Variables

Variables are containers for pieces of data. That data can be one of many different data types. It's important to know and understand those data types and we will go over them in the next lesson, but right now, we are just going to look at the syntax for creating and re-assigning variables as well as the differences between how we declare them.

Declaring a Variable

In JavScript, we need to first declare a variable with one of three keywords

- var
- let
- const

In modern JavaScript, you probably won't see `var` very much. `var` was the original declaration, but in `ES2015` also known as `ES6`, which was a huge update to the language, they introduced `let` and `const`. One of the reasons for this is because of `block scoping`. Now, I'm not going to talk about things like scope right now, because we haven't gotten to functions or anything, so if I talk about scope it'll go right over a lot of people's heads. I just want to focus on how we declare and assign variables and also work with constants. Just know that on the `global scope`, meaning not inside of a function or any kind of control structure, `var` and `let` work in a very similar same way. `const` is a bit different. I'll be using let and const throughout the course, unless there is a specific reason for me to use var.

So let's say that we want a variable called firstName and lastName. Remember that `string` data types need to be wraped in either single or double quotes. You can also use backticks. Backticks (`) have a special use which I'll talk about later.

```
let firstName = 'John';
let lastName = 'Doe';

// We can do a console.log to show the value
console.log(firstName, lastName);
```

We can store other data types such as numbers:

```
let age = 30;

console.log(age);
```

Variable Naming Conventions

So different languages have different rules and conventions when it comes to naming things. There are some rules that you have to follow when it comes to the formatting of your variable names.

- Only letters, numbers, underscores and dollar signs
- Can not start with a number

I wouldn't suggest starting your variable names with a dollar sign or an underscore either.

Multi-Word Variables

When it comes to variables as well as functions and classes with multiple words, it's really up to you on how you format the case. What you'll typically see in JavaScript and what I usually do is `camelCase`. This where we start with a lowercase letter but every word after that starts with an uppercase letter.

```
let firstName = 'John';
```

You may also see underscores like this

```
let first_name = 'Sara';
```

There's also pascal case, where the first word is also capitalized. You typically see this for class names in object oriented programming.

```
let FirstName = 'Tom';
```

You might also see all lowercase, which I wouldn't recommend

```
let firtstname = 'Bob';
```

Reassigning Values

Alright, so if we want to reassign a value we can do that here since we're using var. When it comes to directly reassigning a primitive type like a number, you can't use `const`. Const stands for **constant**. So you have to use `var` or `let` if you want to re-assign a variable value. Again, you're not going to see `var` very much, so I will use `let` for this.

```
let x = 100;
```

Let's reassign x to 200

```
x = 200;
```

Now, in some cases, you may want to simply declare a variable and not assign a value to it.

```
let score;
```

In this case, score would be equal to **undefined**, which is actually one of the seven primitive data types that we're gonna talk about in the next video.

If I want to assign a value to it at any time, I can.

```
score = 1
```

One reason you might do this is because you have a conditional that says if score equals one thing, do this, if another, then do something else. Here is an example

```
let score;

if(userScores) {
  score += 1;
} else {
  score -= 0;
}
```

Constants

Alright, let's look at `const`, which works a bit differently than `let` or `var`. So if I declare a name like this...

```
const x = 100
```

and then I try and re-assign that value

```
x = 200; // Results in error
```

I get an error. You can't directly re-assign a value to a constant. You also can't initialize a constant as undefined.

```
const score1; // Results in error
```

That will also throw an error. It has to be declared with a value.

Now where this can be a little confusing is when we use `const` with values that are not primitive like objects or reference types such as `arrays` and `object literals`. In that case we can't directly re-assign, but we can change them.

So let's say that we have an array

```
const arr = [1,2,3,4]
```

What I can't do is re-assign

```
arr = [1,2,3,4,5] // Results in error
```

But I can for instance, add to that array with the push method. I can even take a specific index of the array and change the value that way.

```
arr.push(5); // [1, 2, 3, 4, 5]
arr[0] = 200; // [200, 2, 3, 4, 5]
```

We can do the same type of thing with object literals. Now I know some of you have no clue what an object literal is, we'll get to that, but I just want to show you that the object itself can be manipulated, even with const.

```
const person = {
  name: 'Brad',
};

person.name = 'John';
person.email = 'john@gmail.com';

console.log(person);

/*
Results in...
{
  name: 'John',
  email: 'john@gmail.com'
}
*/
```

Declaring multiple values at once

We don't have to declare variables line by line, we are able to declare multiple values at once. With let, we can initialize, with const, we have to assign a value.

```
let a, b, c;
const d = 10, e = 20, f = 30;
```

Let or Const - Which to Use?

So how do we figure out which to use when it comes to let and const or even var? We'll it comes down to preference. What I like to do is always use `const` unless it's a primitive value that I think I may need to re-assign at some point. The score example above is a good example. That score number will be re-assigned throughout the game. So I would use let. You'll find in most cases that you don't need to explicitly re-assign values. We're usually dealing objects where we manipulate them but don't re-assign them.

Some people do the opposite and always use `let` no matter what. Which is fine, but I think using `const` is a bit more robust because you know your values can't be re-assigned my mistake. So, I would suggest `const` unless you know you need to either initialize as undefined or re-assign. It's all preference though.