The let keyword

The let keyword will allow you to store data for future reference.

Declaration

We can create variables with the `let` keyword. Think of a variable like a drawer. `let` *declares* a variable, which means to you that a drawer is created with a handle.

```
let myDrawer;
```

You can put a value in your drawer:

```
myDrawer="$1.000";
```

In order to access the value, you have to grab the handle of the box and open it. In this example, you have a drawer called `myDrawer`. It contains a string written `'$1.000'` on it.

To access your thousand bucks, you have to open the drawer:

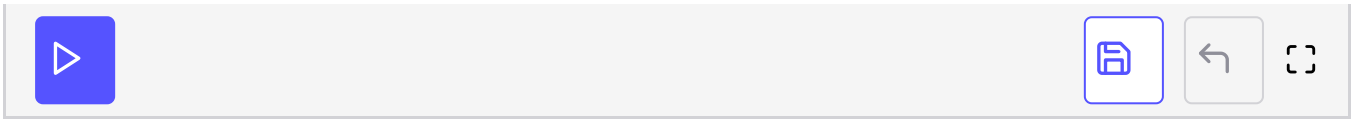
`myDrawer`
`'$1.000'`

Initialization

You can assign an initial value to your variable with the `=` sign. This is called *initialization*, and it can occur either in the same statement where you declared the variable (see `x`), or after the declaration (see `y`). You may access a declared variable even if you have not initialized it. Its value becomes `undefined`.

```
let x = 5;
let y;

y = x ** 2;
let z;
console.log( x, y, z );
```



Move `let z` below the `console.log` statement. You should see a ReferenceError:

```
ReferenceError: z is not defined
```

The message is somewhat misleading, because it means `z is not declared` using the `let` keyword. Don't mix this message with the `undefined` value. You only get the above reference error if you reference a variable that does not exist.

You did the following: you asked for the contents of your drawer `z` in the console log. But the drawer itself does not exist yet. You created the drawer afterwards, with `let z;`.

Side note: I know, in most tutorials, you see var instead of let. This is an advantage of reading an ES2018-compliant tutorial. Don't worry about var for now, you will hear about it later. Ok, I understand. If you do worry about it, read this article.