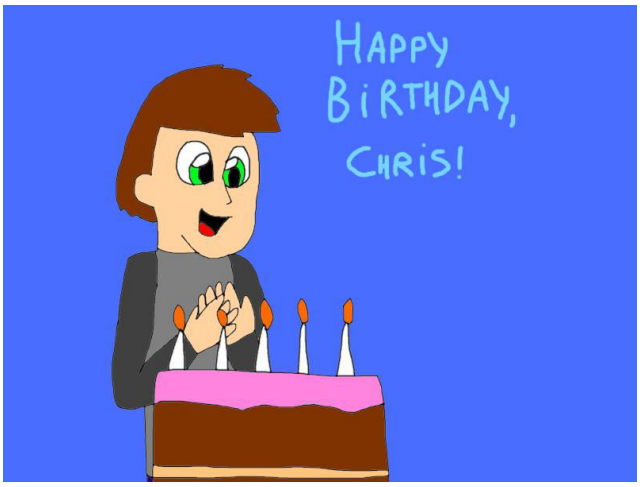
# JavaScript Loops

## Courses Objectives:

1. To achieve repetitive tasks without having to type repetitive code through loops.
2. To understand the different types of loops that exist.
3. What is the difference between a while loop and a for loop and when is appropriate to use the one in favor of the other.
4. What kind of problems loop solve in general.
5. What is the purpose of the 'break' and 'continue' keyword?
6. Learn the use of nested loops.
7. Practice practice practice...

## Happy Birthday to every user!




Imagine that you are having a social network web application with 100.000 users. These users have stored any kind of information to your web app like their name, username, access password, hobbies, interests AND their date of birth.

Imagine that you everyday want to check if a user has it's birthday or not and if yes you want to give him a free premium account for of your app for the next week.

Although you might already know how to accomplish this task for a specific user it is going to be extremely hard to achieve for 100.000 users. That means that you need to write 100.000 conditions and which of them return true in order to give the premium account.

That doesn't sound very efficient right?

# Repetitive tasks with loops

Loops is maybe the most useful and most fundamental concept of structured programming. If it wasn't for loops it would be impossible to achieve all the things that we have done till today, not only in the web industry but in programming in general.
Loops give us the opportunity to write and execute the same block of code repeatedly, many times by changing and modifying a part of it. The modified part executes itself the same way every time, but produces a different result. Let's take a look at our problem.
We have 10 users that they have enrolled themselves to our application. Everyday i want to check if some of them, one of them or maybe all of them have birthday today. This is condition that i need to check. I need to compare if the date today matches the day that they have given us as a birthday. If this is true then i give this user the premium account for the week. If not nothing happens.
Luckily by creating a loop we don't need to write these conditions 10 times. We create a loop that runs 10 times in this case and we check for the same condition at once by changing the date of every user.

How?

You can visualize here a normal loop procedure with this map:

https://www.goconqr.com/c/60238-js-loops/course_modules/90018-loop-flowchart?=

# The while loop

```
// Basic while loop syntax that print all numbers from 1 to 100

var num = 1;

while(num <= 100) {
  // Execute repeatedly until the condition above evaluates to false
  console.log(num);
  /* Don't forget to update the value of num
  in order to avoid loop infinity and browser crash
  */
  num++;
}
```

There are more than one way to execute a repeated code. While loop is one of the commonest. The syntax can be found above in the picture, the first step is instantiating a variable and give a specific value. Most of the times this is going to be a number but as we will see later in the course this is not mandatory.

Inside the while parenthesis follows a condition (that returns a boolean value always). While this condition is true, the code inside the following curly braces will be executed.

Last but not least, we need to update the value of the "Counter" variable. This is very important in order to avoid loop infinity. Otherwise the browser is going to execute forever because you dont have an option to evaluate this initial condition to false and break the loop!

Once the body of the loop is over, the value has been updated and then the conditions is checked again in order to decide if we will execute the body loop once again or we will exit the loop!

Already confused let's see a practical example!

# Loops-4. The container ship problem


Suppose you own a ship.

This is ship can carry up to 10.000 Kg in terms of weight.

A merchant wants you to carry his goods with your ship.

He has 500 containers and he wants you to carry as many as possible. All containers have an identifying number from 1 to 500.
The first 100 containers weight 10Kg each.
The containers with number from 101 up to 200 (included) weight 20kg each.
The containers with number from 201 up to 250 (included) weight 50kg each.
The containers with number from 251 up to 300 (included) weight 100kg each.
The containers with number from 301 up to 400 (included) weight 200kg each.
The rest weight 500 Kg each.

Put as many containers as possible without sinking your ship. Find a way to stop adding weight if you surpass the limit your ship can afford.

Print to the console exact how many containers you have included to your ship and how much is the difference between the total weight of the containers and the weight your ship can afford (free available weight).

# The Break keyword

```
var num = 1;

while(num <= 100) {
  // Execute repeatedly until the condition above evaluates to false
  // Break keyword exits the loop unexpectedly. So the last number here is 49.
  if (num === 50) {
    break;
  }
  console.log(num);
  /* Don't forget to update the value of num
  in order to avoid loop infinity and browser crash
  */
  num++;
}
```

The break keyword can exit the body loop whenever we specify. It is heavily used when we want to stop the loop process under a specific condition is met, usually when an exception occurs.

There is no specific syntax, we just throw the word 'break' inside the condition context and if the code finds itself inside there then and reaches the 'break' line the loop exits!

Try to fix the last problem with container not always by incrementing the id of every container unquestioningly.