Programming

Programming is writing in a language that both computers and humans understand. We will be using the programming language called JavaScript but there are many others.

After writing a program, you can tell a computer to execute the program. This means that the computer reads the program and does what it says. A computer will do one line after another, starting from the top of the program.

Comments

A comment starts with two forward slashes. The computer ignores comments.

```
// This is a comment
```

Booleans

A boolean is a type of value that only has two options, `true` and `false`. We will see how booleans are used when we talk about *conditions* later on.

```
true
false
```

Strings

A string is a list of characters. When programming, strings are always surrounded in quotes. A character can be a letter, number, an empty space, or a symbol. Here are some example strings. Notice that `"45"` is *not* a number. It is a string.

```
"Hello"
"See ya later"
"45"
"I. Like? Symbols:)"
```

Numbers

When programming, you will often want to use numbers. Numbers are exactly what they sound like. Here are some examples:

```
0
4437
9
9823223
```

Variables

Variables are the computers *memory*. You can assign a *value* to a *variable* and then use or update that value later on. For example a computer could remember your name or count from 0 to 10.

In this example, the variable is `myName` and the value is `"Alex"` . Notice that the keyword `let` is put in front of the variable. You need to do this the first time you set a value to a variable. Also notice that *there are no spaces in the name of the variable*. You do not type ~~`my name`~~. Instead, you type `myName` , where there are no spaces and the first letter of each new word is capitalized.

```
let myName = "Alex";
```

In this example, the variable is `score` and the value is `5`

```
let score = 5;
```

In this example, the variable is `didOsuWin` and the value is `true`

```
let didOsueWin = true;
```

Updating Variables

When you want to change the value of a variable, you do the same thing as you did before, except you do not use the `let` keyword.

```
score = 10;
```

Conditions

A condition checks to see if something is true or false. Some examples of every day conditions:

- Did you go to the store?
- Do you like football?
- Are you taller than a Giraffe?

Each of these statements is either `true` or `false` which means that in programming we call them conditions. Here are some examples of things that *are not* conditions.

- How old are you?
- What football team do you like?

A condition must be either `true` or `false`. Here are some examples of conditions in programming.

Equality

In this example, the value `14` is being compared to the value `14` to see if it is equal. Obviously, 14 is equal to 14, so this is true.

```
14 === 14 // true
```

In this example, the value of `10` is being compared to the value of `12` to see if they are equal. Obviously, 10 is *not* equal to 14, so this is false.

In this example, a string is being compared to a number. Even though the string contains the text `"14"`, because this is a string, and not a number, `"14"` is *not* equal to `14`.

```
"14" === 14 // false
```

Greater than

You can also see if a number is bigger than another number. For example, this condition is true.

```
14 > 7 // true
```

```
14 > 14 // false
```

Greater than or equal to

This condition will check if `10` is greater than *or equal to* `9`.

```
10 >= 9 // true
```

And this is also true:

```
10 >= 10 // true
```

Less than

The same conditions exist to check if a number is less than another number.

```
8 < 10 // true

8 < 8 // false

8 < 4 // false
```

Less than or equal to

The same conditions exist to check if a number is less than or equal to another number.

```
8 <= 10 // true

8 <= 8 // true

8 <= 4 // false
```

If Statements

Sometimes in programming, you only want to do something if something is true. For this, we use `if` statements.

```
let age = 10;
let ageToDrive = 16;

if (age >= ageToDrive) {
    console.log("I am old enough to drive");
}
```

In this example, you start with the `if` keyword. You then write an *opening parenthesis* like this: `(`. Between that and the closing parenthesis, which looks like `)`, you write your condition. At the end of the line you write a `{`, which is called a *squigly bracket* (that's the real name). Between the openning squigly bracket and the closing squigly bracket which looks like `}`, you write other lines of code. These lines of code will only be executed if the condition is true.

If Else Statements

You can also do something if a condition is true, and *something else* if a condition is false, using the `else` keyword, like this:

```
let age = 18;
let ageToDrive = 16;

if (age >= ageToDrive) {
    console.log("I am old enough to drive");
} else {
    console.log("I am too younge to drive");
}
```

If Else If

You can combine multiple `if` staments together like this:

```
let age = 14;

if (age >= 18) {
    console.log("You are an adult");
} else if (age >= 13) {
    console.log("You are a teenager");
} else {
    console.log("You are a kid");
}
```

In this example, the *first* condition to be true is what the computer will execute. If none of them are true, then the `else` section is executed.

While Loops

You can also tell the computer to do things over and over again using the `while` keyword. For example:

```
let number = 0;

while (number < 10) {
    console.log(number);
    number = number + 1;
}
```

This will print out the number `0` and then increase it to `1`. After the computer gets to the end of the while loop it goes back to the condition. If it is true, it does everything again. If it is false, it skips to after the while loop closing squigly bracket `}`. The output of this program will be:

```
0
1
2
3
4
5
6
7
8
9
```