Javascript

Introducción

Javascript

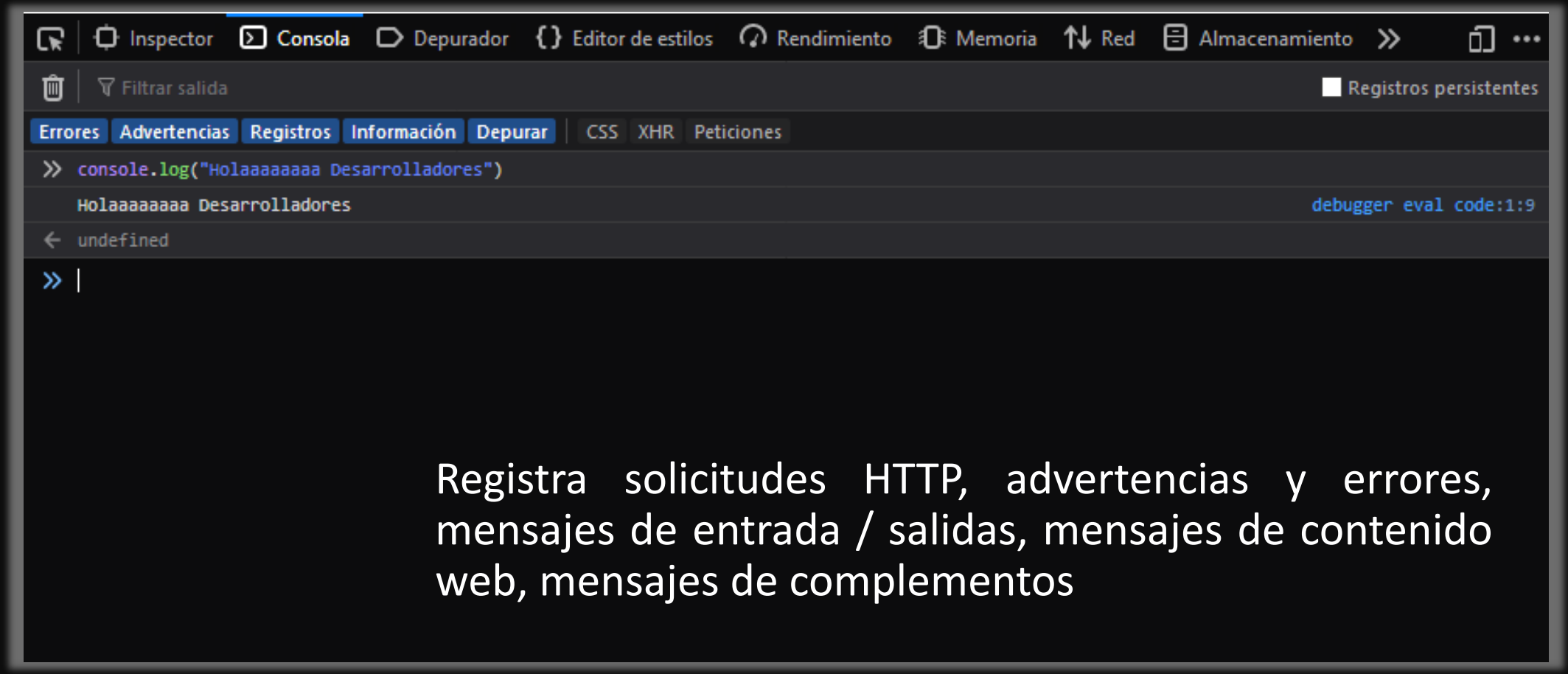
Ejemplo

https://codepen.io/Zaku/pen/EDaun

Es usado para añadir características interactivas a un sitio web.

Juegos, eventos, efectos dinámicos, animación, envío de datos, etc.

Consola



Registra solicitudes HTTP, advertencias y errores, mensajes de entrada / salidas, mensajes de contenido web, mensajes de complementos

Valores y variables

```
var saludo = "Hola desarrolladores“;
console.log(saludo);
typeof(saludo);
```

Las variables son contenedores de valores, para definir una nueva variable se utiliza la palabra reservada “var”, los valores que puede tomar de forma atómica una variable tiene determinado tipo

Tipos de Variables

Las variables son contenedores de valores

Variable	Explicación	Ejemplo
String	Una cadena de texto. Para indicar que la variable es una cadena, debes escribirlo entre comillas.	var miVariable = 'Bob';
Number	Un número. Los números no tienen comillas.	var miVariable = 10;
Boolean	Tienen valor verdadero/falso. true/false son palabras especiales en JS, y no necesitan comillas.	var miVariable = true;
Array	Una estructura que te permite almacenar varios valores en una sola referencia.	var miVariable = [1,'Bob','Steve',10];
Object	Básicamente cualquier cosa. Todo en JavaScript es un objeto y puede ser almacenado en una variable. Mantén esto en mente mientras aprendes.	var miVariable = document.querySelector('h1');

Establecer el tipo de cada uno de los siguientes elementos

Ejercicio

undefined

null

true

8

NaN

“hola”

eval

{a:2, b:3}

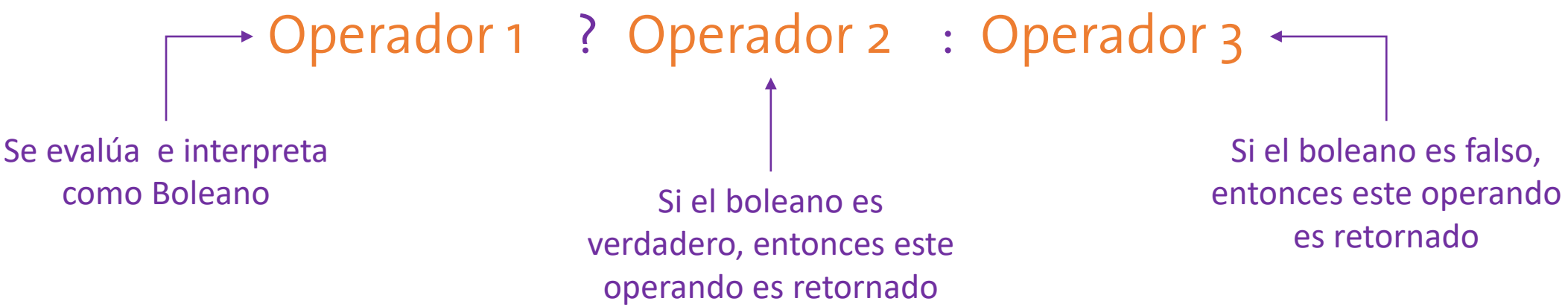
Asignación con operaciones

Operator	Example	Equivalent
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b
<<=	a <<= b	a = a << b

Operator	Example	Equivalent
>>=	a >>= b	a = a >> b
>>>=	a >>>= b	a = a >>> b
&=	a &= b	a = a & b
|=	a |= b	a = a | b
^=	a ^= b	a = a ^ b

Operador condicional (?:)

Único operador ternario (tres operandos).



Ejemplo

```
// Formas de ejecutar una sentencia condicionada al valor que toma una variable
saludo = "hola";
usuario = null
usuario ? saludo +=usuario : saludo += " usuario";

saludo = "hola";
if(usuario)
saludo += usuario;
else
saludo += " usuario";
```

Ejercicio: Crear una función que utilice el operador ternario que retorne el valor absoluto de un valor

Bloque de declaraciones

```
{
        x = Math.PI;
        cx = Math.cos(x);
        cx
}
```

Es la lista de instrucciones que deben ser ejecutadas por el navegador

Una expresión en una única línea termina con un punto y coma

Un bloque de declaraciones combina múltiples sentencias dentro de una sentencia compuesta.

Para crear un bloque de declaraciones se usan las llaves.

Funciones

```
function nombre(parametro1, parametro2, parametro3){
        // Bloque de ejecución
        return resultado // Puede ser opcional
}
```

```
// Forma 1: Crear una variable
var saludo = function(nombre){
        return "Hola " + nombre;
}

saludo("desarrollador")
```

```
// Forma 2: Declarar una función
function saludo(nombre){
        return "Hola " + nombre
}

saludo("desarrollador")
```

Condicionales

```
if(expresion){
        //Bloque de declaraciones
}
```

```
username = undefined
        if(username == null){
        username = "Onomishu";
}
```

```
n_ms = 2

if(n_ms ==1)
        console.log(“Un mensaje nuevo")
else
        console.log(n_ms + " mensajes nuevos")
```

Declaración de cambio

```
switch(expression) {
  case x:
    // Bloque de sentencias
    break;
  case y:
    // Bloque de sentencias
    break;
  default:
    // Bloque de sentencias
}
```

Se evalúa solo una vez, en donde el valor de la expresión se compara con cada uno de los casos

Si hay una coincidencia, se ejecuta el bloque de código asociado

En default se pone la parte del código que debería ser especificado en caso de no encontrar coincidencias

Los casos de cambio usan declaraciones estrictas (===) es decir debe coincidir tanto tipo como valor

Los casos de cambio usan declaraciones estrictas (===) es decir debe coincidir tanto tipo como valor

Ejemplo

```
var text;
var x = "10";
switch (x) {
  case 10:
    text = "Decena";
    break;
  default:
    text = "Sin valor";
}
```

Ejemplo

```
var saludo;
switch (2){
        case 1:
                saludo = "Buenos días";
                break;
        case 2:
                saludo = "Buenas tardes";
                break;
        case 3:
                saludo = "Buenas noches"
                break;
}
```

Declaración de cambio con bloques comunes

```
var text;
switch (new Date().getDay()){
        case 4:
        case 5:
                text = "Tan cerca del fds";
                break;
        case 6:
        case 0:
                text = "Los mejores del fds";
                break;
        default:
                text = "Lejos del fds"
}
```

Es posible hacer una declaración de cambio de forma que el cambio aplique para varios casos

Ejercicio: ¿Que hace el comando new Date().getDay()? Usar el resultado para hacer un saludo automático que incluya el día de la semana y la hora del día (mañana, tarde, noche)

Loops

Los bucles ejecutan partes de código varias veces

Tipos de loop en Javascript:

- while
- do/while
- for
- for/in

While

```
var count = 0;
while (count < 10){
        console.log(count);
        count++;
}
```

Ejecuta una sentencia siempre que una determinada condición sea verdadera.

El interprete ejecuta la declaración y la repite saltando de vuelta al inicio

Si la condición es infinitamente verdadera, el bucle nunca terminara y esto bloqueara el navegador

Do / While

```
do{
        // Bloque de declaraciones
}
while(condicion)
```

```
var a = [1, 2, 3, 4, 5, 6, 7, 8];
i = a.length;
do{
        console.log(a[i]);
        i--;
}
while(i > 0);
```

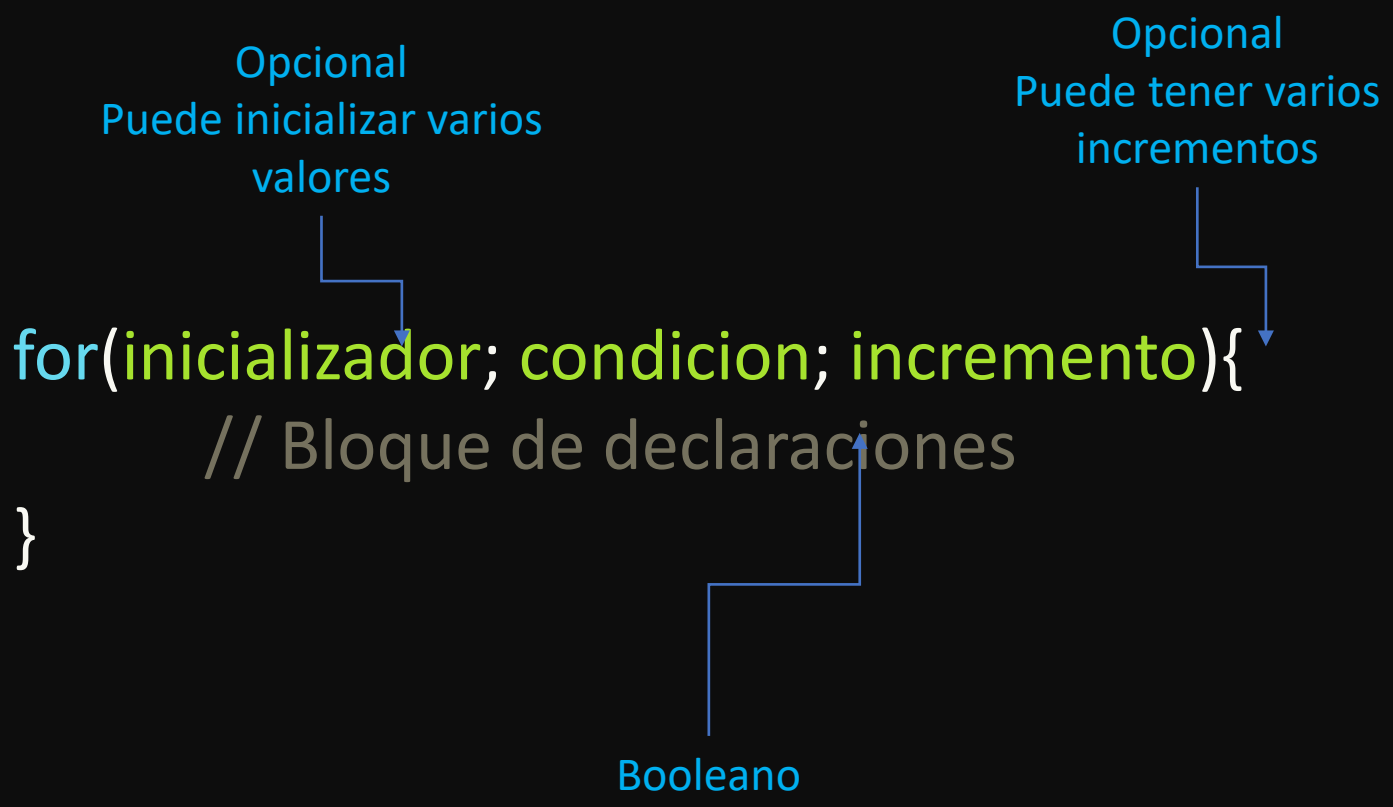
Funciona de la misma forma que el bucle while con la diferencia de que primero se ejecutan las sentencias antes de probar si la condición es verdadera

Ejercicio: ¿Qué hace el comando Math.random()?

- Amplifique ese número a un valor entero entre 0 y 100 usando el comando Math.round()

Escribir un código que imprima en la consola el número de aleatorios que deben ser generados hasta que aparece un número mayor a 80.

For



Es frecuentemente más utilizado que el bucle while.

Con el bucle for se realiza una tarea un número predeterminado de veces siguiendo un patrón.

Ejemplo

```
games = ["gta", "Doom", "TomR", "Ori", "Portal", "watch dogs"]

for(i = 0; i < games.length; i++){
        console.log("El juego " + i + " es " + games[i]);
}
```

Ejemplo

```
b = [1, 2, 3, 4, 5]
i = 0
for(; i < b.length ;){
        console.log(b[i]);
        i++;
}

var i,j;
var sum = 0;
for(i  = 0, j = 10; i < 10; i++, j--){
sum += i*j;
}
```

For /in

```
games_ob = {best : "gta",
                    worst : "Ghostbusters"};

var i;
text = ""
for(i in games_ob){
        console.log(games_ob[i])
}
```

Si queremos recorrer las propiedades de un objeto, usamos la declaración for/in (funciona para matrices y arrays)

Ejercicio: Con el ejercicio anterior cree un código que haga el promedio de aleatorios que deben ser generados hasta que se encuentra uno mayor a 80 basado en 1000 repeticiones

Saltos

Si queremos que el loop pare o que continúe ignorando parte del loop podemos usar las palabras break y continue para iniciar una nueva iteración

Break: Termina un bucle, igual que como en la declaración switch se puede usar en ciclos for o while

Continue: Salta sobre una iteración de un bucle que cumpla con una condición especifica

break

```
games = ["gta", "Doom", "TomR", "Ori", "Portal", "Dota"]

for(i in games){
        if(games[i] == "Ori"){
        break;
}
        console.log("Juego " + i + " " + games[i]);
}
```

break

```
games = ["gta", "Doom", "TomR", "Ori", "Portal", "Dota"]

i = 0
while(i < games.length){
        if(games[i] == "Ori"){
                break;
        }
        console.log("Juego " + i + " " + games[i]);
        i++;
}
```

Ejercicio: Suponga que apuesto 1000 pesos al número 17 en cada una de las rondas de una ruleta.

Cree un código que calcule cuánto perdí y cuántas veces perdí antes de ganar por primera vez utilizando el comando **break**

Apuestas	Números que juega	Beneficios
Suertes sencillas	18 números	1 a 1
Columnas y docenas	12 números	2 a 1
Seisena	6 números	5 a 1
Cuadro	4 números	8 a 1
Transversal	3 números	11 a 1
Caballo o semipleno	2 números	17 a 1
Pleno	1 número	35 a 1

Continue

```
games = ["gta", "Doom", "TomR", "Ori", "Portal", "Dota"]

for(i in games){
        if(games[i] == "Ori"){
                continue;
        }
        console.log("Juego " + i + " " + games[i]);
}
```

Continue

```
games = ["gta", "Doom", "TomR", "Ori", "Portal", "Dota"]

i = 0
while(i < games.length){
        if(games[i] == "Ori"){
                i++;
                continue;
        }
        console.log("Juego " + i + " " + games[i]);
        i++;
}
```

Ejercicio: Cree un código que calcule el porcentaje de números generados menores a 80 en 1000 repeticiones usando el comando continue.

Corrobore que sea alrededor del 80%.

Etiquetas Javascript

```
a = [1, 2, 3, NaN, 5]

sum = 0;
suma: if(a){
        for(i = 0; i < a.length; i++) {
                if(isNaN(a[i])) break suma;
                sum += a[i];
        }
}
```

Es posible etiquetar las declaraciones de JavaScript para saltar fuera de cualquier bloque de código y se usa en combinación con la declaración break

```
var m = [[1, 2, 3], [4, NaN, 6]];
var sum = 0, success = false;
calcular_suma: if (m) {
        for(var x = 0; x < m.length; x++) {
                var row = m[x];
                if (!row) break calcular_suma;
        for(var y = 0; y < row.length; y++) {
                var cell = row[y];
                if (isNaN(cell)) break calcular_suma;
                sum += cell;
                }
        }
        success = true;
}
```

Manejo de errores y excepciones

Inevitablemente, es posible que los códigos que desarrollemos requieran de condiciones sobre los valores de los parámetros sobre los que funcionan, problemas de codificación cometidos al momento de hacer el programa, entre otros.

Para manejar esto tenemos varias declaraciones:

Try
Catch
Throw
Finally

Manejo de errores y excepciones: throw

```
function factorial(x){
        if(x < 0) throw new Error("x no puede ser negativo");
        for(var f = 1; x > 1; f *= x, x--);
                return f
}
```

Una excepción indica que ha ocurrido algo inesperado dentro del código. La declaración throw permite devolver al usuario un mensaje de error entendible por un humano

Manejo de errores y excepciones: try + catch

```
function comparador(x){
        try{
                if(x == "") throw "vacio";
                if(isNaN(x)) throw "es una entrada no numérica";
                x = Number(x);
                if(x < 10) throw "muy pequeño";
                if(x > 20) throw "muy grande";
                return "Buen número";
        }
        catch(err){
                console.log("El número es " + err);
        }
}
```

Manejo de errores y excepciones: try + finally

```
function comparador(x){
        try{
                if(x == "") throw "vacio";
                if(isNaN(x)) throw "es una entrada no numérica";
                x = Number(x);
                if(x < 10) throw "muy pequeño";
                if(x > 20) throw "muy grande";
        }
        catch(err){
                console.log("El número es " + err);
        }
        finally{
                y = 2*x;
                return "Resultado " + y;
        }
}
```