Lesson 01.01

- var vs. let for declaring variables
- primitive variable types: string, number, boolean, undefined
- typeof() method for checking data type

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 types**. These “primitives” are the topic of this lesson.

variable data types

There are two broad 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 (datatype is function)
- **null**: an empty object, typically missing or placeholder values

primitives are variables that are capable of storing just **one** value at a time. Primitives are technically objects too, although object generally refers to the above. Primitives have the following **data types**:

- **string**: value as text in quotes (e.g. "Hello World")
- **number**: value as integer and decimal (e.g. 3, 3.5)
- **boolean**: value is either **true** or **false**
- **undefined**: no value assigned (yet), understanding may be that a value will be assigned later

In this lesson we will focus on primitives.

declaring variables with let

To begin using a variable, you 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 that will never change)
- Choose concise names (**tel**, not **telephoneNumber**)
- Choose precise names (**groceries** not **list**)

A variable is declared with **var** or **let**, although **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
```

3. One difference between **var** and **let** is that a **var** can be redeclared. Redeclare **pet**:

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

redeclaring variables

A variable declared with **var** may be redeclared, but 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
```

5. To change the value of an existing variable, don't redeclare it; just set it equal to something else. Comment out **let petSound = "Grrr!"**, and then set **petSound** to "Grrr!" *without redeclaring* the variable:

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

6. Multiple values can be outputted in the same **console.log**: Output both variables in *one* console.log:

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

7. For added clarity, you can 'label' console output:

```
console.log('pet:', pet, ' petSound:', petSound);
// pet: bunny  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: integer, float, and four-digit int:

```
let price1 = 35; // integer
let price2 = 7; // float
let price3 = 3500; // no comma
console.log(price1, price2, price3); // 35 7 3500
```

9. Naturally, numeric variables can be used to do math:

```
/*
    +   addition
    -   subtraction
    *   multiplication
    /   division
    **  exponentiation
    %   remainder (modulo)
*/

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 2 booleans, one **true**, one **false** :

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

! opearator

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

11. Flip isOnline from true to false by direct assignment:

```
isOnline = false;
console.log('isOnline:', isOnline); // false
// Flip it again, but this time by putting ! in front of itself:
isOnline = !isOnline;
console.log('isOnline:', isOnline); // false
console.log('isOnline:', isOnline); // true
isOnline = !isOnline;
console.log('isOnline:', isOnline); // false
```

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 4 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
```

14. Multiple variables, separated by commas, can be declared in **one** line of code. Declare 3 variables with just one instance of the **let** keyword:

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

If all vars have same initial value, just put the value once at the end:

```
```js
    let score1, score2, score3 = 0;
    // same as:
    // let score1 = 0;
    // let score2 = 0;
    // let score3 = 0;
```
```

15. Such "one-liner" variable declarations are common for vars that are not assigned an initial value. Declare 4 **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;
```

changing a variable's datatype
JavaScript is "loosely typed", meaning you can change the datatype of a variable.

17. Declare a string and change it to a number:

```
let price = "$100";
price = 100;
```

Just be sure to have a good reason for changing a datatype.

DONE: Lesson 01.01

NEXT: Lab 01.01

01.01 Lab Solution

- Debug these variable names, supplying values of indicated datatype.
- Log the result, replacing 'VAR' with your variable.

Example:

0. Dubug this string variable:

```
let first-name = Bob;
// solution:
let firstName = 'Bob';
console.log('0.', 'firstName', firstName, typeof(firstName));
// 0. firstName Bob string
```

1. Dubug this number variable:

```
// let admission Fee = $10;
let admissionFee = 10; // '$10' is a string
console.log('1:', 'admissionFee', admissionFee, typeof(admissionFee));
// 1. admissionFee 10 number
```

2. Dubug this string variable:

```
// let #1sportsCar = "Porsche";
let sportsCar1 = "sportsCar1"; // # no special characters
console.log('2:', 'sportsCar1', sportsCar1, typeof(sportsCar1));
// 2. sportsCar1 sportsCar1 string
```

3. Dubug this boolean variable:

```
// let is Online = FALSE;
let isOnline = false; // JS boolean values are lowercase
console.log('3:', 'isOnline', isOnline, typeof(isOnline));
// 3. isOnline false boolean
```

4. Dubug this number variable:

```
// let %done = 22.5%;
let pctDone = .225; // divide by 100 to convert % to decimal
console.log('4:', 'pctDone', pctDone, typeof(pctDone));
// 4. pctDone .225 number
```

5. Dubug this string variable:

```
// let 26miles = marathon;
let miles26 = "marathon"; // cannot begin variable name with a number
// or:
let _miles26 = "marathon"; // _ is legal character in variable name
console.log('5:', 'miles26', miles26, typeof(miles26));
// 5. miles26 marathon string
```

6. Dubug this string variable:

```
// let $100000Bar = candy bar;
let $100000Bar = "candy bar"; // $ is allowed in JS var name
console.log('6:', '$100000Bar', $100000Bar, typeof($100000Bar));
// 6. $100000Bar candy bar string
```

7. Dubug this number variable:

```
// let first-prize = 7,500;
let firstPrize = 7500; // numeric values do not have commas
console.log('7:', 'firstPrize', firstPrize, typeof(firstPrize));
// 7. firstPrize 7500 number
```

8. Dubug this boolean variable:

```
// let i_Won! = True;
let iWon = true; // i_Won is legal name, but camelCase is better
console.log('8:', 'iWon', iWon, typeof(iWon));
// 8. iWon true boolean
```

9. Without changing the the value of x, set the value of y

```
//    in the following console.logs to get an answer of 10
//    or -10, as indicated.
let x = 20;
let y;

// y = -10;
console.log(x + y); // 10

y = 30;
console.log(x - y); // -10

y = 0.5;
console.log(x * y); // 10

y = -2;
console.log(x / y); // -10

y = 3;
console.log(x ** y); // 8000
```

10. Calculate the total cost, as shown in the comment:

```
let unitCost = 50;
let numUnits = 12;
let shipping = 25;
let totalCost = unitCost * numUnits + shipping;
console.log('totalCost', totalCost); // 625
```

NEXT: Lesson 01.02