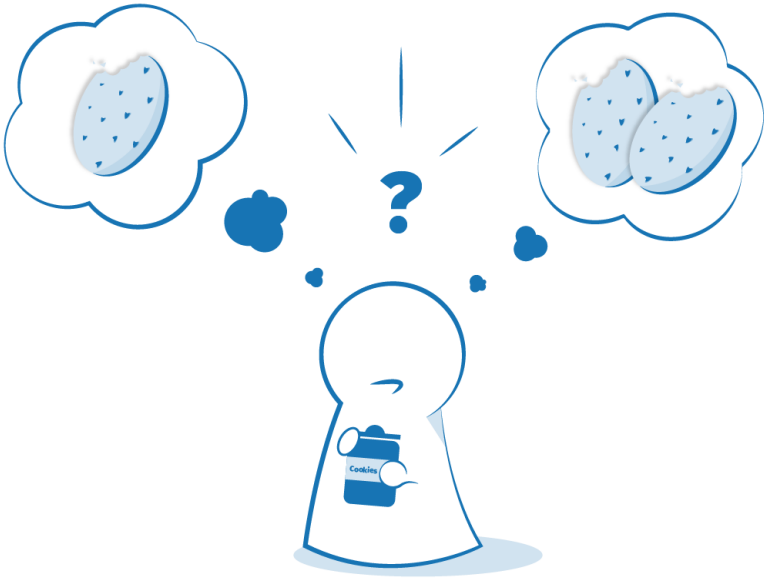
Tomando decisiones en tu código

Condicionales

Puedes hacerlo en una condición..!

Los seres humanos (y otros animales) toman decisiones todo el tiempo que afectan sus vidas, desde la más insignificante ("¿Debería comer una galleta o dos?") hasta la más grande (¿Debería quedarme en mi país y trabajar en la granja de mi padre, o debería mudarme a Estados Unidos y estudiar astrofísica?).



Declaraciones if ... else

Echemos un vistazo a la declaración condicional más común que usarás en JavaScript.

Sintaxis if ... else básica.

Una sintaxis básica if...else luce así.

```
if (condición) {
  código a ejecutar si la condición es verdadera
} else {
  ejecuta este otro código si la condición es falsa
}
```

Aquí tenemos:

1. La palabra clave if seguida de unos paréntesis.
2. Una condición a probar, puesta dentro de los paréntesis (típicamente "¿es este valor mayor que este otro valor?", o "¿existe este valor?"). Esta condición usará los **operadores de comparación** y retorna un valor true o false (verdadero o falso).
3. Un conjunto de llaves, en las cuales tenemos algún código — puede ser cualquier código que deseemos, código que se ejecutará sólamente si la condición retorna true.
4. La palabra clave else.
5. Otro conjunto de llaves, dentro de las cuales tendremos otro código — puede ser cualquier código que deseemos, y sólo se ejecutará si la condición no es true.

Este código es fácil de leer — está diciendo "**si (if)** la **condición** retorna verdadero (true), entonces ejecute el código A, **sino (else)** ejecute el código B"

Habrás notado que no tienes que incluir else y el segundo bloque de llaves — La siguiente declaración también es perfectamente válida.

```
if (condición) {
  ejecuta el código de al ser verdadera la condición
}

ejecuta otro código
```

Sin embargo, hay que ser cuidadosos — en este caso, el segundo bloque no es controlado por una declaración condicional, así que **siempre** se ejecutará, sin importar si la condicional devuelve true o false. Esto no es necesariamente algo malo, pero puede ser algo que no quieras — a menudo desearás ejecutar un bloque de código u otro, no ambos.

Como punto final, habrá ocasiones donde veas declaraciones if...else escritas sin un conjunto de llaves, de esta manera:

```
if (condición) ejecuta código de ser verdadero (true)
else ejecuta este otro código
```

Este código es perfectamente válido, pero no es recomendado usarlo — es mucho más fácil leer el código y determinar qué sucede haciendo uso de las llaves para delimitar los bloques de código y usar varias líneas y sangrías.

Un ejemplo real

Para comprender mejor la sintaxis, realicemos un ejemplo real. Imagínese a un niño a quien su madre o padre le pide ayuda con una tarea. El padre podría decir: "¡Hola, cariño! Si me ayudas yendo y haciendo las compras, te daré un subsidio adicional para que puedas pagar ese juguete que querías". En JavaScript, podríamos representar esto así:

```
let compraRealizada = false;

if (compraRealizada === true) {
  let subsidioAdicional = 10;
} else {
  let subsidioAdicional = 5;
}
```

La variable compraRealizada escrita en este código dará siempre como resultado un retorno de valor false, lo cuál significa una desilusión para nuestro pobre hijo. Depende de nosotros proporcionar un mecanismo para que el padre cambie el valor de la variable compraRealizada a true si el niño realizó la compra.

Nota: Podrás ver una versión más completa de este ejemplo en GitHub (también podrás verlo corriendo en vivo.)

else if

El último ejemplo nos proporcionó dos opciones o resultados, pero ¿qué ocurre si queremos más de dos?

Hay una forma de encadenar opciones / resultados adicionales extras a if...else — usando else if. Cada opción extra requiere un bloque adicional para poner en medio de bloque if() { ... } y else { ... } — Vea el siguiente ejemplo un poco más complicado, que podría ser parte de una aplicación para un simple pronóstico del tiempo:

```
<label for="clima">Seleccione el tipo de clima de hoy: </label>
<select id="clima">
  <option value="">--Haga una elección--</option>
  <option value="soleado">Soleado</option>
  <option value="lluvioso">Lluvioso</option>
  <option value="nevando">Nevando</option>
  <option value="nublado">Nublado</option>
</select>

<p></p>
let seleccionar = document.querySelector('select');
let parrafo = document.querySelector('p');
```

```
seleccionar.addEventListener('change', establecerClima);

function establecerClima() {
  let eleccion = seleccionar.value;

  if (eleccion === 'soleado') {
    parrafo.textContent = 'El día esta agradable y soleado hoy. ¡Use pantalones cortos! Ve a la playa o al parque y come un helado.';
  } else if (eleccion === 'lluvioso') {
    parrafo.textContent = 'Está lloviendo, tome un abrigo para lluvia y un paraguas, y no se quede por fuera mucho tiempo.';
  } else if (eleccion === 'nevando') {
    parrafo.textContent = 'Está nevando — ¡está congelando! Lo mejor es quedarse en casa con una taza caliente de chocolate, o hacer un muñeco de nieve.';
  } else if (eleccion === 'nublado') {
    parrafo.textContent = 'No está lloviendo, pero el cielo está gris y nublado; podría llover en cualquier momento, así que lleve un saco solo por si acaso.';
  } else {
    parrafo.textContent = '';
  }
}
```

1. Aquí tenemos un elemento HTML `<select>` que nos permite realizar varias elecciones sobre el clima, y un párrafo simple.
2. En el JavaScript, estamos almacenando una referencia para ambos elementos `<select>` y `<p>` , y añadiendo un Event Listener o en español un Detector de Eventos al elemento `<select>` así cuando su valor cambie se ejecuta la función `establecerClima()`.
3. Cuando la función es ejecutada, primero establecemos la variable `eleccion` con el valor obtenido del elemento `<select>`. Luego usamos una declaración condicional para mostrar distintos textos dentro del párrafo `<p>` dependiendo del valor de la variable `eleccion`. Note como todas las condicionales son probadas en los bloques `else if() {...}` , a excepción del primero, el cual es probado en el primer bloque `if() {...}`.
4. La última elección, dentro del bloque `else {...}`, es básicamente el "último recurso" como opción— El código dentro de este bloque se ejecutará si ninguna de las condiciones es `true`. En este caso, sirve para vaciar el contenido del párrafo si nada ha sido seleccionado, por ejemplo, si el usuario decide elegir de nuevo "--Haga una elección--" mostrado al inicio.

Nota: Puedes encontrar este ejemplo en GitHub (También podrás verlo correr en vivo.)

Una nota en los operadores de comparación

Los operadores de comparación son usados para probar las condiciones dentro de nuestra declaración condicional. Nuestras opciones son:

- `===` y `!==` — prueba si un valor es exactamente igual a otro, o sino es idéntico a otro valor.
- `<` y `>` — prueba si un valor es menor o mayor que otro.
- `<=` y `>=` — prueba si un valor es menor e igual o mayor e igual que otro.

Queremos hacer una mención especial al probar los valores (true/false) , y un patrón común que te encontrarás una y otra vez. Cualquier valor que no sea `false`, `undefined`, `null`, `0`, `NaN`, o una cadena vacía string (`''`) realmente retorna el valor `true` cuando es probada como una declaración condicional, por lo tanto puedes simplemente usar el nombre de una variable para probar si es `true`, o si al menos existe (i.e. no está definido.) Por ejemplo:

```
let queso = 'Cheddar';

if (queso) {
  console.log('¡Siii! Hay queso para hacer tostadas con queso.');
} else {
  console.log('No hay tostadas con queso para ti hoy :(.');
}
```

En el ejemplo anterior la variable `queso` contiene el valor 'Cheddar', y como su valor está definido o no es `false`, `undefined`, `null`, `0`, `NaN` y (`' '`) es considerado como `true` lo cual hará mostrar el mensaje dentro del primer bloque de llaves.

Y, devolviéndonos al ejemplo previo del niño haciendo las compras para su padre, lo podrías haber escrito así:

```
let compraRealizada = false;

if (compraRealizada) { //no necesitas especificar explícitamente '=== true'
   let subsidioAdicional = 10;
} else {
   let subsidioAdicional = 5;
}
```

Anidando if ... else

Está perfectamente permitido poner una declaración if...else dentro de otra declaración if...else — para anidarlas. Por ejemplo, podemos actualizar nuestra aplicación del clima para mostrar una serie de opciones dependiendo de cual sea la temperatura:

```
if (elección === 'soleado') {
  if (temperatura < 86) {
    parrafo.textContent = 'Está a ' + temperatura + ' grados afuera -- agradable y soleado. Vamos a la playa, o al parque, y comer helado.';
  } else if (temperatura >= 86) {
    parrafo.textContent = 'Está a ' + temperatura + ' grados afuera -- ¡QUÉ CALOR! Si deseas salir, asegúrate de aplicarte bloqueador solar.';
  }
}
```

Aunque el código funciona en conjunto, cada declaración if...else funciona completamente independiente del otro.

Operadores lógicos: AND, OR y NOT

Si quieres probar múltiples condiciones sin escribir declaraciones if...else anidados, los operadores lógicos pueden ayudarte. Cuando se usa en condiciones, los primeros dos hacen lo siguiente:

- && — **AND**; le permite encadenar dos o más expresiones para que todas ellas se tengan que evaluar individualmente true para que la expresión entera retorne true.
- || — **OR**; le permite encadenar dos o más expresiones para que una o más de ellas se tengan que evaluar individualmente true para que la expresión entera retorne true.

Para poner un ejemplo de **AND**, el anterior código puede ser reescrito de esta manera:

```
if (eleccion === 'soleado' && temperatura < 86) {
  parrafo.textContent = 'Está a ' + temperatura + ' grados afuera -- agradable y soleado. Vamos a la playa, o al parque, y comer helado.';
} else if (eleccion === 'soleado' && temperatura >= 86) {
  parrafo.textContent = 'Está a ' + temperatura + ' grados afuera -- ¡QUÉ CALOR! Si deseas salir, asegúrate de aplicarte bloqueador solar.';
}
```

Así que por ejemplo, el primer bloque sólo se ejecutará si la variable eleccion === 'soleado' *y* temperatura < 86 devuelven un valor verdadero o true.

Observemos un ejemplo rápido del operador **OR**:

```
if (carritoDeHelados || estadoDeLaCasa === 'en llamas') {
  console.log('Debes salir de la casa rápidamente.');
} else {
  console.log('Es mejor que te quedes dentro de casa');
}
```

El último tipo de operador lógico, **NOT**, es expresado con el operador **!**, puede ser usado para negar una expresión. Vamos a combinarlo con el operador **OR** en el siguiente ejemplo:

```
if (!(carritoDeHelados || estadoDeLaCasa === 'en llamas')) {
  console.log('Es mejor que te quedes dentro de casa');
} else {
  console.log('Debes salir de la casa rápidamente.');
}
```

En el anterior ejemplo, si las declaraciones del operador OR retornan un valor true, el operador NOT negará toda la expresión dentro de los paréntesis, por lo tanto retornará un valor false.

Puedes combinar los operadores que quieras dentro de las sentencias, en cualquier estructura. El siguiente ejemplo ejecuta el código dentro del condicional sólo si ambas sentencias OR devuelven verdadero, lo que significa que la instrucción general AND devolverá verdadero:

```
if ((x === 5 || y > 3 || z <= 10) && (logueado || nombreUsuario === 'Steve')) {
  // ejecuta el código
}
```

Un error común cuando se utiliza el operador OR en las declaraciones condicionales es intentar verificar el valor de la variable una sola vez, y luego darle una lista de valores que podrían retornar verdadero separados por operadores ||. Por ejemplo:

```
if (x === 5 || 7 || 10 || 20) {
  // ejecuta mi código
}
```

En este caso la condición `if(...)` siempre evaluará a verdadero siendo que 7 (u otro valor que no sea 0) siempre será verdadero. Esta condición lo que realmente está diciendo es que "if x es igual a 5, o 7 es verdadero— lo cual siempre es". ¡Esto no es lógicamente lo que queremos! Para hacer que esto funcione, tenemos que especificar una prueba completa para cada lado del operador OR:

```
if (x === 5 || x === 7 || x === 10 ||x === 20) {
  // ejecuta mi código
}
```

Declaraciones con switch

El condicional if...else hace un buen trabajo permitiéndonos realizar un buen código, pero esto viene con sus desventajas. Hay variedad de casos donde necesitarás realizar varias elecciones, y cada una requiere una cantidad razonable de código para ser ejecutado y/o sus condicionales son complejas (operadores lógicos múltiples).

Para los casos en los que solo se desea establecer una variable para una determinada opción de valores o imprimir una declaración particular dependiendo de una condición, la sintaxis puede ser un poco engorrosa, especialmente si se tiene una gran cantidad de opciones.

Para estos casos los switch son de gran ayuda — toman una sola expresión / valor como una entrada, y luego pasan a través de una serie de opciones hasta que encuentran una que coincida con ese valor, ejecutando el código correspondiente que va junto con ella. Aquí hay un pseudocódigo más para hacerte una idea:

```
switch (expresion) {
  case choice1:
    ejecuta este código
    break;

  case choice2:
    ejecuta este código
    break;

  // Se pueden incluir todos los casos que quieras

  default:
    por si acaso, corre este código
}
```

Aquí tenemos:

1. La palabra clave switch, seguida por un conjunto de paréntesis.
2. Una expresión o valor dentro de los paréntesis.
3. La palabra clave case, seguida de una elección con la expresión / valor que podría ser, seguido de dos puntos.
4. Algún código a correr si la elección coincide con la expresión.
5. Una declaración llamada break, seguida de un punto y coma.
   Si la elección previa coincide con la expresión / valor, el explorador dejará de

ejecutar el bloque de código aquí, y continuará a la siguiente línea de código. Si la opción anterior coincide con la expresión / valor, aquí el navegador deja de ejecutar el bloque de código y pasa a cualquier código que aparece debajo de la declaración de switch.

6. Como muchos otros casos, los que quieras.
7. La palabra clave default, seguido exactamente del mismo patrón de código que en los casos anteriores , excepto que el valor predeterminado no tiene opciones después de él, y no es necesario que se use break porque no hay nada que ejecutar después de este bloque de todas formas. Esta es la opción predeterminada o por defecto que se ejecuta si ninguna de las opciones coinciden.

Nota: No tiene que incluir la sección default; se puede omitir con seguridad si no hay posibilidades de que la expresión termine igualando un valor desconocido. Sin embargo, si existe la posibilidad de que esto ocurra, debe incluirlo para evitar casos desconocidos o comportamientos extraños en tu código.

Un ejemplo con switch

Echemos un vistazo a un ejemplo real: reescribiremos nuestra aplicación de pronóstico del tiempo para usar una declaración de cambio en su lugar:

```
<label for="weather">Select the weather type today: </label>
<select id="weather">
  <option value="">--Make a choice--</option>
  <option value="sunny">Sunny</option>
  <option value="rainy">Rainy</option>
  <option value="snowing">Snowing</option>
  <option value="overcast">Overcast</option>
</select>

<p></p>
let select = document.querySelector('select');
let para = document.querySelector('p');

select.addEventListener('change', setWeather);


function setWeather() {
  let choice = select.value;

  switch (choice) {
    case 'sunny':
      para.textContent = 'It is nice and sunny outside today. Wear shorts! Go to the beach, or the park, and
get an ice cream.';
      break;
```

```
    case 'rainy':
      para.textContent = 'Rain is falling outside; take a rain coat and a brolly, and don\'t stay out for too long.';
      break;
    case 'snowing':
      para.textContent = 'The snow is coming down -- it is freezing! Best to stay in with a cup of hot chocolate, or go build a snowman.';
      break;
    case 'overcast':
      para.textContent = 'It isn\'t raining, but the sky is grey and gloomy; it could turn any minute, so take a rain coat just in case.';
      break;
    default:
      para.textContent = '';
  }
}
```

Nota: También puedes encontrar este ejemplo en GitHub (también puedes verlo en ejecución aquí.)

Operador Ternario

Hay una última sintaxis que queremos presentarte antes de que juegues con algunos ejemplos. El **operador ternario o condicional** es una pequeña sintaxis que prueba una condición y devuelve un valor/expresión, si es true, y otro si es false — Esto puede ser útil en algunas situaciones, y puede ocupar mucho menos código que un bloque if...else si simplemente tienes dos opciones que se eligen a través de una condición true/false. El pseudocódigo se ve así:

```
( condición ) ? ejecuta este código : ejecuta este código en su lugar
```

Veamos un ejemplo simple:

```
let greeting = ( isBirthday ) ? 'Happy birthday Mrs. Smith -- we hope you have a great day!' : 'Good morning Mrs. Smith.';
```

Aquí tenemos una variable llamada isBirthday — si esta es true, le damos a nuestro invitado un mensaje de feliz cumpleaños; si no, le damos el saludo diario estándar.

Ejemplo con operador ternario

No sólo puedes establecer valores variables con el operador ternario; También puedes ejecutar funciones o líneas de código — lo que quieras. El siguiente ejemplo muestra un selector de tema simple donde el estilo del sitio se aplica utilizando un operador ternario.

```
<label for="theme">Select theme: </label>
<select id="theme">
  <option value="white">White</option>
  <option value="black">Black</option>
</select>

<h1>This is my website</h1>
let select = document.querySelector('select');
let html = document.querySelector('html');
document.body.style.padding = '10px';

function update(bgColor, textColor) {
  html.style.backgroundColor = bgColor;
  html.style.color = textColor;
}

select.onchange = function() {
  ( select.value === 'black' ) ? update('black','white') : update('white','black');
}
```

Aquí tenemos un elemento `<select>` para elegir un tema (blanco o negro), más un simple `<h1>` para mostrar el título de un sitio web. También tenemos una función llamada `update()`, que toma dos colores como parámetros (entradas). El color de fondo del sitio web se establece en el primer color proporcionado y el color del texto se establece en el segundo color proporcionado.

Finalmente, también tenemos un detector de eventos `onchange` que sirve para ejecutar una función que contiene un operador ternario.

Comienza con una condición de prueba — `select.value === 'black'`. Si esto devuelve `true`, ejecutamos la función `update()` con parámetros de blanco y negro, lo que significa que terminamos con un color de fondo negro y un color de texto blanco. Si devuelve `false`, ejecutamos las función `update()` con parámetros de blanco y negro, lo que significa que el color del sitio está invertido.

Nota: También puedes encontrar este ejemplo en GitHub (y verlo en ejecución aquí.)

Aprendizaje activo: Un calendario simple

En este ejemplo, nos ayudará a terminar una aplicación de calendario simple. En el código tienes:

- Un elemento `<select>` para permitir al usuario elegir entre diferentes meses.
- Un controlador de eventos `onchange` para detectar cuándo se cambia el valor seleccionado en el menú de `<select>`.
- Una función llamada `createCalendar()` que dibuja el calendario y muestra el mes correcto en el elemento `<h1>`.

Necesitamos que escriba una declaración condicional dentro de la función del controlador `onchange` justo debajo del comentario `// AGREGAR DECLARACIÓN DE CAMBIO:`. Debería:

1. Mire el mes seleccionado (almacenado en la variable `choice`. Este será el valor del elemento `<select>` después de que cambie el valor, por ejemplo "January")
2. Establezca una variable llamada `days` para que sea igual al número de días del mes seleccionado. Para hacer esto, tendrá que buscar el número de días en cada mes del año. Puede ignorar los años bisiestos a los efectos de este ejemplo.

Sugerencias:

- Se le aconseja que utilice el operador lógico **OR** para agrupar varios meses en una sola condición; Muchos de ellos comparten el mismo número de días.
- Piense qué número de días es el más común y utilícelo como valor predeterminado.

Si comete un error, siempre puede restablecer el ejemplo con el botón "Reset". Si se queda realmente atascado, presione "Show solution" para ver una solución.

Aprendizaje activo: ¡Más opciones de colores!

En este ejemplo, usaremos el ejemplo del operador ternario que vimos anteriormente y convertiremos el operador ternario en una declaración que nos permitirá aplicar más opciones.

Como podemos ver, el `<select>` no tiene dos opciones, sino cinco. Deberás agregar un `switch` justo debajo del comentario `// AGREGAR DECLARACIÓN DE CAMBIO`:

- Debe aceptar la variable `choice` como su expresión de entrada.
- Para cada caso, `choice` debe ser igual a cada una de las posibles opciones, (white, black, purple, yellow, or psychedelic). Debe tener en cuenta que los valores diferencian las minúsculas y mayúsculas. y deben ser iguales a los valores del elemento `<option>`.

- Para cada caso, la funcion update() debe ser ejecutada, y recibir dos valores de color, el primero es el color del fondo y el segundo el color del texto. Debes recordar que los colores son strings y deben estar entre comillas.