



BOOTCAMP

Generación T

 streambe

Clase 27

Sentencia IF



[11 6133-1747](https://wa.me/5491161331747)



GENERACIÓN T

generaciont@generaciont.org

Introducción Al Uso De Condicionales Y Estructuras De Control if

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.

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

Introducción Al Uso De Condicionales Y Estructuras De Control if

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.

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

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

Else

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.

Sintaxis De La Función if...else:

```
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.

Else

Veamos un ejemplo. Si hiciéramos una app para evaluar a las personas que llegan a una guardia en un hospital, podríamos escribir el siguiente código que organice la atención de los pacientes en función de su urgencia:

```
let temperatura = prompt("Ingrese su temperatura.")  
if (temperatura < 37.5) { alert("Prioridad baja") } else { alert("Prioridad alta") }
```

<https://crubier.github.io/code-to-graph/?code=bGV0IHRIbXBpcmF0dXJhID0gcHJvbXB0KCJJbmdyZXNlIHN1IHRIbXBpcmF0dXJhLiIpDQppZiAodGVtcGVyYXR1cmEgPCAzNy41KSB7IGFsZXJ0KCJQcmVcmkYWQgYmFqYSIpIH0gZWxzZSB7IGFsZXJ0KCJQcmVcmkYWQgYWx0YSIpIH0>

Operadores De Comparación En JavaScript

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.

... != ... : 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 .

¿Qué Diferencia Hay Entre = Y ==?

El = asigna un valor. Es decir, es un operador de asignación.

Por ejemplo:

```
let edad  
edad = 17
```

En cambio, == compara que dos valores sean iguales.

Por ejemplo:

```
if (edad==18) {alert("Bienvenid@ a la mayoría de edad")  
}
```

Anidar if... else

Las Estructuras Condicionales de tipo if...else pueden anidarse para generar distintos caminos según el input del usuario.



Aspectos Importantes A Tener En Cuenta

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.

Anidar if... else

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

else... if

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.")
}
```

importancia de los comentarios

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.

importancia de los comentarios

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.

¿Para Qué Se Comenta El Código?

Simplemente para dejar registrado y documentado el por qué de ciertas decisiones a la hora de codear. Por ejemplo, qué datos acepta una función, por qué escribimos algo de una forma y no de otra o, simplemente, para dejar escrita una línea o bloque de código sin ser eliminado y que el browser no lo ejecute.

 Importante: Estas indicaciones quedarán expuestas en el código fuente de la web, por lo tanto, no deben contener información sensible ni comentarios desatinados ya que son accesibles a todos los que investiguen el código fuente de la página.

Operadores Lógicos Y De Desigualdad En JavaScript

Hay otros operadores en JS que te permitirán armar programas más complejos: los operadores lógicos y los de desigualdad.

or, and, not

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.

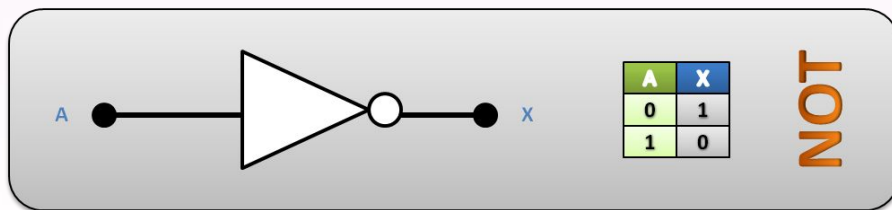
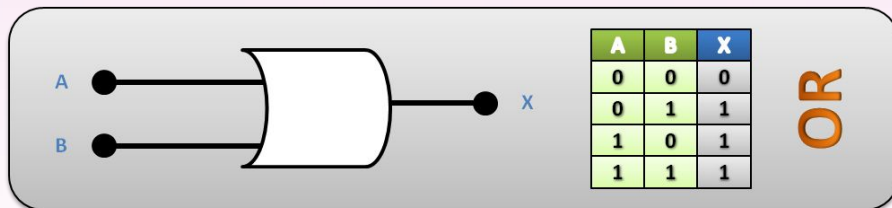
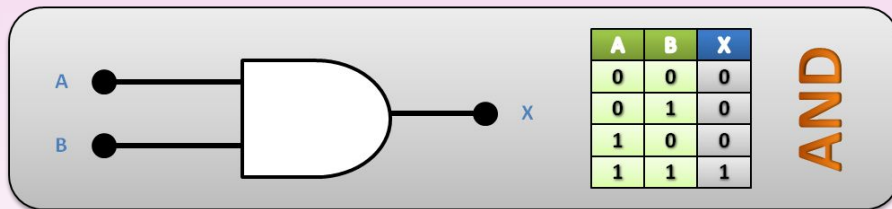
Operadores Lógicos

... || ... : 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 .

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

Operadores vistos como compuertas



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.

La Naturaleza Booleana De Los Datos

	false	true
Number ^{Int} _{float}	0	2 1.19 4 -9 0.01
String	" "	" " "a" " "
Boolean	False	true
NULL undefined	Siempre	



GENERACIÓN T