

Bienvenidos a fundamentos de programacion, capitulo 2!

Antes de arrancar, hagamos un breve repaso

En la clase pasada dimos:

- Logica
- Condicionales
- Datos
- Sintaxis basica de JavaScript (if, else, var..)

Ahora si, hoy vamos a ver **estructuras de control**

- Vamos a presentar un problema
- Estructura repetitivas, al rescate
- Solución al problema del contador con estructura repetitiva for
- Solución al problema del contador con estructura repetitiva while
- Que cuidados hay que tener con las estructuras repetitivas
- Ejercicios

El problema:

- Tenemos una tarea: incrementar una variable en 1 unidad
- Supongamos que necesitamos hacer esa misma tarea un numero determinado de veces ('X' veces)
- Ese numero 'X' de veces debe variar (Puede ser 10 hoy, y mañana 100)
- Diseñar un programa que solucione el problema

Modelando la solución

- Sea el caso que la tarea se deba repetir 5 veces

Primero: Definir la variable

```
var contador = 0;
```

Segundo: Incrementamos 5 veces la variable contador

```
var contador = 0;  
contador = contador + 1;  
contador = contador + 1;  
contador = contador + 1;  
contador = contador + 1;  
contador = contador + 1;  
console.log(contador); // 5
```

... pero si debemos hacer la tarea 100 veces?

Estructuras repetitivas, al rescate

A nuestro pseudocodigo que creamos la clase pasada, le vamos a agregar una nueva funcion,

la de repetir un bloque de codigo X veces

```
x = 100; // cantidad de veces que debemos repetir  
contador = 0;
```

```
repetir (x) {  
    contador = contador + 1;  
    informar(contador)  
}
```

el pseudocodigo anterior imprimiria:

1

2

3

4

...

Estructura repetitiva FOR:

Traduzcamos el 'repetir' a la sintaxis de JavaScript. Y esto va a ser algo complicado

```
var x = 10;
```

```
for (var i = 0; i < x; i++) {  
    console.log(i);  
}
```

El programa imprime:

```
// 0
```

```
// 1
```

```
// 2
```

```
// 3
```

```
// 4
```

```
// 5
```

```
// 6
```

```
// 7
```

```
// 8
```

```
// 9
```

Pero y todo lo demas dentro del for ?!

```
var x = 10;
```

```
for (var i = 0; i < x; i++) {  
    console.log(i);  
}
```

➔ Antes de empezar a repetir, definimos la variable auxiliar, esta variable indica el numero de la repeticion (de 0, a 9)

➔ Antes de cada repeticion, realizamos esta comparacion. Si esta comparacion resulta verdadera repetimos, en caso contrario se detiene la estructura repetitiva
La primera repeticion se compara $0 < 10$

➔ Luego de cada repeticion se realiza esta operacion.
El operador `++` aumenta en **1** el valor de la variable `i`
Es lo mismo que hacer
`i = i + 1;`

Veamos un par de ejemplos mas

Para eso, abramos [CodeSandbox](#) y juguemos un rato con la estructura for

Estructuras repetitivas, al rescate

Ahora, solucionemos el problema del contador con nuestra nueva herramienta

Antes de ver la solución, resolvámoslo en [CodeSandbox](#).

Solucion al problema con la estructura repetitiva FOR

```
var x = 100;
var contador = 0;

for (var i = 0; i < x; i++) {
  contador = contador + 1;
}

console.log(contador); // 100
```

Otra solucion al problema, con otra estructura repetitiva

Hasta ahora resolvimos el contador con una estructura **for**, veamos otra solucion

Otra vez, a nuestro pseudocodigo le vamos a agregar una nueva funcionalidad, la estructura repetitiva **mientras**

```
x = 10;  
contador = 0;  
  
mientras (contador < x) {      // mientras 'contador' sea menor qu  
  
    contador = contador + 1  
  
}
```

Esta estructura 'mientras' a diferencia del 'repetir' repetira el bloque de codigo siempre que la condicion sea verdadera y no necesita una variable auxiliar. En este caso la condicion es (contador < x)

Estructura repetitiva WHILE:

Traduzcamos el 'mientras' a la sintaxis de JavaScript.

```
var x = 10;
var contador = 0;

while (contador < x){
  console.log(contador)
  contador = contador + 1;
}

console.log('Final')
console.log(contador)
```

El programa imprime:

```
// 0
// 1
// 2
// 3
// 4
// 5
// 6
// 7
// 8
// 9
// Final
// 10
```

Veamos un par de ejemplos mas

Para eso, abramos [CodeSandbox](#) y juguemos un rato con la estructura for

Estructuras repetitivas, al rescate

Ahora, solucionemos el mismo problema pero con la estructura WHILE

Antes de ver la solución, resolvámoslo en [CodeSandbox](#).

Solucion al problema con la estructura repetitiva WHILE

```
var x = 100;  
var contador = 0;  
  
while (contador < x) {  
    contador = contador + 1;  
}  
  
console.log(contador) // 100
```

Cuidado con estas estructuras

- Debemos tener cuidado de no hacer muchas repeticiones
- Debemos tener cuidado de no crear loops infinitos

No intenten esto en casa

```
while (true) {  
    console.log('Esto no parece una buena idea');  
}
```

Antes de saltar a los ejercicios, veamos como utilizar el operador '%'

Este operador nos devuelve el resto de la division de dos numeros:

```
var resto = 9 % 2;    //    (4 * 2 = 8) => (9 - 8 = 1)
console.log(resto)    //    1
```

```
var resto2 = 13 % 5 //    (5 * 2 = 10) => (13 - 10 = 3)
console.log(resto2)  //    3
```

```
var resto2 = 8 % 3 //    (2 * 3 = 6) => (8 - 6 = 2)
console.log(resto2)  //    2
```

Reconocer numeros pares con el operador '%'

```
var esNuevePar = 9 % 2 === 0;  
console.log('El numero 9 es par? ', esNuevePar);  
  
var esCuatroPar = 4 % 2 === 0;  
console.log('El numero 4 es par? ', esCuatroPar);
```

Ejercicios

1) Pares

Sea una variable numerica entera y positiva 'limite':

- Recorrer desde 0 hasta `limite`
- Imprimir en pantalla los numeros pares (No la cantidad de nume

2) Contador

Sea una variable numerica entera y positiva '`limite`':

- Recorrer desde 0 hasta `limite`
- Imprimir al final del programa la cantidad de numeros impares
- Imprimir al final del programa la cantidad de numeros menores de `limite / 2`

3) Contador 2: La venganza del contador

Sea una variable numerica entera y positiva '`limite`':

- Recorrer desde 0 hasta que se cumpla una de las siguientes condiciones:
 - Se llega a `limite`
 - La cantidad de numeros pares desde 0 hasta `limite` es mayor a una variable anteriormente creada llamada `final`
- Al final del recorrido imprimir la cantidad de numeros multiplos de 3
- Al final del recorrido imprimir la suma de todos los numeros entre el 0 hasta que se termine el recorrido

4) La secuencia de Fibonacci

En matemáticas, la sucesión o serie de Fibonacci es la siguiente sucesión infinita de números naturales:

0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21

La sucesión comienza con los números 0 y 1, a partir de estos cada término es la suma de sus dos anteriores

Sea una variable numerica entera y positiva 'limite':

- Recorrer desde 0 hasta `limite`
- Imprimir tantos terminos de la secuencia como repeticions de 0 hasta `limite`

Ejemplo: si `limite = 6`, imprimir 0, 1, 1, 2, 3, 5

Fuentes

- <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Statements/while>
- <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Statements/for>
- https://es.wikipedia.org/wiki/Sucesi%C3%B3n_de_Fibonacci