JavaScript variables

Example

```
var x = 5;
var y = 6;
var z = x + y;
```

Much Like Algebra

x = 5
y = 6
z = x + y

In algebra we use letters (like x) to hold values (like 5).

From the expression z = x + y above, we can calculate the value of z to be 11.

In JavaScript these letters are called **variables.**

 JavaScript variables are containers for storing data.

JavaScript Variables

As with algebra, JavaScript variables can be used to hold values (x = 5) or expressions (z = x + y).

Variable can have short names (like x and y) or more descriptive names (age, sum, totalVolume).

- Variable names must begin with a letter
- Variable names can also begin with $ and _ (but we will not use it)
- Variable names are case sensitive (y and Y are different variables)

 Both JavaScript statements and JavaScript variables are case-sensitive.

The Assignment Operator

In JavaScript, the equal sign (=) is an "assignment" operator, is not an "equal to" operator.

This is different from algebra. The following does not make any sense in algebra:

```
x = x + 5
```

In JavaScript, however it makes perfect sense: Assign the value of x + 5 to the variable x.

In reality: Calculate the value of x + 5. Then put the result into the variable x.

 The "equal to" operator in JavaScript, is written like == or ===. You will see it soon!.

JavaScript Data Types

JavaScript variables can hold many types of data, like text values (person = "John Doe").

In JavaScript texts are called strings or text strings.

There are many types of JavaScript variables, but for now, just think of numbers and strings.

When you assign a string value to a variable, you put double or single quotes around the value.

When you assign a numeric value to a variable, you do not put quotes around the value.

If you put quotes around a numeric value, it will be treated as a text string.

Example

```
var pi = 3.14;
var person = "John Doe";
var answer = 'Yes I am!';
```

Declaring (Creating) JavaScript Variables

Creating a variable in JavaScript is called "declaring" a variable.

You declare JavaScript variables with the **var** keyword:

```
var carName;
```

After the declaration, the variable is empty (it has no value).

To assign a value to the variable, use the equal sign:

```
carName = "Volvo";
```

You can also assign a value to the variable when you declare it:

```
var carName = "Volvo";
```

In the example below we create a variable called carName, assigns the value "Volvo" to it, and put the value inside the HTML paragraph with id="demo":

Example

```
<p id="demo"></p>
var carName = "Volvo";
document.getElementById("demo").innerHTML = carName;
```

One Statement, Many Variables

You can declare many variables in one statement.

Start the statement with **var** and separate the variables by **comma**:

```
var lastName = "Doe", age = 30, job = "carpenter";
```

Your declaration can also span multiple lines:

```
var lastName = "Doe",
age = 30,
job = "carpenter";
```

In JavaScript you can always separate statements by semicolon, but then you cannot omit the var keyword.

Wrong:

```
var lastName = "Doe"; age = 30; job = "carpenter";
```

Right;

```
var lastName = "Doe"; var age = 30; var job = "carpenter";
```

Value = undefined

In computer programs, variables are often declared without a value. The value can be something that has to be calculated, or something that will be provided later, like user input. Variable declared without a value will have the value**undefined**.

The variable carName will have the value undefined after the execution of the following statement:

```
var carName;
```

Re-Declaring JavaScript Variables

If you re-declare a JavaScript variable, it will not lose its value:.

The value of the variable carName will still have the value "Volvo" after the execution of the following two statements:

```
var carName = "Volvo";
var carName;
```
