JAVASCRIPT

Lección 1: Sentencias en JS

eip ESCUELA INTERNACIONAL POSGRADOS

ÍNDICE

Sentencias en JS

PRESENTACIÓN Y OBJETIVOS

En esta unidad se estudiarán las principales sentencias disponibles en JS.

Objetivos

En esta unidad aprenderás:

- ✓ **Que existen sentencias de flujo y de control**
- ✓ **Sobre sentencias de flujo: if ... else, switch**
- ✓ **Sobre sentencias de control: for, while, do ... while**

1 SENTENCIAS EN JS

Las sentencias son un grupo de palabras clave que se usan para controlar el flujo de un script o un programa.

Las sentencias se categorizan en:

Sentencias de control de flujo

- return – especifica el valor que devuelve una función
- break – detiene la instrucción actual y transfiere el control a la próxima instrucción
- continue – detiene la ejecución de la iteración actual y continua con la ejecución de la próxima iteración
- throw – lanza una excepción
- if … else
- switch

Sentencias de repeticiones

- for
- for … in
- for … of
- while
- do … while

En un script se pueden usar tantas sentencias como necesitemos. También algunas se pueden usar dentro de otras. Por ejemplo, un if dentro de un for, un break o continue dentro de un if.

** Presta atención que todas estas palabras claves están escritas en minúscula. Como JS es case sensitive, si las escribimos con mayúscula o mayúscula + minúscula, se va a generar un error.

A continuación, veremos en más detalle las sentencias más importantes y usadas.

2 IF ... ELSE

if ... else es una sentencia de control porque, dependiendo de si una condición se evalúa a true, se ejecuta un código u otro.

Sintaxis if ... else

```
if (condición) {
  // codigo que se va a ejecutar si la condición se evalúa a true
} else {
// codigo que se va a ejecutar si la condición se evalúa a false
}
```

Muy importante

- **Un if puede existir sin un else**
- **Un else, no existe sin un if**
- **Si después del if sigue una sola instrucción de código, las llaves {} no son requeridas; aunque se recomienda usarlas para mejor legibilidad del código**

A continuación, vamos a ver algunos ejemplos.

2.1 Ejemplos

Ejemplo if ... else

```
const age = 30
if (age > 65) {
  console.log("You should retire soon!")
} else {
  console.log("Keep working!")
}

// Keep working!
```

(*) Ejemplo sólo if

```
const age = 30
if (age > 65) {
  console.log("You should retire soon!")
}
```

(**) Ejemplo sólo if sin llaves {}

```
const age = 30
if (age > 65) console.log("You should retire soon!")
```

En los últimos 2 ejemplos (*) y (**), no se va a mostrar nada en la consola porque la condición no está cumplida y sólo añadimos instrucciones para el caso de true.

Dentro de ifs se pueden usar sin problema otros if. Es muy recomendado usar {}

2.2 ELSE IF

El else if se usa para especificar una nueva condición, cuando la primera se evalúa a false. Tiene que estar dentro de un if.

Se pueden añadir tantas sentencias else if como se necesiten.

** Presta atención que se escribe **else if**. No existe la palabra clave elseif en JavaScript.

Sintaxis else if

if (**condición1**) **{**
// código que se va a ejecutar si la condición 1 se evalúa a true
} else if (condición2) {
// código que se va a ejecutar si la condición 1 se evalúa a false y la condición 2 a true
} else {
// código que se va a ejecutar si la condición 1 y 2 se evalúan a false
}

Ejemplo if else if else

```
const age = 60
if (age < 30) {
  console.log("You just started working!")
} else if (age < 50) {
  console.log("Keep working!")
} else if (age < 65) {
  console.log("Almost done with work!")
} else {
  console.log("Go and relax, now!")
}

// Almost done with work!
```

Como se pueden añadir muchísimas else if dentro de la estructura de if, aun usando llaves {}, el código puede llegar a ser difícil de leer y seguir.

En esos casos, se recomienda ver si se puede usar la estructura **SWITCH**.

3 SWITCH

La instrucción switch evalúa una expresión y decide qué código ejecutar dependiendo de las cláusulas "**case**" existentes.

Sintaxis switch

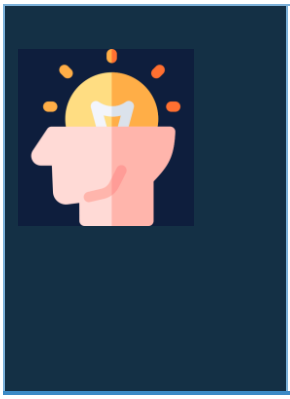
```
switch (expresión) {
  case valor1:
    // código que se ejecuta si el resultado de la expresión es valor1
    [break]
  case valor2:
    // código que se ejecuta si el resultado de la expresión es valor2
    [break]
  ...

  [default: valor1:
    // código que se ejecuta si el resultado de la expresión no coincide con ninguno de los valores dados en las cláusulas "case"
    [break]]
```

- **switch** – palabra clave
- **expresión** – la expresión que se va a evaluar
- **case** valorN – los valores de los posibles resultados de la expresión
- **default** – una cláusula que se ejecuta si el resultado de la expresión no coincide con ningún case; es opcional

Cómo funciona:

- Se evalúa (una sola vez) la expresión dentro del switch
- El valor de la expresión se compara con el valor de cada caso
- Si coincide con algún valor, se ejecuta el código asociado al respectivo case
- Si no coincide con ningún valor, se ejecuta el default; si la cláusula default no está presente, se continua con las instrucciones escritas después del switch



Importante

switch usa el operador de estrictamente igual **(===)**

Los valores que se comparan tienen que ser del mismo tipo, si no la comparación devolverá directamente un false.

3.1 Palabra clave – break

En el caso de switch, la palabra break es muy importante. Cuando JS encuentra la palabra break, la ejecución para y sale del bloque del switch.

Al final de las instrucciones de cada caso, se suele añadir break. En caso contrario, si hay una coincidencia y al final del código correspondiente no hay un break, se van a ejecutar todas las instrucciones que quedan (de otros casos que no coinciden).

3.2 Ejemplos

```
const fruit = "Mango"
switch (fruit) {
  case "Apple":
     console.log("We have apples");
     break;
  case "Mango":
     console.log("We have mangoes");
     break;
  case "Orange":
     console.log("We have oranges");
     break;
  default:
     console.log("I do not know what fruit this is");
}

// We have mangoes
```

En el caso de arriba, si no se añade la palabra clave **break** al final de la instrucción de cada caso, la consola muestra: "We have mangoes" / "We have oranges" / "I do not know what fruit this is".

4 FOR

La instrucción **for** crea un bucle que consiste en 3 expresiones opcionales, (dentro de unos paréntesis y separadas por puntos y comas), seguidos de una sentencia.

Sintaxis for

```
for ([expresión-inicial]; [condición]; [expresión-final])
{
  // código ejecutado en bucle
}
```

- **for** – palabra clave
- **expresión-inicial** – una expresión que se evalúa una vez antes de empezar a iterar; se suele usar para la declaración de variables auxiliares
- **condición** – una expresión que se evalúa antes de cada iteración y decide si continuar con la siguiente iteración
- **expresión-final** – una expresión que se evalúa al final de cada bucle; se suele usar para actualizar o incrementar una variable contador

Como funciona:

- La expresión inicial se ejecuta (una sola vez) antes de la ejecución del bloque de código
- Se evalúa la condición para ver si el bucle se va a ejecutar o no;
  - Si esta condición se evalúa en true, el código del bucle se ejecuta
  - Si la condición se evalúa en false, el código del bucle nunca llega a ejecutarse
  - Si esta condición se omite, la condición siempre se evalúa a true
- La expresión final se ejecuta (cada vez) después de que el código del bucle se ejecuta

4.1 Ejemplos

```
console.log("The first 10 numbers are: ")

for (let i = 0; i < 10; i++) {
   console.log(i)
}
```

```
let i = 0;

for (;;) {
 if (i > 3) break;
 console.log(i);
 i++;
}

// 0 1 2 3
```

En el segundo ejemplo podemos ver que un bucle **for** funciona también sin ningún parámetro (pero con ;; para separar el lugar de los parámetros).

En este caso, es necesario añadir dentro del bucle un break porque si no, tendremos un bucle infinito. Si no hay condición de parada, el bucle no se detiene.

4.2 FOR ... IN

Es una sentencia de repeticiones especial para objetos. Itera sobre todas las propiedades enumerables de un objeto.

Sintaxis for ... in

for (const variable **in** objeto) **statement**

Ejemplo

```
const person = {name: 'Roxana', age: 30}

for (const prop in person) {
  console.log("key: " + prop)
  console.log("value: " + person[prop])
}
```

En la consola se va a mostrar:

```
key: name
value: Roxana
key: age
value: 30
```

4.3 FOR ... OF

Itera sobre cualquier objeto iterable (String, Array, Map, etc).

Sintaxis for ... of

for (variable **of** objeto) **statement**

Ejemplo 1

```
const people = ['Roxana', 'Daniel', 'Ivan']

for (const person of people) {
  console.log(person)
}
```

Ejemplo 2

```
const person = 'Rox'

for (const letter of person) {
  console.log(letter)
}
```

En la consola se va a mostrar (para Ejemplo 1):

```
Roxana
Daniel
Ivan
```

En la consola se va a mostrar (para Ejemplo 2):

```
R
o
x
```

5 WHILE

La sentencia **while** crea un bucle que se ejecuta mientras la condición se evalúe a true.

Sintaxis while

```
while (condición) {
  // código
}
```

- **while** – palabra clave
- **condición** – la condición que se evalúa antes de cada iteración

Como funciona:

- la condición se evalúa
  - si se evalúa a true, el código del bucle se ejecuta
  - si se evalúa a false, el código del bucle nunca se ejecuta y se pasa al siguiente código después del while

5.1 Ejemplos

```
let i = 0;

while (i < 10) {
  console.log(i);
  i++;
}
// 0 1 2 3 4 5 6 7 8 9
```

* Cuidado con los bucles infinitos, el navegador se quedará bloqueado.

** Para salir de un bucle antes de que la condición se haya cumplido, usa break.

6 DO ... WHILE

La sentencia do ... while crea un bucle que se ejecuta hasta que la condición se evalúa en false. La diferencia de while es que la condición se evalúa después de ejecutar el código. De esta forma, el código llega a ejecutarse al menos una vez.

Sintaxis do ... while

do {
  // código
} **while** (**condición**)

- do – palabra clave
- while – palabra clave
- condición – la condición que se evalúa después de cada iteración

Como funciona:

- El código se ejecuta
- La condición se evalúa
  - Si se evalúa a true, el código se ejecuta nuevamente
  - Si se evalúa a false, se sale del bucle

6.1 Ejemplos

```
let i = 0

do {
  console.log(i);
  i++;
}
while (i < 10)

// 0 1 2 3 4 5 6 7 8 9
```

** Para salir de un bucle antes de que la condición se haya cumplido, usa break.

7 PUNTOS CLAVE

- if ... else y switch son sentencias de control de flujo
- Se puede usar if sin else, pero no else sin if
- Cuando necesitamos más puntos de decisión, se puede usar else if, dentro de un if
- Switch te deja definir varios casos de ejecución para los posibles valores de tu variable
- No te olvides de añadir break al final de cada caso. Si no, después de un match, en vez de salir de la ejecución del switch, pasa por todos los casos que quedan
- En el caso de la sentencia for, todos los parámetros son opcionales, pero hay que añadir los (;). También hay que usar un break para salir de ese bucle, si no será un bucle infinito
- While verifica primero la condición, y dependiendo de eso decide si ejecuta el código del bucle o no
- Do ... while ejecuta primero el código y después evalúa la condición; Por eso, en caso de do ... while, el código se ejecuta al menos una vez
- Cuidado con las condiciones en while, do ... while, y for. Es muy fácil llegar a crear un bucle infinito y bloquear el navegador