If, else, fallback

Recordando

Ya sabemos lo que es un if y ya sabemos lo que es un else. Por si te lo perdiste, la condición que evalúa un if termina siendo siempre **true** o **false.**

```
if ( true ) {
  Se ejecuta este código
}
```

```
if ( false ) {
  No se ejecuta este código
}
```

Por eso cuando ponemos la condición, en última instancia esa evaluación da un resultado que es true o false. Si tenemos algo como:

```
let edad = 20;
if (edad > 18) {
  console.log("Es mayor de edad");
} else {
  console.log("Es menor de edad");
}
```

En este caso evaluamos si edad es mayor a 18 `(edad > 18)` esta condición puede ser o true o false, por lo que nos deja con algo como

```
if ( true ) {
  Se ejecuta este código
}
```

Else if

Muchas veces nuestro programa se va a volver complejo y vamos a tener que hacer muchas preguntas, no va a ser suficiente solo con un if-else. Pensemos un ejemplo en un juego con privilegios:

- Si el usuario tiene el código “USER”, accede a los beneficios básicos del juego.
- Si el usuario tiene el código “SUPERUSER”, accede a mayores beneficios del juego.
- Si el usuario tiene el código “FULLUSER”, accede a todas las funcionalidades del juego.

Este programa se podría resolver con tres preguntas:

```
const code = "USER";
if(code === "USER") {
  console.log("Tiene beneficios en el juego");
} else if (code === "SUPERUSER") {
  console.log("Tiene los beneficios del USER más nuevos beneficios");
} else if (code === "FULLUSER") {
  console.log("Tiene todos los beneficios del juego");
}
```

1. Si el code es igual a USER, tiene beneficios en el juego
2. Si el code es igual a SUPERUSER, tiene los beneficios del USER más nuevos beneficios.
3. Si el code es FULLUSER, tiene todos los beneficios del juego.

Utilizando la palabra **else if,** podemos comenzar a hacer muchas más preguntas y que nuestro programa solucione problemas más complejos.

Fallback

En programación hay un término que se denomina “fallback” que hace alusión a la última opción posible. Supongamos que tenemos un programa que evalúa ciertas condiciones y si no se cumplen ninguna de estas, se ejecuta una última… esta última se le denomina fallback. Siempre es la última opción.

Un usuario ingresa un código para desbloquear un celular:

1. Si ingresa mal el código, le doy otra oportunidad
2. Si vuelve ingresar mal el código, le doy otra oportunidad
3. Si ingresa el código mal una vez más, le digo que espere 5 minutos
4. Si lo vuelve a ingresar mal, le bloqueo el teléfono hasta que activen el modo recuperación.

En este caso, el fallback siempre es el último caso, la última opción, a esa que no querríamos llegar pero tiene que estar como última posibilidad. Este ya famoso “fallback” se condice con el último **if** de las condiciones.

```
if(intentoErroneo === 1) {
  console.log("Pruebe de nuevo");
} else if (intentoErroneo === 2) {
  console.log("Pruebe una vez más");
} else if (intentoErroneo === 3) {
  console.log("Espere 5 minutos e intente de nuevo");
} else {
  console.log("El teléfono ha sido bloqueado, utilice algún modo de recup
eración");
}
```

Sea cual sea el programa que estemos creando, siempre es bueno hacerse las siguientes preguntas:

1. La condición que estoy evaluando, ¿necesita un fallback?
2. ¿Qué pasa si no se cumple ninguna de las condiciones que estoy evaluando?
3. ¿Cuál es el comportamiento por defecto?

Y en base a estas preguntas poder determinar si nuestra evaluación necesita o no un fallback.