Condicionales JS

Los condicionales son una parte fundamental de cualquier lenguaje de programación y permiten ejecutar un bloque de código sólo si se cumple una determinada condición. En JavaScript, existen diferentes tipos de condicionales, cada uno con su propia sintaxis y usos específicos.

Condicional `if`

El condicional `if` es el más básico y se utiliza para ejecutar un bloque de código si se cumple una determinada condición. La sintaxis de un condicional `if` es la siguiente:

```
if (condición) {
  // código a ejecutar si se cumple la condición
}
```

Por ejemplo:

```
let age = 18;

if (age >= 18) {
  console.log('Eres mayor de edad');
}
```

En este caso, la condición `age >= 18` se cumple, por lo que se ejecuta el código dentro del bloque `if` y se imprime "Eres mayor de edad" en la consola.

Condicional `if...else`

El condicional `if...else` es una extensión del condicional `if` y se utiliza para ejecutar un bloque de código si se cumple una determinada condición y otro bloque de código si no se cumple. La sintaxis de un condicional `if...else` es la siguiente:

```
if (condición) {
  // código a ejecutar si se cumple la condición
} else {
  // código a ejecutar si no se cumple la condición
}
```

Por ejemplo:

```
let age = 17;

if (age >= 18) {
  console.log('Eres mayor de edad');
} else {
  console.log('Eres menor de edad');
}
```

En este caso, la condición `age >= 18` no se cumple, por lo que se ejecuta el código dentro del bloque `else` y se imprime "Eres menor de edad" en la consola.

Condicional `if...else if...else`

El condicional `if...else if...else` es otra extensión del condicional `if` y se utiliza para evaluar varias condiciones de manera secuencial. La sintaxis de un condicional `if...else if...else` es la siguiente:

```
if (condición1) {
  // código a ejecutar si se cumple la condición1
} else if (condición2) {
  // código a ejecutar si no se cumple la condición y se cump
le la segunda
} else {
    // código a ejecutar si ninguna de las anteriores condici
ones se cumple
}

// ejemplo:
if (age > 18) {
  console.log('Eres mayor de edad');
} else if (age === 18) {
  console.log('Tienes 18 años');
} else {
    console.log('Eres menor de edad')
}
```

Condicional `if`

La sentencia **if(...)** evalúa la condición en los paréntesis, y si el resultado es true ejecuta un bloque de código.

```
let bestJsWeb = 'Alberto';

if (bestJsWeb == 'Alberto') console.log('¡Woh, un fan!');
```

Nuestro mensaje de consola solo se imprime en caso de que la condición se cumpla, si el valor no coincidiese no se ejecutaría nunca.

```
let bestJsWeb = 'Peter';

if (bestJsWeb == 'Alberto') console.log('¡Woh, un fan!');
```

Ahora nuestro mensaje no se imprimirá porque no se cumple la condición planteada, así podemos ver cómo funciona la sentencia **if** o condicionales.

Respuestas Booleanas

Los booleanos nos van a permitir observar si una condición se cumple fácilmente, ya que su valor está comprendido en **true** o **false**. Es por ello que la sentencia **if (...)** evalúa la expresión dentro de sus paréntesis y convierte el resultado en booleano.

- El número **0**, un string vacío **""**, **null**, **undefined**, y **NaN** se convierte en **false**. Por esto son llamados valores **false**.
- El resto de los valores se convierten en **true**, entonces los llamaremos valores **verdadero**.

Entonces, el código bajo las condiciones **false** nunca se ejecutaría, y en las **true** lo hará.

```
if (0) { console.log('No me ejecuto') }
if (1) { console.log('Hola 👋 Students') }
```

Condicional `if else`

¿Qué sucede cuando queremos indicar que tome un camino y si la condición no se cumple tome otro? Sería algo recargado tener que realizar una doble condición, ya que obligamos a hacer dos cálculos a nuestra aplicación.

Para ello, en nuestra sentencia **if**, tenemos el bloque **else**, que es opcional. Este se ejecutaría cuando la condición es falsa.

```
let name = 'Antonio';

if (name == 'Antonio'){
    console.log('¡Woh, eres un fan de React!' );
} else {
    console.log('Ohhh, te gusta Angular :(')
}
```

Anidando varias condiciones

Podemos anidar también condiciones a través de un **else if (...)** de tal modo que en vez de ejecutar por defecto si la primera condición no se cumple vuelve a preguntar hasta que exista o encuentre la apropiada, en caso de no existir puede ejecutar el **else** que vimos antes.

```
let name = 'Antonio';

if (name == 'Antonio'){
    console.log('¡Woh, eres un fan de React!' );
} else if(name == 'Alberto') {
    console.log('¡Woh, eres un fan de Angular!');
}   else {
    console.log('ohhh, te gusta la programación? ');
}
```

Ternario `?`

En ocasiones queremos asignar un valor a una variable dependiendo de una condición.

```
let accessAllowed;

let age = 18;

if (age > 18) {
  accessAllowed = true;
} else {
  accessAllowed = false;
}

console.log(accessAllowed);
```

No está mal por por legibilidad desde MiniCode te recomendamos usar los ternarios que nos permite ejecutar esto en una forma más corta y simple.

El operador está representado por un signo de interrogación de cierre ?. A veces es llamado “ternario” porque el operador tiene tres operandos. Es el único operador de JavaScript que tiene esta cantidad de ellos.

```
let result = condition ? value1 : value2;
```

Se evalúa condition: si es verdadera entonces devuelve primer valor , de lo contrario segundo valor.

```
let accessAllowed = (age > 18) ? true : false;
```

Se pueden anidar tantos ternarios como queramos, pero desde nuestra experiencia no recomendamos su uso porque hace que nuestro código parezca más complejo que usando el **if else** tradicional.

```
let age = 18;

let message = (age < 3) ? '¡Wow cada día los developers son más
  (age < 18) ? '¡Bienvenido a MiniCode!' :
  (age < 100) ? 'Debe gustarte mucho el desarrollo' :
  '¡Meehh error en el sistema!';

console.log( message );
```

Por último los ternarios como hemos comentado se podían remplazar al if tradicional y os dejamos un pequeño ejemplo para verlo más claro.

```
let name = 'Antonio';

(name == 'Antonio') ?
    console.log('¡Fan de React!') : console.log('Ouch! no te gus
```

Condicional `Switch`

El condicional **switch** anida múltiples instrucciones de tipo if...else, este tipo de forma de expresión nos resultará útil al tener un número elevado de condiciones y aportan una mayor limpieza al código.

Veamos qué aporta este cambio de sintaxis frente a sentencias if else complejas:

```
const superhero = "Spider-Man";

if (superhero === "Spider-Man") {
  console.log("Su nombre real es Peter Parker");
} else if (superhero === "Daredevil") {
  console.log("Su nombre real es Matt Murdock");
} else if (superhero === "Iron Man") {
```

```
  console.log("Su nombre real es Tony Stark");
} else if (superhero === "Ant-Man") {
  console.log("Su nombre real es Scott Lang");
} else if (superhero === "Black Widow") {
  console.log("Su nombre real es Natasha Romanov");
} else {
  console.log("No hay superheroe");
}
```

```
const superhero = "Spider-Man";

switch(superhero) {
    case "Spider-Man":
        console.log("Su nombre real es Peter Parker");
    break;
    case "Daredevil":
        console.log("Su nombre real es Matt Murdock");
    break;
    case "Iron Man":
        console.log("Su nombre real es Tony Stark");
    break;
    case "Ant-Man":
        console.log("Su nombre real es Scott Lang");
    break;
    case "Black Widow":
        console.log("Su nombre real es Natasha Romanov");
    break;
    default:
        console.log("No hay superheroe");
        break;
}
```

La ejecución del código, en el caso del switch, revisará la igualdad estricta entre el case y el valor de la expresión insertada en el propio switch, por lo que

ejecutará el código dentro del case coincidente. Si ninguno de los casos coincide, ejecutará el código por defecto indicado en **default**.

Si las sentencias **break** no están en cada uno de los casos, la ejecución de código continuará automáticamente hasta encontrar la coincidencia. Si en nuestros casos tenemos un **return** los **break** no son necesarios, ya que el propio return haría que la ejecución de código termine.

Es importante recordar que switch siempre realizará la comparación con "estrictamente igual", por lo que en el siguiente caso ejecutará el default al no coincidir el tipo.

```
switch (6) {
  case "3":
    console.log("Número 3");
    break;
  case "6":
    console.log("Número 6");
    break;
  default:
    console.log("No hay número de tipo number");
    break;
}
```

Reto coding

1. Crea una variable llamada `character` y asígnale el valor "Legolas"
2. Crea una variable llamada `race` y asígnale el valor "Elfo"
3. Crea una variable llamada `hasRing` y asígnale el valor `false` (Legolas no tiene el Anillo Único)
4. Crea una variable llamada `isArcher` y asígnale el valor `true` (Legolas es un arquero)

5. Utiliza un condicional `if` para evaluar si `hasRing` es `true`. Si lo es, imprime en la consola el mensaje "Legolas es el portador del Anillo Único"
6. Utiliza `else if` para evaluar si `isArcher` es `true`. Si lo es, imprime en la consola el mensaje "Legolas es un arquero experto"
7. Utiliza `else` para imprimir en la consola el mensaje "Legolas es un guerrero valiente"