PROGRAMACIÓN BÁSICA EN JAVASCRIPT

Módulo 2

OTEC PLATAFORMA 5 CHILE

MÓDULO 2

CLASE 2

Introducción

Como segunda clase de esta introducción a JavaScript hablaremos de los condicionales.

De más está decir que en varias oportunidades usted se encontró con condicionales en distintas páginas web, pero nunca lo pensó como tal.
Como hemos dicho, es necesario que ahora piense desde el "cómo se hace".

Por lo tanto, a priori sabemos que las *Funciones Condicionales* de JavaScript son una herramienta que posibilita la toma de decisiones y permite realizar acciones de acuerdo a la entrada de información que reciba.

Con esto nos zambullimos de lleno en los if... else...

Introducción al uso de *condicionales* y *estructuras de control if*

La sentencia if... se usa, principalmente, para tomar decisiones. Permite que, si la condición es verdadera (true), se ejecute un código.

Sintaxis de la función if...

```
if (condición) {
// Si la condición resulta verdadera, ejecuta este código.
}
```

Sin embargo, la Función Condicional más común es if... else. Con esta, nos aseguramos que, cuando una condición se cumple (es igual a true), retornemos una cosa. Sino (si es false), retornemos otra.

```
if (condición) {
// Si la condición es true, ejecuta este código.
} else {
// Sino, ejecuta este otro código (la condición es false).
}
```

¿Qué tipo de datos son *true* y *false*?

true y false son datos de tipo *booleano*. Es lo que resulta de la comparación entre una condición y un input. Básicamente, determina si se cumple (o no) la condición para activar (o desactivar) cierta parte del programa.

Operadores de comparación

Para poder declarar una sentencia condicional necesitamos operadores que establezcan la relación entre ambas condiciones.

- **... < ... : Indica que la condición de la izquierda es menor que la de la derecha.**
- **... > ... : Indica que la condición de la izquierda es mayor que la de la derecha.**
- **... >= ... : Indica que la condición de la izquierda es mayor o igual que la de la derecha.**
- **... <= ... : Indica que la condición de la izquierda es menor o igual que la de la derecha.**
- **... == ...: Hace una comparación blanda entre dos valores. Es decir, JavaScript hace una coerción de datos, para que ambos sean del mismo tipo y pueda compararlos.**
- **...===... : Indica que la condición de la izquierda tiene una igualdad estricta respecto a la de la derecha. Es decir, evalúa que el contenido y el tipo de dato sea el mismo. Al usar este comparador evitarás bugs a futuro.**

El código se lee y ejecuta de arriba para abajo. Por lo tanto, el orden es muy importante a la hora de codear, tanto a la hora de declarar Variables como de generar las Estructuras Condicionales.

Las Estructuras Condicionales de tipo if...else pueden anidarse, unas dentro de otras, para:

- ➢ Generar múltiples bifurcaciones en función del objetivo del proyecto.
- ➢ Mostrar un único camino lógico a cada usuario.

por ejemplo...

```
if (edad >= 21) {
 alert("Puede pasar al bar.");

 let numeroSecreto = 10;
 let loQueDiceElUsuario = prompt("¿Cuál es el
número secreto?");

 if (loQueDiceElUsuario == numeroSecreto) {
   alert("Puede pasar a la fiesta.");
 } else {
   alert("No puede pasar a la fiesta, ese no
es el número secreto.");
 }
 } else {
 alert("No puede pasar al bar.");
}
```

Else... if

else... if es un recurso para poder anidar caminos intermedios entre el if y el else final. Una vez que se toma uno de los caminos, se completa el bloque lógico.

```
if(condicion1) {
// Si es true, se ejecuta este código.
} else if(condicion2) {
// Si es true, se ejecuta este código.
 } else {
// Sino, se ejecuta este código.
}
```

Siguiendo el ejemplo del bar, si un usuario puede pasar al bar si tiene 18 años, pero no puede tomar alcohol hasta ser mayor de edad a los 21, podríamos escribir el siguiente código:

```
let edad = prompt("Ingrese su edad.");

if (edad < 18) {
 alert("No puede pasar al bar.");
} else if (edad < 21) {
 alert("Puede pasar al bar, pero no
puede tomar alcohol.");
} else {
 alert("Puede pasar al bar y tomar
alcohol.");
}
```

Pseudocódigo

Una buena práctica a la hora de programar, sobre todo cuando las operaciones lógicas son complejas, es describir lo que debe hacer el programa en palabras y acciones simples. Estas indicaciones, generalmente escritas en código comentado, te ayudarán a idear un plan de acción que, luego, será convertido en código.

Dejar comentarios en el código es una de las prácticas más recomendadas a la hora de programar, sobre todo en proyectos escalables o colaborativos. Se usa para describir algo importante y para dejar indicaciones que puedan servir a futuro, en caso de que otro programador retome ese código.

Al usar ciertos símbolos, la doble barra (//) en el caso de comentar una única línea de código o encerrando comentarios multilínea entre barras y asteriscos (/* ... */), el navegador detecta que esa información no debe ejecutarse y la ignora.

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

/*
Este es un comentario multilínea
*/
```

Operadores lógicos y de *desigualdad* en JavaScript

En síntesis, todos los operadores lógicos y de desigualdad retornarán valores booleanos. La ventaja de usarlos es que permiten agrupar muchas condiciones y refactorizar el código, haciéndolo más rápido, legible y eficiente.

- **... || ... : Este operador, llamado "o", permite comparar un valor con 2, o más, condiciones. Para que la estructura dé como resultado true, alcanza con que solo 1 de ellas se cumpla. Si ninguna condición es true, la estructura será false.**
- **... && ... : Este operador, llamado "y", permite comparar un valor con 2, o más, condiciones. Para que la estructura dé como resultado true, todas las condiciones deben cumplirse. Basta con que 1 de las condiciones no se cumpla para que toda la estructura sea false .**
- **... != ... : Este operador, llamado "diferente de..." o "de desigualdad", permite comparar un valor con 2, o más, condiciones. Para que la estructura dé como resultado true, todas las condiciones deben ser diferentes entre sí. Basta con que 1 de las condiciones no sea diferente para que toda la estructura sea false.**

Para que || devuelva true, alcanza con que solo 1 sea verdadera.

```
true || true // La estructura es true.
true || false // La estructura es true.
false || true // La estructura es true.
false || false // La estructura es false.
```

Para que && devuelva true, todas las expresiones deben ser verdaderas.

```
true && true // La estructura es true.
true && false // La estructura es false.
false && true // La estructura es false.
false && false // La estructura es false.
```

Diferencia Blanda (!=) O Estricta (!==)

Al igual que con los operadores de comparación, la diferencia entre dos valores puede ser blanda o estricta. Mientras que, en la primera, se evaluará sólo el contenido, en la segunda se considerará tanto el contenido como el tipo de dato.

Importante: **los operadores de igualdad y de desigualdad estricta no intentan convertir los operandos a tipos compatibles antes de verificar la relación entre ellos.**

Operador de negación en JavaScript

El signo de exclamación ! niega el valor booleano de cada dato. Es decir que, si es true devolverá false y si es false, retornará true.

La naturaleza booleana de los datos

Los datos de JavaScript tienen una naturaleza intrínseca asociada a lo positivo o negativo. Por ejemplo, el número 0, al indicar ausencia de algo, tiene una naturaleza negativa. En cambio, el número 1, al indicar presencia, tiene una naturaleza positiva.

De la misma manera sucede con los Strings: uno vacío tendrá una naturaleza negativa mientras que uno lleno tendrá una positiva.

```
0 // false

1 // true

"" // false

"Indique su nombre" // true
```

Importante: **La doble negación de un dato (representada con !!) dará su equivalente booleano.**

Operador ternario

El Operador Ternario es una manera de simplificar las estructuras condicionales de tipo If...else para escribirlas en una sola línea.

Sintaxis De Los Operadores Ternarios:

Los Operadores Ternarios tienen una estructura con 3 Argumentos:

- **Una condición**
- **Un signo de interrogación (?)**
- **Los dos caminos posibles, separados por dos puntos (:)**

condición ? lo que se ejecuta si es true : lo que se ejecuta si es false.

Veamos un ejemplo:

```
let numeroDeTragos = 0;
numeroDeTragos > 0 //condición
 ? alert("Usted no puede manejar.") //ejecuta esto si es true
 : alert("Nos alegra que sea un conductor responsable."); //ejecuta esto si es false
```

MÓDULO 2

GRACIAS

Aquí finaliza la clase n°2 del módulo 2

OTEC PLATAFORMA 5 CHILE