente son lo mismo que una variable, excepto por el nexo de la misma con el objeto.
- Adicionalmente, el valor de una propiedad puede ser una función. En este caso dicha propiedad se le conoce como método.

El futuro digital es de todos MinTIC

Estructura general de un objeto en JavaScript:

- A continuación una breve explicación sobre cómo crear un objeto en JavaScript.

```
var miCarro = { //Llaves al principio y al final del objeto

    "marca": "Ford", //propiedad 1 seguido por una coma
    "modelo": "Mustang", // propiedad 2
    // ...
    "anio": 1969 // propiedad n
};
```

UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC2022

Acceso a las propiedades de un objeto con notación de punto:

- Existen dos maneras de acceder a un objeto en JavaScript. La primera de estas se denomina acceso por notación de punto. A continuación un ejemplo.

```
var persona = { // Se crea un objeto nuevo con sus propiedades
   "altura": 1.75,
     "edad": 35,
     "sexo": "Masculino"
};
// Se accede a sus propiedades con el nombre del objeto seguido de un punto
var altura = persona.altura; // 1.75
var edad = persona.edad; // 35
var sexo = persona.sexo; // "Masculino"
```

Acceso a propiedades de un objeto con corchetes:

- De la misma manera que con el acceso a arreglos por medio de índices entre corchetes, es posible acceder a las propiedades de un objeto utilizando [ ].

```
var persona = { // Se crea un objeto nuevo con sus propiedades
   "altura": 1.75,
     "edad": 35,
     "sexo": "Masculino"

};
// Se accede a sus propiedades con el nombre del objeto y la propiedad entre
corchetes y //paréntesis

var altura = persona["altura"]; // 1.75
var edad = persona.edad["edad"]; // 35
var sexo = persona.sexo["sexo"]; // "Masculino"
```

Modificación de las propiedades de un objeto:

- Las propiedades de un objeto pueden ser modificadas mediante la notación de punto de la siguiente manera:

```
var persona = { //Se inicializa el objeto
    "nombre" : "Camilo" // Se asigna "Camilo" a nombre
};
// Modificación de la propiedad:
persona.nombre = "Juan"; //Se modifica la propiedad del
//objeto por el string "Juan"
```

Agregar una nueva propiedad a un objeto:

- Es posible agregar una nueva propiedad a un objeto, mediante la notación de punto, de una manera muy similar al ejemplo pasado.

```
var persona = { //Se crea un nuevo objeto llamado persona
   "nombre" : "Camilo"
};
persona.apellido = "Perez"; //Se ha agregado la propiedad
// apellido al objeto persona
```

Eliminar una propiedad de un objeto:

- Mediante la palabra reservada ***delete*** es posible eliminar una propiedad de un objeto ya instanciado de la siguiente manera:

```
//Se crea un nuevo objeto persona con las propiedades nombre y apellido.
var persona = {
   "nombre"   : "Camilo",
   "apellido": "Perez"
};
delete persona.apellido; // se elimina la propiedad apellido del objeto
```

Objetos anidados en JavaScript:

- Este lenguaje permite crear objetos dentro de otros objetos. Mediante el uso de llaves se determina el inicio y el fin de un objeto. Por ejemplo:

```
var automovil = { //Se crea el objeto automóvil
    "marca" : "Ford",
    "modelo": "Mustang",
    "motor": { // Se anida el objeto motor dentro de automóvil (llaves)
        "cilindraje": 3500,
        "caballos"  : 175
    }// Fin del objeto motor
} // Fin del objeto automóvil
```

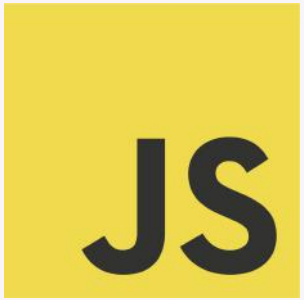
Acceso a objetos anidados en JavaScript:

- El acceso a objetos anidados se efectúa mediante la notación de puntos.

```
// Se crea un objeto automóvil, con un objeto motor anidado.
var automovil = {
   "marca" : "Ford",
   "modelo": "Mustang",
   "motor": {
        "cilindraje": 3500,
        "caballos"  : 175
   }
}
//Se accede al objeto anidado (objeto.objetoAnidado.propiedad)
var miCilindraje = automovil.motor.cilindraje; // = 3500
```

El futuro digital es de todos MinTIC

Expresiones Lambda



UNIVERSIDAD SERGIO ARBOLEDA

Mision TIC 2022

Expresiones Lambda en javaScript:

- Las expresiones Lambda (o expresiones de flecha) son una alternativa compacta a la manera tradicional de escribir las funciones. Sin embargo, las expresiones Lambda o de flecha tienen algunas restricciones en el lenguaje JavaScript. Su estructura básica es:

```
(param1, paramN) => instrucción;
```

Parámetros entre paréntesis y separados por coma

Expresión de flecha

Instrucción

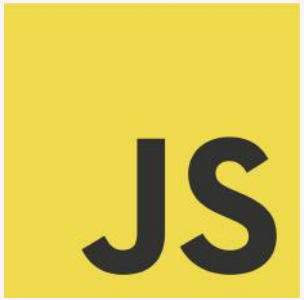
Comparación entre una función tradicional y una Lambda:

- A continuación una comparación entre dos funciones que, si bien ejecutan el mismo trabajo, se escriben de manera diferente:

```
// Función con escritura tradicional
function suma(a, b){
    return a + b + 100;
 }
// Expresión Lambda o de flecha
(a, b) => a + b;
```

El futuro digital es de todos MinTIC

Expresiones Regulares



UNIVERSIDAD SERGIO ARBOLEDA

Mision TIC 2022

El futuro digital es de todos MinTIC

Expresiones Regulares en JavaScript:

- Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de un string.
- En JavaScript, las expresiones regulares también son objetos.
- Son muy útiles para la validación de campos en un formulario, o para reemplazar patrones en un string.

UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC 2022

Estructura general de una Expresión Regular:

- Las expresiones regulares se pueden escribir de dos maneras distintas en javascript:

1. ```
   var expresionRegular = /ab+c/;
   ```
2. ```
   var expresionRegular = new RegExp('ab+c');
   ```

Ejemplos útiles de expresiones Regulares

- La expresión regular que contiene todas las letras del alfabeto se escribe de la siguiente manera:

```
var alfabeto = /[a-z A-Z]/;
```

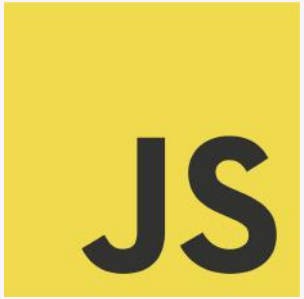
- La expresión regular que contiene los dígitos del 0 al 9 se escribe de la siguiente manera:

```
var digitos = /[0-9]/;
```

El futuro digital es de todos MinTIC

Ejercicio de Práctica:

Validar Formulario de Inicio de Sesión con JavaScript



UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC 2022

Nota:

- Tanto los archivos base para realizar este ejercicio, como también los archivos con el ejercicio resuelto, están disponibles en GitHub, bajo el siguiente enlace:

https://github.com/miguelsalazar88/introJavaScript.git

Objetivos del Ejercicio de Práctica

El objetivo principal de este ejercicio es el de crear un formulario de inicio de sesión con dos campos: nombre de usuario y contraseña.

Adicionalmente, se efectuará un proceso de validación con los requisitos comúnmente requeridos en los controles de formulario.

A este proceso se le denomina ***validación de formulario en el lado del cliente*** y ayuda a garantizar que los datos que se envían coinciden con los requisitos establecidos.

El futuro digital es de todos MinTIC

Herramientas computacionales necesarias para este proyecto:

- Existen varios IDE que se pueden utilizar para este proyecto (WebStorm, VS Code, Atom, Sublime Text, Komodo Edit, entre muchos otros).

- También es posible desarrollar este proyecto en compiladores de código online como codepen.io, scrimba.com o playcode.io.

- Para esta presentación se utilizará Codepen como herramienta para su implementación.

UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC 2022

Criterios de validación del formulario:

1. Nombre de Usuario:
   a. Validación de longitud (mínimo 8 caracteres)
   b. Validación de espacios (sin espacios entre caracteres)
2. Contraseña:
   a. Validación de longitud (mínimo 8 caracteres)
   b. Mínimo un caracter del alfabeto.
   c. Mínimo un dígito

HTML, CSS y JavaScript: un poco de contexto.

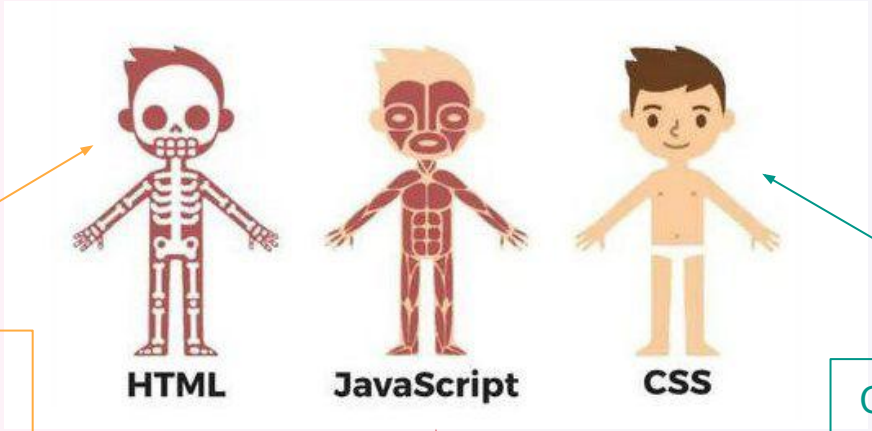
- HTML: lenguaje de marcado utilizado para estructurar elementos de la página, como párrafos, enlaces, títulos, tablas, imágenes e incluso videos.

- CSS: lenguaje de estilo utilizado para definir colores, fuentes, tamaños, posicionamiento y cualquier otro valor estético para los elementos de la página.

- Javascript: lenguaje de programación utilizado para hacer la página con más movimiento, pudiendo actualizar elementos de forma dinámica y manejar mejor los datos enviados y recibidos en la página.

HTML, CSS y JavaScript en una imagen:

Podemos usar el cuerpo humano como una página web como ejemplo, de la siguiente manera:



HTML representa la estructura del cuerpo, los huesos

Javascript es el músculo que le da movimiento al cuerpo.

CSS es la piel, el cabello y las ropas, creando la apariencia.

Código HTML

Se crea la estructura del formulario de inicio de sesión, con una casilla para el nombre de usuario, una para la contraseña y un botón para efectuar el inicio de sesión.

Puede descargar el archivo con nombre “baseFormulario.html”, que se encuentra en el fichero “archivoBase” del repositorio.

Código HTML

```
<html>
    <head>
        <script language="javascript">
            function validar(){
                var user = document.f1.username.value;
                var pass = document.f1.pass.value;
                // ***** Introduzca su código aquí *****
            }
        </script>
    </head>
    <body>
        <form name="f1" action="" onsubmit="validar();">
            <center>
            <h2>Formulario de Validación de Inicio de Sesión Usando JavaScript <h2>
            Usuario: <input type="text" name="username" value=""><br>
            Contraseña <input type="password" name="pass" value=""><br>
            <input type="submit" name="submit" value="Ingresar"><br>
            </center>
        </form>
    </body>    <
```

Esta es la sección del código (tag <script>) en la que se escribirá el código en JavaScript para la función validar().

Función Validar()

El primer paso para la creación de la función consiste en declararla dentro del tag <script> del archivo baseFormulario.html y asignar las variables con los valores introducidos por el usuario en el formulario de inicio de sesión.

```
<script language="javascript"> //Tag script del archivo html

function validar(){
   var user = document.f1.username.value; //Recibe el dato de usuario del formulario
   var pass = document.f1.pass.value;  //Recibe el dato de contraseña del formulario
}

</script>
```

Validación de longitud del nombre de usuario:

Una vez creada la función, se requiere validar que el nombre de usuario contenga más de 8 caracteres y no corresponda a un string vacío. De lo contrario, se imprime una alerta indicando el error.

```
if(user == "" || user.length < 9){
   alert("Por favor ingrese un nombre de usuario de más de 8 caracteres.");
}
```

Validación de ausencia de espacios (nombre de usuario):

Se requiere también que ninguno de los caracteres corresponda a un espacio (‘ ’). Para esto se implementa un bucle for con una variable bandera que cambia de valor 0 a 1 en caso de que encuentre un caracter con valor de espacio en el string (user).

```
var espacio = 0; //Variable bandera

for(let i = 0; i< user.length; i++){ //Ciclo for que itera por los caracteres del string
    ch = user.charAt(i); //Variable que toma el valor del carácter en el índice i
    if(ch == ' '){
        espacio = 1; //Cambio de valor en la variable bandera si encuentra un espacio
    }
}
```

Validación de la variable bandera (espacio):

- Una vez efectuado el bucle que pasa por todos los caracteres, se hace necesario verificar que la variable bandera (var espacio) no haya cambiado de valor. De lo contrario, se le indica al usuario mediante una alerta en el navegador.

```
if(espacio == 1){
    alert("Por favor ingrese un nombre de usuario sin espacios" );
    }
```

Validación de longitud de la contraseña:

- Se valida que la contraseña contenga más de 8 caracteres y no corresponda e un espacio vacío.

```
if(pass == ""|| pass.length < 9){
   alert("Por favor ingrese una contraseña de más de 8 caracteres." );
}
```

Validación de alfabeto y dígitos en la contraseña:

- Se crean dos expresiones regulares: una para el alfabeto y una para los dígitos del 0 al 9 para comprobar que al menos uno de cada uno existe en la contraseña mediante una estructura if y una variable bandera.

```
alfabeto = /[a-z A-Z]/; //Expresión regular para el alfabeto
digitos = /[0-9]/;      //Expresión regular para los dígitos del 0 al 9

var alfadigitos = 0; //Variable bandera

//La función .match() verifica que al menos un caracter del string coincida con la expresión
   if(pass.match(alfabeto) && pass.match(digitos)){
       alfadigitos =1;

   }

   if(alfadigitos==0){
       alert("Por favor introduzca al menos una letra del alfabeto y un dígito" );
   }
```

El futuro digital es de todos MinTIC

Referencias:

- Stack Overflow Developer Survey 2020. (2021). Stack Overflow. https://insights.stackoverflow.com/survey/2020
- Web Developers and Digital Designers : Occupational Outlook Handbook: : U.S. Bureau of Labor Statistics. (s. f.). U.S. Bureau of Labor Statistics. https://www.bls.gov/ooh/computer-and-information-technology/web-developers.htm
- freeCodeCamp.org. (2018, 10 de diciembre). Learn JavaScript - Full Course for Beginners [Video]. YouTube. https://www.youtube.com/watch?v=PkZNo7MFNFg

UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC 2022