

# INTRODUCTION TO JAVASCRIPT, DATA TYPES, VARIABLES

# DATA TYPES

A **data type** is a way of classifying or identifying different types of data

With data types, we can determine:

- possible values for that type
- operations that can be performed on that type
- the meaning of the data
- the way values of that type can be stored

Most programming languages have the same **data types** (there's always a caveat)

- **strings**: word or sentence surrounded by "quotes"  
-- 'kittens are soft'
- **integers**: any number without a decimal point -- 7
- **floats**: any number with a decimal point -- 1.03
- **booleans**: true or false
- **arrays**: collections of data -- []
- **associative array**: collection of data with key-value pairs -- ['a': 200, 'b': 300]
- **objects**: a representation of something

## Array:

```
[ 'dogs', 'cats', 'turtles', 'birds' ]
```

## Object:

```
let fakeObject = {  
  color: "blue",  
  fabric: "cotton",  
  size: "M"  
};
```

In JavaScript, there are five **primitive data types**

- **string**
- **number**
- **boolean**
- **undefined**
- **null**

A **string** is a sequence of characters, surround by either  
" " or ' '

A **number** is literally just a number, no " " or ' '  
necessary

A **boolean** has one of two possible values, for us those  
are true or false

**Undefined** just mean something has not yet been  
declared or defined

**Null** is an intentional absence of a value



Everything else is an **object**

- **arrays, [ ]**
- **functions,**
- **objects, { },**

An **array** holds multiple and many values under one name, which you can access via an **index**

**Functions** are blocks of code that we can use or reuse to perform tasks for us

JavaScript **objects** are containers for named values called **properties** or **methods**

## Array:

```
let seinfeld = ['Jerry', 'George', 'Elaine', 'Kramer'];  
let numbers = [44, 3455, 3432434, 9877];
```

## Function:

```
function nameOfFunctionWeInvented() {  
  // do cool things  
}
```

## Object:

```
{  
  name : 'Justin Elm'  
}
```

There's also a core set of language elements:

- operators: +, -, ===
- control structures
- statements

```
if ( somethingIsTrue ) {  
  // do something else  
}
```

```
let x = 1;
```

So really, in the most fundamental sense, we're talking about various tools available to us at the start that we can use to create our projects.

Today, and over the next few weeks, we're going to use those tools to create basic programs, software, and websites.

# VARIABLES

In trying to understand the concepts of **variables**, it might be helpful to look at how the dictionary defines the word.

variable (adjective):

1. apt or liable to vary or change; changeable
2. capable of being varied or changed; alterable

variable; mathematics, computer (noun):

1. a quantity or function that may assume any given value or set of values

Trying to define further, and borrowing from MDN, we will use variables as symbolic names for storing values in our code. We often call the names of variables "identifiers", and we often want them to conform to convention.

One note, variable names/identifiers are case sensitive.



In JavaScript there are three ways to declare variables.  
The three different ways behave slightly differently,  
which we will elaborate later.

**var:** Declares a variable

**let:** Declares a block-scoped, local variable

**const:** Declares a block-scoped, read-only named  
constant

## Examples:

```
var x = 15;  
  
let x = 'a string';  
  
let z = '@VeryG00dpa$$word!';
```

Note, the values/data types being assigned are not particular to the different type of variable declaration.

As mentioned before, **variables** are case sensitive.

```
let namedVariable = 'justin';  
let NAMEDVARIABLE = 'justin';
```

These are considered two separate things, even though the assigned value is the same.

You can also declare a variable without assigning a value to be used later.

```
var a;  
  
let useThisLater;
```

These **variables** will have an assigned value of 'undefined' automatically.

In the next example, pay attention to when the variable is declared relative to the `console.log` function.

```
var a;  
console.log('The value of a is ' + a);  
// The value of a is undefined  
  
console.log('The value of b is ' + b);  
var b;  
// The value of b is undefined  
  
console.log('The value of c is ' + c);  
// Uncaught ReferenceError: c is not defined
```

```
let x;  
console.log('The value of x is ' + x);  
// The value of x is undefined  
  
console.log('The value of y is ' + y);  
let y;  
// Uncaught ReferenceError: y is not defined
```



We'll go over these ideas more in class.

As usual, email me with questions.