HACK REACTOR

JavaScript 101

Syntax, Types, Variables, Operators & Functions

Part I: What is JavaScript?

JavaScript is not Java

- JavaScript and Java are nothing alike
- JavaScript got its name from Java for marketing hype

What is JavaScript?

- A programming language for computers
- Commonly runs inside of web browsers
  - Think: Google Chrome, Internet Explorer, Firefox, Safari
  - Helps to create web page interactivity
- Less commonly, JavaScript can be used for almost anything as a general-purpose programming language
  - Programming robots or home automation tools
  - Programming a drone
  - Programming iOS and Android apps

What is JavaScript primarily used for?

- Animations (i.e slideshows)
- Event handling (i.e clicking a button)
- Dynamic styling & visualizations
- Information entered in forms
- Much, much more

Who uses JavaScript?

Everybody.

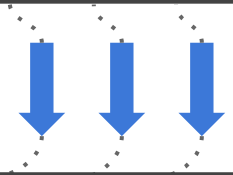
While learning JavaScript, remember:

- Web browsers will interpret JavaScript code just like you:
  - Left-to-right, top-to-bottom
  - Think of what it's like to read a book
- All JavaScript code is limited by very specific constraints
  - These constraints are defined by JavaScript's "syntax"
  - The word "syntax" relates to the structure and formatting of your code, down to the character
  - If you see a "syntax error", it could mean something as simple as a missing or extra character
- JavaScript works as an input/output system
  - When you put something in, you should intend to get something back

What code looks like

JavaScript Code Input:

```
Some code goes here
```



Output:

```
Some output appears here
```

Part II: Types in JavaScript

What is a type?

A predefined category used to differentiate information in a computer program.

What types exist in JavaScript?

- Strings
- Numbers
- Objects (and Arrays)
- Booleans (true or false)
- Null
- Undefined

Note: Every programming language has a different nomenclature for its types; for instance, in Python, JavaScript Arrays are recognized as "Lists"

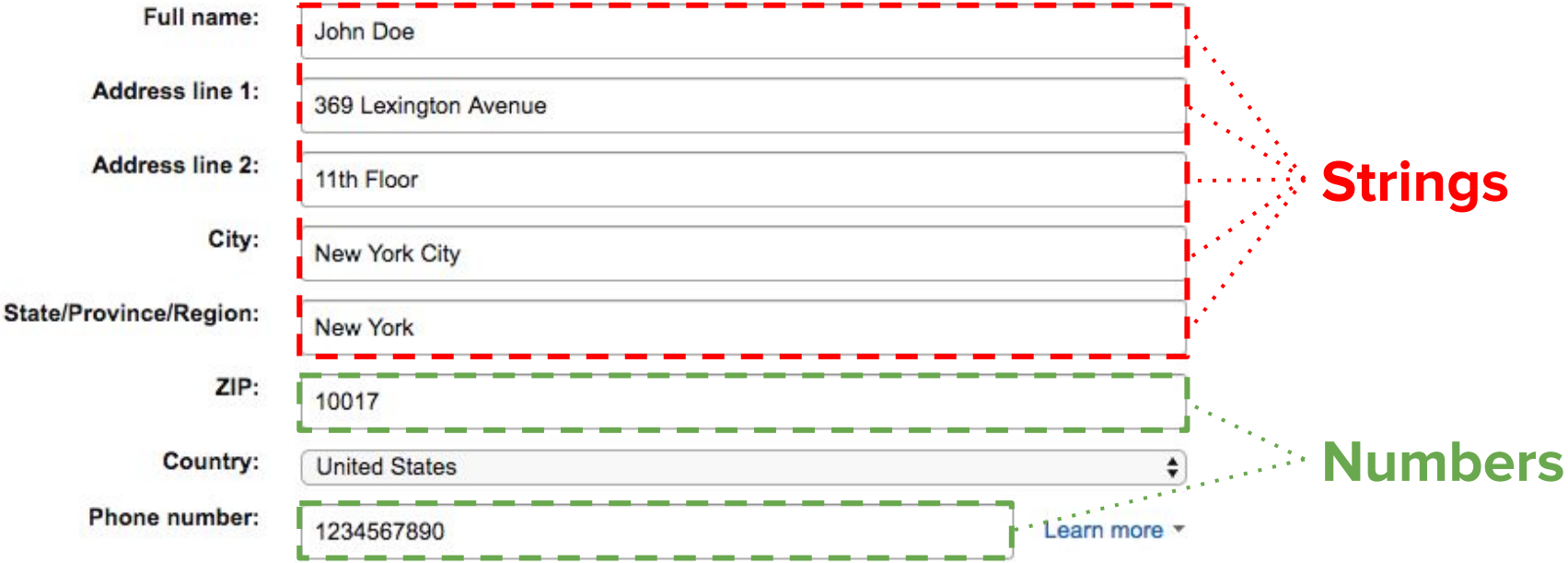
Different types of information

- When writing a JavaScript program, you will inevitably need to store different kinds (or *types*) of information
- JavaScript cares deeply about the *type* of this information
- JavaScript will dictate what you can do with the information based on its *type*
- Think about the various fields required to complete a checkout on Amazon.com
  - Your Zip Code vs. Full Name
  - Different inputs are relevant to different types
  - Let’s take a look at an example

Different types of information

Full name:	John Doe
Address line 1:	369 Lexington Avenue
Address line 2:	11th Floor
City:	New York City
State/Province/Region:	New York
ZIP:	10017
Country:	United States
Phone number:	1234567890 Learn more

Different types of information



Different types of information

Full name:	John Doe
Address line 1:	369 Lexington Avenue
Address line 2:	11th Floor
City:	New York City
State/Province/Region:	New York
ZIP:	10017
Country:	United States
Phone number:	123-456-7890 Learn more

AUDIENCE POLL: Can this “phone number” be considered as a number by JavaScript?

Different types of information

Full name:	John Doe
Address line 1:	369 Lexington Avenue
Address line 2:	11th Floor
City:	New York City
State/Province/Region:	New York
ZIP:	10017
Country:	United States
Phone number:	123-456-7890

Learn more

It's now a string!

ANSWER: No. A type of number must be strictly comprised of integers.

Why do types exist?

Input:

```
22 + 4
```

Output:

```
26
```

Input:

```
22 + “Foo”
```

Output:

```
“22Foo”
```

Why do types exist?

Input:

```
2020 + “1”
```

Output:

```
“20201”
```

Input:

```
“2020” + “1”
```

Output:

```
“20201”
```

Part III: Variables in JavaScript

What is a variable?

A memory container for information.

Variables

- Variables, as memory containers, are used to persist (remember) information in a JavaScript program
- Persisting information is the foundation for making complex programs
- The intention of persisting information is so that you can continue to reference it throughout your entire program

Note: Your program will generally be comprised largely by variables and logic

“Declaring” a variable

- When you want a new variable, you need to “declare” it using JavaScript’s expected syntax

The Name

The Value

```
var myName = “John”
```

Keyword

Assignment Operator

“Declaring” a variable

The Name

The Value

```
var theYear = 2017
```

Keyword

Assignment Operator

Variable naming constraints

- When naming a variable, you must follow some rules:
  - Must start with a letter (or one of two special, allowed characters)
  - No spaces
  - Are case sensitive
  - Can only contain letters, numbers (and two special characters)
  - Must be uniquely named (unless you want to overwrite an existing variable)
- When a variable name doesn't fit these constraints, it's "illegal"

Note: The two special, allowed characters are a dollar sign ($) and an underscore (_)

Variable naming constraints

Based on the aforementioned constraints, can you identify which of these variable names is illegal?

1

```
var myName =
```

2

```
var % =
```

3

```
var hi_there =
```

Variable naming constraints

1

```
var myName =
```

LEGAL

2

```
var % =
```

ILLEGAL

% is not one of the special, allowed characters

3

```
var hi_there =
```

LEGAL

Variable naming constraints

Let's try again...

1

```
var x5 =
```

2

```
var 12 =
```

3

```
var phone# =
```

Variable naming constraints

1

```
var x5 =
```

LEGAL

2

```
var 12 =
```

ILLEGAL

Must start with a letter or one of two special characters

3

```
var phone# =
```

ILLEGAL

Can only contain letters, numbers and special characters

Variable naming convention

- When writing JavaScript code, it's important to keep in mind how "readable" your code is
- Contributing to your code's "readability" is the format of your variable names
- There are two commonly used naming conventions:
  - camelCase
  - snake_case
- Let's take a look at them

Naming Convention: camelCase

- The first word is lowercase
- All subsequent words have proper capitalization

```
var myName =
```

FOLLOWS CONVENTION

```
var myname =
```

DOESN'T FOLLOW CONVENTION

```
var Myname =
```

DOESN'T FOLLOW CONVENTION

Naming Convention: snake_case

- All letters are lowercase
- All words are separated by an underscore

```
var my_name =
```

FOLLOWS CONVENTION

```
var myname =
```

DOESN'T FOLLOW CONVENTION

```
var my_name_is =
```

FOLLOWS CONVENTION

Referencing variables

- After you've *declared* a variable, you can *reference* it by its exact name to *get* its value

Input:

```
var myName = "John"
myName
```

Output:

```
John
```

Console.log

- When you want to display information from your JavaScript program, you can use *console.log(variableName)*

Input:

```
var myName = “John”
console.log( myName )
```

Output:

```
John
```

Note: console.log is a function – we will learn more about these later

Re-assigning variable values

- Once you've *declared* a variable, you can *re-assign* the value of that variable

Input:

```
var myName = "John"
console.log( myName )
myName = "Bob"
console.log( myName )
```

Output:

```
John
Bob
```

Note: You don't need to include "var" during a re-assignment

Re-assigning variable values

AUDIENCE POLL: What will be the output?

Input:

```
var x = 10
var y = x
console.log( x )
console.log( y )
```

Output:

```
10
10
```

Re-assigning variable values

Let's try again, this time with something more complex..

Input:

```
var x = 10
var y = x
x = 15
console.log( y )
console.log( x )
```

Output:

```
10
15
```

AUDIENCE POLL: What will be the output?

Part IV: Operators in JavaScript

What is an operator?

A special character that represents an action.

Operators

- There are many different categories of operators
- Some common kinds of operators:
  - Assignment operators
  - Logical operators
  - Arithmetic operators
  - Comparison operators
  - String operators

Operators

- There are many different categories of operators
- Some common kinds of operators:
  - **Assignment operators (we've seen these already!)**
  - Logical operators
  - Arithmetic operators
  - Comparison operators
  - String operators

Operators

- There are many different categories of operators
- Some common kinds of operators:
  - Assignment operators
  - Logical operators
  - **Arithmetic operators (let's explore these!)**
  - Comparison operators
  - String operators

The Addition Operator (+)

- The addition operator doubles as an arithmetic operator and a string operator

Input:

```
console.log( 15 + 10 )
```

Output:

```
25
```

Input:

```
console.log( “John” + “Doe” )
```

Output:

```
JohnDoe
```

The Subtraction Operator (-)

Input:

```
console.log( 15 - 10 )
```

Output:

```
5
```

The Multiplication Operator (*)

Input:

```
console.log( 10 * 5 )
```

Output:

```
50
```

The Division Operator (/)

Input:

```
console.log( 10 / 5 )
```

Output:

```
2
```

The order of operations

Input:

```
console.log( ( 22 + 4 * 2 - 2 ) / 4 )
```

Output:

```
7
```

Remember: PEMDAS Applies!

(Parentheses, Exponents, Multiplication and Division, Addition and Subtraction)

Part V: Functions in JavaScript

What is a function?

A procedure or routine; a collection of repeatable code.

Functions

- In programming, there is a principle known as “DRY”
  - **D**on’t **R**epeat **Y**ourself
- Functions exist to:
  - 1: Bundle potentially-repeated code in a single place
  - 2: Encapsulate (compartmentalize) code away from the rest of your JavaScript program (for organization)
- Functions have the ability to take and return information from within
- Functions are treated as values in JavaScript

The anatomy of a function

Let's start by defining a variable...

```
var add =
```

The anatomy of a function

Name Keyword Parameters

```
var add = function( a, b ) {
  return a + b
}
```

Keyword

The anatomy of a function

This is the function's *"body"*, where all of the associated code is housed:

```
var add = function( a, b ) {
  return a + b
}
```

The anatomy of a function

A function body can have more than 1 line of code:

```
var add = function( a, b ) {
  var prefix = “Your sum is: ”
  return prefix + ( a + b )
}
```

The anatomy of a function

Functions aren't required to return information:

```
var add = function( a, b ) {
  var prefix = "Your sum is: " + ( a + b )
}
```

AUDIENCE POLL: Do you think this function is useful?

Functions

- Review:
  - Functions are treated as values in JavaScript, which is why we store them in variables
  - Functions can have *parameters* to take in information
  - Functions aren't required to return information
- Functions are first *declared* and then optionally *invoked*
- Functions are *invoked* by referencing the name and attaching the invocation operator after: **add ( )**
- A function can be declared *once* and be used *infinitely*

Function invocation

```
var add = function( a, b ) {  1
  return a + b  2
}

add( 4, 6 )  3
```

AUDIENCE POLL: Where is the invocation happening?

Function invocation

```
var add = function( a, b ) {  1
  return a + b  2
}

add( 4, 6 )  3
```

ANSWER: #3 is where we are referencing the name of the function followed by the invocation operator.

Function re-use case

```
var add = function( a, b ) {
  return a + b
}

add( 4, 6 ) // 10
add( 10, 8 ) // 18
add( 46, 4 ) // 50
```

IMAGINE: The adding of two numbers was a greatly complex operation.

Functions

- Review:
  - Functions are first *declared* and then optionally *invoked*
  - A function can be declared *once* and be used *infinitely*
- *Parameters* act as “placeholders” for information to be passed through
- *Arguments* are the “actual” values that are passed through a function upon *invocation*
- Both parameters and arguments are optional

Function parameters vs. arguments

Parameters

```
var add = function( a, b ) {
  return a + b
}

add( 4, 6 )
```

Arguments

Pre-existing Functions

- JavaScript has many pre-existing functions included that you can leverage in your programs
- Let’s explore some of the arithmetic-related functions

Math.random()

Returns a random number between 0.00 and 1.00

Input:

```
Math.random()
```

Output:

```
.900939112631665
```

Math.random()

Returns a random number between 0.00 and 10.00

Input:

```
Math.random() * 10
```

Output:

```
9.00939112631665
```

Math.random()

Returns a random number between 0.00 and 100.00

Input:

```
Math.random() * 100
```

Output:

```
90.0939112631665
```

Math.round(*number*)

Returns the rounded version of the provided number

Input:

```
Math.round(4.6)
```

Output:

```
5
```

Math.floor(*number*)

Returns a rounded-down version of the provided number

Input:

```
Math.floor(4.6)
```

Output:

```
4
```

Math.max(*number, .., .., ..*)

Returns the largest of the provided numbers

Input:

```
Math.max(23, 18, 14, 24, 31, 42)
```

Output:

```
42
```

Exercises

https://github.com/ChrisHuie/WebDevWorkshop