TEXT CLASS REVIEW

TEMAS A TRATAR EN LA CUE

- Bucles.
- Try Catch.
- Array.
- Objetos.

Cuando estamos escribiendo el código de nuestro software, en muchas ocasiones nos veremos en la necesidad de que, dependiendo de cierta condición, cierta porción de código se repita, lo que para el usuario se vería reflejado en repetir una acción. ¿Cuándo, por ejemplo? Imaginemos que estamos ingresando a un sitio en el cual queremos registrarnos y, como indicación, nos piden una contraseña que cumpla con ciertas características específicas. Podemos equivocarnos y no completar todos los requisitos, por lo que veremos que la página nos muestra que está procesando la información enviada y nos volverá a solicitar una y otra vez la información requerida hasta que completemos los requisitos.

Eso es solo un ejemplo, pero existen múltiples situaciones en las que podríamos repetir un ciclo, como, por ejemplo, cuando en un cajero automático elegimos seguir navegando y no salir.

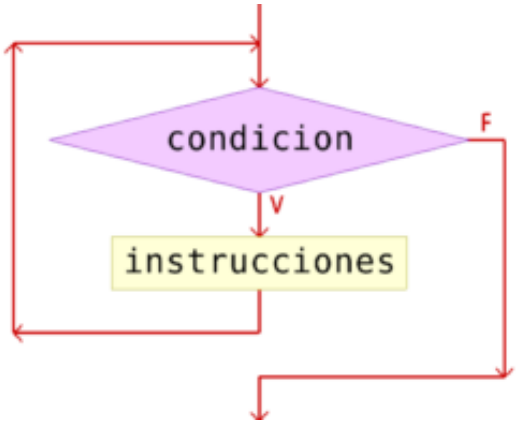
Podríamos pensar que simplemente escribimos el código una y otra vez hasta alcanzar todas las ocasiones en que un usuario necesitará repetir cierta información, pero, no es así. Basta con escribir el bloque de instrucciones que se repetirá en una sola ocasión y utilizar una sentencia cíclica de control de flujo o Bucle de iteración.

QUÉ ES UN BUCLE DE ITERQACIÓN

Los bucles de iteraciones (podemos nombrarlos sentencias cíclicas de control de flujo o ciclos iterativos) son una forma rápida y sencilla de hacer algo repetitivamente.

Existen distintos tipos de bucles, pero, en esencia, todos realizan la misma función: repetir una acción varias veces en consecuencia de la evaluación de una condición especifica. Como evalúan cierta

condición para saber cuántas veces repetir un ciclo, se puede dar la posibilidad de que jamás se ingrese a un ciclo y no se muestre jamás cierto código o que se repita infinitas veces (bucle infinito).



Las declaraciones para bucles en JavaScript son muchas, pero nos centraremos en **for**, **while** y **do while**.

FOR

For es un ciclo que se repite hasta que una condición especifica se evalúe como falsa. La declaración de un ciclo **for** tiene el siguiente aspecto:

```
for ([expresiónInicial]; [expresiónCondicional]; [expresiónDeActualización])
  instrucción
```

For in

Dentro de las instrucciones **for** podemos encontrar **for in,** que itera una variable específica sobre todas las propiedades enumerables de un objeto. Sigue el siguiente aspecto:

```
for (variable in objeto)
  instrucción
```

For of

Bucle que se repite sobre objetos iterables (hablaremos del concepto de objeto en un momento). Su aspecto es el siguiente:

```
for (variable of objeto)
  instrucción
```

Un objeto es una colección de datos relacionadas. Generalmente los objetos cuentan con variables y funciones, que se denominan propiedades y métodos cuando están dentro de objetos.

WHILE

El bucle iterativo **while** evalúa una condición y siempre que esa condición se siga considerando true (verdadera) seguirá ejecutando las instrucciones. El aspecto es el siguiente:

```
while (condición)
  expresión
```

DO WHILE

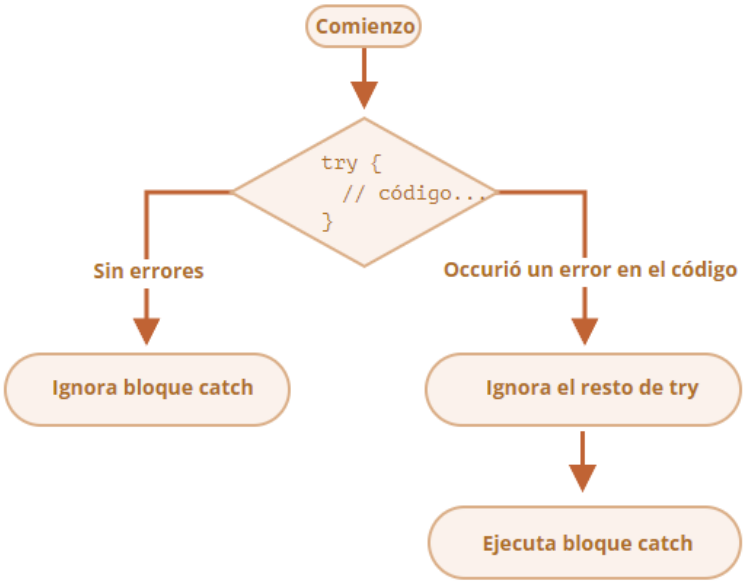
Esta instrucción repite un bloque de instrucciones hasta que una condición especifica se evalúe como falsa. Siempre se evalúa la condición después de las instrucciones.

```
do
  expresión
while (condición);
```

MANEJO DE EXCEPCIONES

Dentro de un flujo normal siempre habrá situaciones propicias para que ocurran excepciones tales que lleven a un error de código y permitan que el programa "se caiga". Para evitar ese tipo de situaciones existen instrucciones que permiten detectar errores, como **Try Catch**.

- **Try** es la declaración que permite probar el bloque de código en busca de errores.
- **Catch** es la declaración que permite manejar el error.
- **Throw** es la declaración que permite crear errores personalizados.
- **Finally** es la declaración que permite ejecutar código, después de intentar y capturar errores, independiente del resultado (independiente de si ocurrió un error o no).

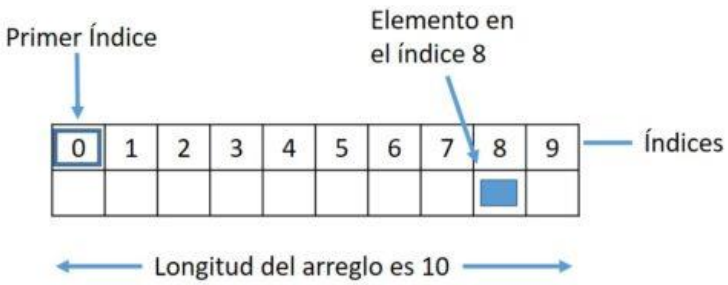


Los errores que pueden ocurrir pueden ser errores de codificación, errores de entrada incorrectas (datos que se capturan por algún lugar, por ejemplo, enviados por el usuario) y cualquier cosa imprevisible.

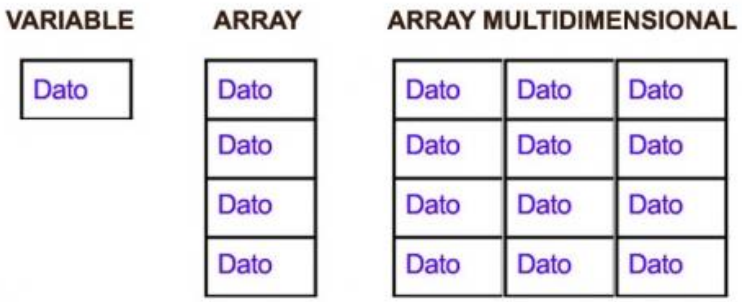
QUE ES ARRAY

Los **Array** (o arreglos) son objetos similares a una lista cuyo prototipo proporciona métodos para realizar operaciones de recorrido y de mutación.

Los **Array** guardan elementos en cada espacio del índice, partiendo en cero y teniendo un largo que puede variar durante la ejecución del programa. El primer espacio siempre será cero y cada elemento se guardará en cada uno de los espacios.



Existen los arreglos unidimensionales, como el del ejemplo, que constan de elementos guardados índice por índice y, existen los arreglos multidimensionales que constan de filas y columnas y a cuyas posiciones se ingresa indicando [fila][columna].



OBJETOS

En **JavaScript**, como en otros lenguajes de programación, los objetos se pueden comparar con objetos de la vida real. El concepto de Objetos en **JavaScript** se puede entender como objetos tangibles de la vida real.

Pensando en Objetos, imaginemos en una taza: una taza es un objeto con propiedades: tiene un color, un diseño, un peso, un material del cual está hecha. Es así como, al trabajar con **JavaScript** y crear nuestro objeto taza incorporaremos todas esas propiedades o atributos propios del objeto.

Además de los objetos que podemos definir nosotros, existen objetos predefinidos en el navegador, como **String**, del cual podemos utilizas sus métodos. También tenemos **Array** o **Math** que nos entregan importantes métodos (o funciones) que nos facilitan nuestro trabajo. Es así como trabajamos con objetos sin darnos cuenta.

Los temas abordados hoy son complejos y profundizar en ellos requiere de mucho tiempo, por lo que podemos acceder a enlaces que nos ayuden a ir más allá con esta información:

- Errores de JavaScript: lanzar e intentar detectar.
- Try…catch.
- JavaScript orientado a objetos para principiantes.
- Array.
- Bucles e iteraciones.
- Matrices de JavaScript.
- Objetos JavaScript.
- Manejo de errores, “try…catch”.