UNIT 01

LESSON 01.01

---

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

- **string** is a **datatype** for text.
- **string** values go in quotes (double or single quotes both work)

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 'label' 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 numeric values.
- Be it integer or float, a number's **datatype** is number.

8. Declare three numeric 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, numeric variables can be used in mathematical operations, the result of which may also be assigned to a variable.

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(price2 * price3); // 12250
console.log(price3 / price1); // 100
```

boolean variables

A boolean is a variable with a value that is either **true** or **false**.

- boolean names often begin with *is* to emphasize the either-or concept
- *toggling* or *flipping* a boolean refers to changing its value

10. Declare two booleans, each with a value of **true**:

```
let premiumMember = true;
let isOnline = false; // 'is' indicates that this is a boolean
console.log(premiumMember, isOnline); // true false
```

Toggling or *flipping* a boolean can be done either by direct assignment or by putting an exclamation point (**!**) in front of it.

11. Flip **premiumMember** from true to false by direct assignment. Also flip **isOnline**, but do so by putting **!** in front of it:

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

The advantage of using **!** to toggle/flip a boolean is that you do not need to know the current value; whatever it is, **!** makes it the opposite.

undefined

A variable can be declared *without* a value being assigned. The assumption is that a value will be provided later. Until then, both 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** datatypes: string, number, boolean and undefined. Then log the name, value and datatype:

```
    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 isFresh = true;
    console.log('isFresh', isFresh, typeof(isFresh)); // isFresh true
boolean

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

Multiple variables, separated by commas, can be delared in *one* line of code.

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 **undefined** (no value) variables with just one **let**:

```
    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 with the intention of changing it later. This has the advantage of indicating the datatype:

16. Declare a string and a number, with starter values of empty string **""** and **0**:

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

declaring a variable equal to another variable
If you set a variable equal to another variable, the "copy" is its own independent entity. If you change the value of the "copy", the "original" remains unchanged.

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

```
let greeting = "Hola";
let greeting2 = greeting;
greeting2 = "Howdy";
console.log(greeting2); // "Howdy"
console.log(greeting); // "Hola"
```

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