UNIT 01

LESSON 01.01

JavaScript BOOTCAMP

---

var vs. let

string variables

number variables

typeof() method

boolean variables

undefined variables

---

variables

A variable is a container that stores a value. Some variables (vars, for short) can hold many values at a time. Other vars can only hold one value at a time. Vars that only hold one value at a time are sometimes referred to as **primitive tyes**. These "primitives" are the topic of this lesson.

variable data types

There are two major categories of variables, **objects** and **primitives**:

- **object** is a broad **data type** encompassing variables that can store multiple values. These include:
    - **object** -- a data structure with properties as name-value pairs and, optionally, with methods (functions scoped to the object)
    - **array** -- ordered lists of items, with each item stored by its numeric position, called the index.
    - **DOM object** -- JS "versions" of html elements (div, button, etc.)
    - **function** -- code blocks which only run when invoked (called)
    - **null** -- an empty object, used typically as a placeholder value
    -
- **primitives** are variables that are capable of storing only *one* value at a time. Primitives come in various **data types**:
    - **string** -- with value as text in quotes (e.g. "Hello World")
    - **number** -- with value as integer and decimal (e.g. 3, 3.5)
    - **boolean** -- with value of only **true** or **false**
    - **undefined** -- with no value assigned, typically used with the understanding that a value will be assigned later
    -

In this lesson we will focus on primitives.

declaring variables with let

To begin using a variable, we need to *declare* it. This is done by writing a *keyword* (**var** or **let**), followed by the name of your variable, which can be pretty much anything you like, although naming rules and conventions do apply:

variable naming rules and restrictions

- No spaces allowed in variable names
- No special characters allowed, except **$** and **_**
- Name cannot start with a number (**1day** bad; **day1** good)
- No reserved words allowed (**alert** bad; **myAlert** good)

variable naming conventions (best practices)

- Use **camelCase** (it's **highScore**, not **high_score**)
- Don't use all UPPERCASE, unless declaring a constant (value will never change)
- Choose concise names (**tel**, not **telephoneNumber**)
- Choose precise names (**salesTax** not **additionalCharge**)

A variable is declared with **var** or **let**, altnough **let** is the more modern syntax and should be used instead of **var** for reasons that we will expound when we start talking about **variable scope**.

string variables

Strings are text enclosed in either single or double quotes:

1. Declare a variable with **var** and assign it a string in double quotes:

```
var pet = "cat";
console.log(pet); // cat
```

2. Change the value to another string, this time in single quotes:

```
pet = 'dog';
console.log(pet); // dog
```

One difference between **var** and **let** is that a **var** can be redeclared.

3. Redeclare **pet**:

```
var pet = 'bunny';
console.log(pet); // bunny
```

A variable declared with **let** cannot be redeclared--attempting to do so throws an error:

4. Declare a variable with **let**, and then try to redeclare it:

```
let petSound = "Woof!";
console.log(petSound); // Woof!
let petSound = "Grrr!";
Error: Identifier 'petSound' has already been declared
```

To change the value of an existing variable, don't redeclare it. Just set it equal to something else.

5. Comment out **let petSound = "Grrr!"**, and then set **petSound** to "Grrr!" without redeclaring the variable:

```
// let petSound = "Grrr!";
petSound = "Grrr!";
console.log(petSound); // Grrr!
```

Multiple values can be outputted in the same console.log:

6. Output both variables in one console.log:

```
console.log(pet, petSound); // dog Grrr!
```

7. For added clarity, you can include 'labels' in the console output:

```
console.log('pet:', pet, ' petSound:', petSound);
// pet: dog  petSound: Grrr!
```

number variables

A number can be an integer (int, for short) or a float (decimal). There are no commas in numbers. Whether a number is an integer or a float, its data type is number.

8. Declare three number variables, setting them equal to integer, float, and four-digit int:

```
let price1 = 35; // integer
let price2 = 3.5; // float
let price3 = 3500; // no comma
console.log(price1, price2, price3); // 35 3.5 3500
```

Naturally, number variables can be used to do math. The result of a mathematical calculations is often saved to a variable, but directly console logging the answer also works:

9. Take these basic math operators for a spin: **+, -, *, /**:

```
    let sum = price1 + price2 + price3;
    console.log(sum); // 3538.5
    console.log(price1 - price2); // 31.5
    console.log(price1 * price2); // 122.5
    console.log(price1 / price2); // 10
```

boolean variables

A boolean is a variable that can only be **true** or **false**. The names of booleans often begin with *is* to emphasize the either-or concept and to let you know at a glance that this is a boolean.

10. Declare a boolean variable with a value of *true*:

```
    let online = true;
    let isOnline = true; // prefix 'is' helps us know that this is a
boolean
    console.log(online, isOnline); // true true
```

Changing the value of a boolean is known as *toggling* or *flipping* the boolean. This can be done by direct assignment, or by setting the variable equal to itself with **!** in front of it.

11. Flip the boolean from true to false, and then flip it back by putting **!** in front of it:

```
    isOnline = false;
    console.log('isOnline', isOnline); // isOnline false
    isOnline = !isOnline;
    console.log('isOnline', isOnline); // isOnline true
```

undefined

A variable can be declared without a value being assigned. The assumption is that a value will be assigned later. Until then, both the value and datatype are **undefined**.

12. Declare a variable, but don't assign it a value:

```
    let player1;
    console.log('player1', player1);
    // player1 undefined
```

typeof() method

The **typeof()** method takes a variable as its argument and returns the datatype.

13. Declare variables of each of the four major "primitive" types: string, number, boolean and undefined. Then log the variable name, value and data type:

```
    let ketchup = "Heinz";
    console.log('ketchup', ketchup, typeof(ketchup));
    // ketchup Heinz string

    let varieties = 57;
    console.log('varieties', varieties, typeof(varieties));
    // varieties 57 number

    let fresh = true;
    console.log(fresh); // true
    console.log('fresh', fresh, typeof(fresh));

    let total;
    console.log('total', total, typeof(total));
    // total undefined undefined
```

Multiple variables can be delared in one line with a single keyword. The individual declarations are separated by commas.

14. Declare three variables with just one instance of the **let** keyword:

```
    let x = 1, y = 2, z = 3;
    console.log(x + y * z); // 7
```

Such "one-liner" variable declarations are more common when the vars that are not to be assiged an initial value.

15. Declare four variables with just one instance of the **let** keyword, leaving them all without an initial value, so *undefined*:

```
    let day, date, month, year;
    console.log(day, date, month, year);
    // undefined undefined undefined undefined
```

Undefined variables are not errors, but some programmers prefer to avoid them, anyway. As a workaround, you can assign a starter value and then just change it later. The conventional starters are **""** (empty string) for strings and **0** for numbers:

16. Declare a string and a number, assigning them starter values of "" and 0, respectively:

```
    let grade = "";
    let score = 0;
```

declaring a variable with value equal to another variable
If you declare a variable and set its value equal to an existing variable, the copy becomes its own independent entity, which means that the copy is in no way "linked" to the original. If you change the value of the copy, the original is not aaffected.

17. Declare a variable, and set it equal to score from the previous step:

```
    let myScore = score;
    myScore = 98;
    console.log(myScore); // 98
    console.log(score); // 0
```

- **END Lesson 01.01**
- **NEXT: Lab 01.01**
- **Lesson 01.02**