LA PROGRAMACIÓN: INICIATE EN UN MUNDO APASIONANTE

# 2. ESTRUCTURAS DE CONTROL

Las estructuras de control nos sirven para controlar qué instrucciones de nuestro programa queremos ejecutar y en qué orden. Vamos a ver dos tipos: estructuras de selección y de iteración.

## Estructuras de selección

Hasta ahora hemos visto instrucciones en un orden secuencial, es decir, una detrás de otra. Por ejemplo:

```
var carta = '1c'
var valor = carta[0]
var palo = carta[1]
```

Pero habrá veces que queramos ejecutar instrucciones sólo bajo algunas condiciones. Por ejemplo, volviendo a nuestra baraja de cartas, querremos llevar al montón de cartas de corazones una carta sólo si es de corazones.

Para eso los lenguajes de programación utilizan las estructuras de selección. Usándolas, podemos indicar las instrucciones a ejecutar dada una determinada condición lógica. Una condición lógica es una pregunta cuya respuesta es verdadero o falso. Por ejemplo, tenemos nuestra baraja, sacamos una carta y hacemos la pregunta: ¿es de picas o treboles? En caso afirmativo (`if`), llevamos la carta al montón de carta negras. En caso contrario (`else`), llevamos la carta al montón de cartas rojas.

Además, estas preguntas lógicas que hacemos con `if / else` pueden anidarse. Por ejemplo, dada una carta de la baraja, pregunto si es de picas. En caso afirmativo, la paso al montón de picas. En caso contrario, hago una nueva pregunta: ¿es de corazones? En caso afirmativo, la llevo al montón de corazones. En caso contrario hago una nueva pregunta, y así sucesivamente hasta que compruebo si es de un palo determinado.

**Ejemplo 1: if**

```
var carta = "3c";
if (carta[1] === "c"){
  console.log("Es un 3 de corazones");
}
```

Comenzamos la estructura de selección con la palabra `if` seguida de la condición lógica (pregunta) entre paréntesis y luego un bloque de código entre `{` y `}` que será lo que nuestro programa ejecute en caso de cumplirse la condición lógica.

Nota: Usamos `console.log` para escribir texto y datos en la consola JavaScript del navegador.

Nota: Como estamos escribiendo varias sentencias en nuestro programa, una buena práctica en JavaScript es escribir un punto y coma (`;`) al final de cada sentencia. En nuestro ejemplo, las líneas con `if/else` son estructuras de selección no son sentencias y por tanto no llevan "`;`" al final.

Nota: Indentar consiste en poner espacios a la izquierda del código, de forma que cuando anidamos un bloque (código entre `{` y `}`) queda visualmente más a la derecha. De esta forma, veremos visualmente el nivel de anidamiento de nuestro código. Así, nuestro código será más legible y ordenado.

**Ejemplo 2: else**

```
var carta = "3c";
if (carta[1] === "c"){
  console.log("Es un 3 de corazones");
} else {
  console.log("No es un 3 de corazones");
}
```

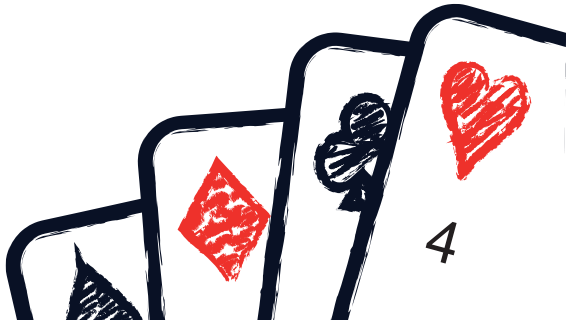
Como el ejemplo anterior, pero en este caso usamos la palabra `else` para indicar un bloque de código a ejecutar en caso de que no se cumpla la condición del `if`.

**Ejemplo 3: anidamiento**

```
var carta = "3p";
if (carta[1] === "c"){
  console.log("Es un 3 de corazones");
} else {
  if(carta[1] === "p"){
    console.log("Es un 3 de picas");
  }
}
```

Podemos anidar, es decir, encadenar unas preguntas dentro de otras.

Nota: Cuando no formamos bien el código, la consola de JavaScript de nuestro navegador nos dará un error de sintaxis (`syntax error`). Para evitarlo, tenemos que colocar las llaves y los paréntesis en su lugar adecuado.



# Estructuras de iteración

Son también llamadas de repetición o bucles. Nos permiten repetir la ejecución de un bloque de código tantas veces como queramos, hasta que se cumpla una condición lógica, que en este caso llamamos condición de parada.

Por ejemplo, si vamos a nuestro montón de cartas, sacamos una carta y preguntamos si es el uno de corazones. En caso contrario, tomamos una nueva carta y comprobamos si es el uno de corazones. Así de forma continua hasta que lo encontramos, cumpliéndose nuestra condición de parada.

La estructura de iteración más común es el bucle for, que consta de 3 parámetros entre paréntesis y el bloque de código a repetir. El primer parámetro es un índice que declaramos para controlar las iteraciones del bucle; el segundo es la condición de parada; y el tercero es la actualización, que es una instrucción que ejecuta nuestro programa al final del bloque de código.

**Ejemplo 1: for**

```
for (var i=1; i<= 12; i=i+1){
  console.log(i + " de picas");
}
```

En este bucle `for` iteramos con el índice i desde el valor 1 hasta el 12, aumentando +1 en cada iteración. En el bloque de código imprimimos con `console.log` el valor de `i` seguido de la cadena " de picas", de forma que como resultado tenemos "1 de picas", "2 de picas", y así sucesivamente hasta 12.

Ejemplo 2: do while

```
var i = 1;
do {
  console.log( i + " de picas");
  i = i + 1;
}while(i <= 12);
```

En este caso hemos construido un bucle do/while con la misma funcionalidad que el for del ejemplo anterior. En este caso, tenemos que incluir la inicialización del índice al comienzo del programa. Y también el bloque de actualización ponerlo explícitamente como última sentencia del bloque a repetir.