Introduction and Variables

This lesson will introduce you to the very basics of JavaScript and how to declare Variables.

WE'LL COVER THE FOLLOWING

Introduction to `JavaScript`

`JavaScript` is a programming language created by Brendan Eich in 1995 that enables interactive web pages and is an essential part of web applications.

If you want to learn more about the history of the language and its name, I suggest that you read this brief article on Medium.com.

How to insert `JavaScript` into `HTML`

If you've ever opened the Chrome developer tools or a similar tool you may have already seen how `JavaScript` is inserted into an `HTML` page.

We do that by using the `script` tag and either inserting our `JavaScript` code directly inside of it or by referencing an external file.

Code inside of the script tag:

```
<script type="text/javascript"> [YOUR_SCRIPT_HERE] </script>
```

Reference an external file:

```
<script src="/home/script.js"></script>
```

Of course, you can add as many scripts as you want and also use both relative, absolute and full path such as:

```
<!-- absolute path from the root of our project -->
<script src="/home/script.js"></script>
<!-- relative path to our current folder -->
<script src="script.js"></script>
<!-- full url to the jquery library -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/core.js"></script>
```

> **Note:** It's better to not write your code inside of the script tag, but instead put it in its own file so that it can be cached by the browser and downloaded only once, regardless of how many files import it. Those files will use the cached version of the file, improving performance.

Variables

We use variables to store values, which can be anything from a username, an address or an item from our e-commerce site, for example.

Prior to ES6 (ES2015) the way we would declare a variable was:

```
var username = "Alberto Montalesi"
```

Now we have 2 more options when it comes to declaring variables:

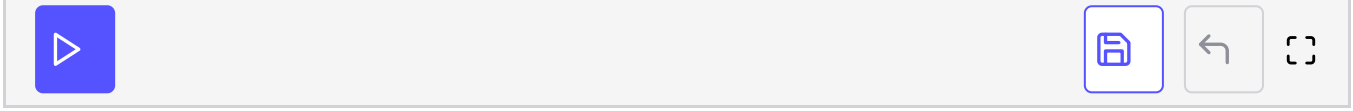
```
let username = "Alberto Montalesi"
const username = "Alberto Montalesi"
```

We will go deeper into the differences between these three keywords later in the course, but let me give you a quick explanation.

Variables created with the keyword `const` are, as the name implies, constant, meaning that they *cannot* be overwritten.

Open your Chrome Developer Tools and try typing the following:

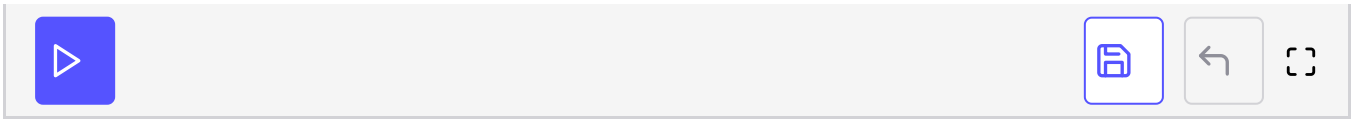
```
const age = 26;
age = 27;
// Uncaught TypeError: Assignment to constant variable
```



As you can see we are not allowed to assign a new value to our constant.

On the other hand if we were to do this:

```
let height  = 190;
height = 189
```



We get no complaint this time, `let` can be reassigned, similarly to the old `var` keyboard.

If both `var` and `let` can be reassigned, then why should we use `let` instead of `var`? The answer for that requires a bit more explanation of how `JavaScript` works, which will be discussed in a later lesson.

Many people argue what the best practice is when it comes to `let` and `const`. Here is my take: use `const` all the time unless you know in advance that you will need to reassign its value.

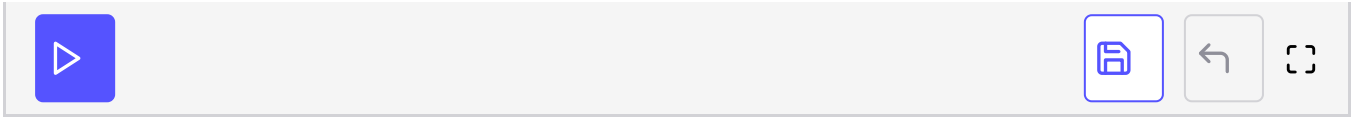
If later on you need to reassign one of your `const`, simply make it a `let` and it will be enough. I find that for myself, it's better to make them `const` by default. Then I see errors if I accidentally try to reassign them rather than having to debug the code later just to find out that I was referencing the wrong variable.

A note about naming variables

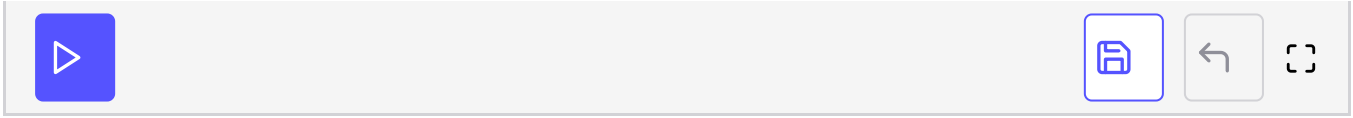
There are certain rules to respect when it comes to naming variables. But don't worry, most of them are very easy to remember.

The following are all forbidden:

```
// variables name cannot start with a number
let 1apple = "one apple";
```



```
// variables name cannot include any character such as spaces, symbols and punctuation marks
let hello! = "hello!";
```



There are also certain words that are reserved and cannot be used as names for variables and functions.

`abstract`	`arguments`	`await`	`boolean`
`break`	`byte`	`case`	`catch`
`char`	`class`	`const`	`continue`
`debugger`	`default`	`delete`	`do`
`double`	`else`	`enum`	`eval`
`export`	`extends`	`false`	`final`
`finally`	`float`	`for`	`function`
`goto`	`if`	`implements`	`import`
`in`	`instanceof`	`int`	`interface`
`let`	`long`	`native`	`new`

`null`	`package`	`private`	`protected`
`public`	`return`	`short`	`static`
`super`	`switch`	`synchronized`	`this`
`throw`	`throws`	`transient`	`true`
`try`	`typeof`	`var`	`void`
`volatile`	`while`	`with`	`yield`

The rule of thumb when it comes to choosing the name for your variable is to make them **descriptive**. Avoid using *acronyms, abbreviations and meaningless names*.

```
// BAD
let cid = 12; // what is a `cid`
// GOOD
let clientID = 12; // oh, a `client id`

// BAD
let id = 12 // what id? userID? dogID? catID?
// GOOD
let userID = 12 // be specific
```

If you want your variable names to be as descriptive as possible, chances are they are going to be multi-words. In that case, the two most common ways of writing variable names are `camelCase` and `snake_case`.

```
// BAD
let lastloggedin = '' // hard to read
// GOOD
let lastLoggedIn = '' // camelCase
let last_logged_in = '' // snake_case
```

Whether you choose to use `camelCase` and capitalize each word of the name after the first one, or you choose to use `snake_case` and put an underscore between each word, remember to be **consistent** and stick to your choice.