Springboard

Variables and Data Types

Goals

- Identify and use the core data types of JavaScript
- Use all the different ways to declare and assign variables
- Explain why JavaScript is called a dynamically-typed and weakly-typed language
- Use your understanding of Numbers and Strings to get started coding!

What Are Data Types?

- In any programming language, you have to distinguish between different kinds of data
- For example, in English we might say that 1 is a number, whereas “one” is a word.
- Programming languages define separate data structures which are built-in representations for types.

Primitives

- String (‘hello’)
- Number (3.141592)

Goals
What Are Data Types?
Primitives
Numbers
Number Methods
Numeric Operators
Operator Preceden...
NaN
Variables
Variable Naming Conventi...
Variables Can Change Type
Variables Can not be redeclared
With const, variables can not be reassigned
Compari... of Variable Declarati... Keywords
What about var?

- Boolean (false)
- Undefined (undefined)
- Null (null)

Numbers

The number primitive in JavaScript is different than in other languages, because it supports both integers and floating-point (decimal) numbers by default.

```
let anInteger = 9000; let floatingPointNumber = 2.71828;
```

The number constructor function is written as Number(). You can convert strings to numbers using it, or + for shorthand.

```
let nine = '9'; typeof nine; // string // with number
constructor typeof Number(nine); // number // shorthand with +
typeof +nine; // number
```

Number Methods

There are a handful of useful methods to remember on the Number function, in addition to Number.isNaN().

- ***Number.isInteger()*** returns true or false
- ***Number.parseInt()*** takes an integer out of a string or float, non-numeric trailing characters ignored
- ***Number.parseFloat()*** takes a float value out of a string, non-numeric trailing characters ignored.

```
Number.isInteger('15'); // false Number.isInteger(1.234); //
false Number.isInteger(15); // true Number.parseInt('15'); //
15 Number.parseInt(3.14); // 3 Number.parseFloat('4.65xyz'); //
4.65
```

Numeric Operators

In JavaScript, there are many additional operators, and we've already seen some (= assignment, == equality, === strict equality).

- + Addition 10 + 5 === 15
- Subtraction 17 - -5 === 22
- Multiplication 10 * 10 === 100
- / Division (floating-point) 5 / 4 === 1.25
- % Remainder after dividing left number by right number. (Sometimes called modulo) 16 % 5 === 1 (returns 1 because 5 goes into 15 cleanly, then 1 remaining)
- ** Exponent (power) 5 ** 2 === 25

Operator Precedence

Do you remember PEMDAS from primary / secondary school?

That order of precedence applies to JavaScript!

- **P** Parentheses (Please)
- **E** Exponents (Excuse)
- **M** Multiplication (My)
- **D** Division (Dear)
- **A** Addition (Aunt)
- **S** Subtraction (Sally)

NaN

A big frustration with JavaScript can be NaN (Not a Number).

NaN is a special value; the result of a failed conversion (or type coercion) to number.

```
let oldNan = Number('winter is coming'); // NaN let
sillyDivision = 5 / 'tofu'; // NaN
```

Variables

- Variables are just containers that hold data types for us.
- In JavaScript, variables are declared with special keywords: var, let, or const.
- We're going to be using ***let*** and ***const*** exclusively, it's very rare that you will need ***var*** anymore.
- We'll be covering the differences in far more detail later on!

```
let myName = 'Colt Steele'; let myAge = 87;
```

```
let cool; // we say 'cool' is declared but not assigned a value
Then they are assigned a value with the single equals sign:
cool = false; // cool is assigned (previously was undefined)
```

Variable Naming Conventions

Most variables in JavaScript are lowerCamelCase.

```
let thisIsLowerCamelCase = true;
```

Constants might be represented as UPPER_SNAKE_CASE

```
const AVOGADROS_CONSTANT = 6.022140857 * 10 ** 23;
```

Variables Can Change Type

If you declare a variable, the default value is undefined.

```
let cat; typeof cat; // undefined
```

```
cat = 'Scout'; typeof cat; // string
```

```
cat = true; typeof cat; // boolean
```

```
cat = 1.57 typeof cat; // number
```

Then you can re-assign the variable as many times as you want*.

We'll talk about the one exception to this soon!

Variables Can not be redeclared

If you use ***let*** or ***const***, you can not redeclare variables

```
let firstName = "Tamara"; let firstName = "Tamara"; // Error!
const numberOfStates = 50 const numberOfStates = 51 // Error!
```

With const, variables can not be reassigned

The biggest difference with let and const is assignment and declaration.

```
let firstName = "Tamara"; firstName = "Isadore"; // works fine!
const numberOfStates = 50 numberOfStates = 51 // Error!
```

Comparison of Variable Declaration Keywords

Keyword	Can Reassign	Can Redeclare
var	yes	yes
let	yes	no
const	no	no

What about *var*?

There’s really no need to use it