Recap the Basics

Reading and Writing Javascript

Intro

I'm Kyle Pace and I work at Olo.

I've been using javascript for 8 or so years for server-side code, front-end apps and mobile applications.

This week is primarily focused on the building blocks of javascript as a programming language and a refresher for things we went over last summer.

https://github.com/kylepace

@therealkylepace

Overview

1. Primitive types and Variables
2. Arrays
3. Conditional Statements
4. For Loops
5. Functions
6. Objects

Primitive Types

Every programming language is made up of primitive types. Javascript has five types.

1. String - `“Hello World”`
2. Number - `67 48.67 -12`
3. Boolean - `true / false`
4. Null - `null (having no value)`
5. Undefined - `? (never having been assigned)`

Variables

Variables are used to store primitive types or more complex objects for use later on.

They can also used to communicate the intent of a program.

```
var firstName = ‘John’;

var lastName = ‘Doe’;

var age = 30;

var willGoSkydiving = false;
```

Variables

Once assigned, variables can be modified to have different values.

```
var legalLimit = 18;

var canVote = false;

var currentAge = 19;

if (currentAge > legalLimit) {

    canVote = true;

}
```

Variables - Arithmetic Operators

Once you've defined some variables, you can perform basic comparative or additive operations using javascript *operators.*

```
var timesClicked = 11;

timesClicked = timesClicked + 1;

console.log(timesClicked); // 12

timesClicked = timesClicked + 6;

console.log(timesClicked); // 18
```

Variables - Arithmetic Operators

The subtraction operator behaves the same way as addition.

```
var currentYear = 2016;

var yearsSinceMillenium = 16;

var millenium = currentYear - yearsSinceMillenium;

console.log(millenium); // 2000
```

Variables - Arithmetic Operators

Remaining arithmetic operators

* - multiplication => `var minutesInDay = 60 * 24;` `// 1440`

/ - division => `var hoursFromMinutes = 120 / 60;` `// 2`

%- modulus => `var remainder = 3 % 2;` `// 1`

Variables - String Concatenation

You can combine strings together by using the same addition operator (+) used to add two numbers together.

```
var firstName = “John”;

var lastName = “Doe”;

var fullName = firstName + ‘ ‘ + lastName;

console.log(fullName); // John Doe
```

Variables - String Concatenation

Since javascript is not *strongly typed*, adding a string and a number is allowed and together will result in a new string.

```
var myAge = 30;

var ageToAdd = “34”;

var newAge = myAge + ageToAdd;

console.log(newAge); // 3034
```

Arrays

You can store multiple variables in list-like objects called *arrays*.

Think of arrays as *lists* of objects.

```
var paintColors = [‘Red’, ‘Yellow’, ‘Green’];
```

Arrays can be made up of any javascript object.

```
var responses = [true, 1, ‘Marble’];
```

Arrays

You access objects in an array by using *zero-based array indexing*.

Zero-based indexing just means the first item in an array is found by using the integer 0.

```
var airlines = [‘Delta’, ‘United’, ‘Jet Blue’];

console.log(airlines[0]); // Delta

console.log(airlines[1]); // Jet Blue
```

Arrays

Once you have a collection of items in an array, you can add or remove objects. Use the *push* method to add another item to an array.

```
var baseballTeams = [‘Mets’, ‘Yankees’];

baseballTeams.push(‘Dodgers’);

console.log(baseballTeams); // [‘Mets’, ‘Yankees’, ‘Dodgers’]
```

push adds the object to the end of the array.

Arrays

Removing an item from an array is a little more tricky. Use *splice* to remove items from an array.

```
var companies = [‘Google’, ‘Yahoo’, ‘Facebook’];

companies.splice(2, 1);

console.log(companies); // [‘Google’, ‘Yahoo’]
```

splice removes items from an array beginning at the index in the first parameter for as many items specified in the second parameter.

Arrays - String

Fun fact: you can access an individual character in a string by using array indexing.

```
var myName = ‘Kid Rock’;

console.log(myName[4]); // R

console.log(myName[1]); // i
```

Conditional Statements - If-Statements

An *if-statement* is a boolean condition that, if true, will execute the code inside of a code-block.

```
var ageLimit = 19;

if (ageLimit > 17) {

    console.log('You can vote.');

}
```

Conditional Statements - If-Statements

Any boolean value can be used in a conditional statement.

```
var ageLimit = 19;

var canVote = 19 > 17;

if (canVote) {

    console.log('You can vote.');

}
```

Conditional Statements - Operators

In addition to the greater than operator (>) you can use the following other comparison operators in if-statements.

`>=` greater than or equal to

`<` less than

`<=` less than or equal to

`===` equal to

`!==` not equal to

Conditional Statements - Combine

You can combine more than one comparison in an if-statement with the help of *logical operators*.

```
var age = 17;

if (20 > age && age > 3) {

    console.log('this will run');

}
```

Conditional Statements - Logical Operators

The logical operators in javascript include:

&&	AND	-	both statements must be true
||	OR	-	any statement can be true
!	NOT	-	used to negate a statement

Conditional Statements - If-Else

If a condition is not met, then a code block within an else-statement will be executed instead.

```
if (19 > 17) {

    console.log('You can vote.');

} else {

    console.log('Not allowed to vote, sorry.');

}
```

Looping - For Loops

Loops execute the same block of code over and over again until a condition occurs.

```
for (var i = 0; i < 10; i++) {

    console.log(‘Going to print this ’ + i + ‘ times‘);

}
```

Looping - For Loops

There are three necessary statements in for-loops.

1. *Initialization* statement `var i = 0;`
2. *Condition* statement `i < 10;`
3. *Increment/Decrement statement* `i++;`

The condition statement can be any boolean condition, if true the loop will then run the increment/decrement statement.

If the condition is false then the program will move to the next line of code after the code block.

Looping - For Loops

When iterating through a javascript array, you will commonly need to use a loop. Use the loops current iterator as the index to the array.

```
var cars = [‘Ford’, ‘Chevy’, ‘Buick’];

for (var i = 0; i < cars.length; i++) {

    console.log(cars[i]);

}
```

Functions - Where things start to get difficult....

Functions are blocks of code designed to reuse to same code or abstract a complex set of operations.

```
function add(x, y) {

    return x + y;

}
```

Functions

A function is made up of the *function* keyword, parameters, and an optional return statement.

```
function subtract(x, y) {

    return x - y;

}
```

Here, `x` and `y` are the parameters passed into the function.

Functions

You run, or *invoke*, a function by using open and closed parenthesis `()`. Any parameters belong inside the parenthesis separated by commas.

```
function subtract(x, y) {

    return x - y;

}

var result = subtract(6, 3);

console.log(result); // 3
```

Functions

Pass parameters into a function in order to share data or operate on specific values.

```
var finalValue = subtract(5, 3); // 2

function subtract(x, y) {

    return x - y; // 5 - 3

}
```

Functions

Functions stop running when the *return* keyword is used. Return can either be used to abruptly stop processing or to output a value.

```
function getFirstName() {

    return ‘John’;

}

var myFirstName = getFirstName();

console.log(myFirstName); // John
```

Functions

Managing complexity is a primary concern for functions, and a function can be as long as is necessary (but generally should only do one thing really well).

```
function padName(name) {
    if (name.length < 5)
        return name + '----------';
    else
        return name;
}
```

Functions - Scope

Variables created inside a function cannot be accessed outside of a function.

```
function scopeTesting() {

    var name = ‘John’;

    return name + ‘Doe’

}

console.log(name); // undefined
```

Functions - Scope

Variables created outside of a function *can* be accessed inside a function, if they share the same parent function.

```
var name = ‘peter’;

function echoPeter() {

    return name + ‘ JR’;

}

echoPeter(); // peter JR
```

Objects

Objects are ways of grouping information together to make sense of complex applications.

```
var person = {

    firstName: ‘John’,

    lastName: ‘Doe’

};
```

Objects

Create an object using curly-braces { } followed by a semicolon.

```
var car = {
    make: ‘Jeep’,
    model: ‘Grand Cherokee’
};
```

Objects

Objects are made up of properties and functions. Functions assigned to objects can also be called methods.

```
var dog = {

    name: ‘Fido’,

    speak: function () {

        console.log(‘woof’);

    }

}
```

Objects

Once an object is created, you can access its properties by using *dot-notation*.

```
var person = {

    firstName: ‘John’

};

console.log(person.firstName);
```

Objects

Access object methods in the same way as it's properties, just make sure to use parenthesis to invoke the function.

```
var dog = {

    speak: function () {

        console.log('woof');

    }

}

dog.speak();
```

Basics

Javascript is a programming language that, up until a few years ago, is primarily run in a web browser.

In order to run javascript in a browser, you need to put any code into script blocks.

```
<script type=”text/javascript”>

    // Your code here

</script>
```

Basics

When using a browser to do development, you'll find you want to debug or print out values to find out what's going on in your program.

```
console.log("Hello World");
```

Logging to the console is one tool to help diagnose the value of variables when making an application run properly.

Group Exercises

Tips

- Use google or stackoverflow, but only if you are stuck and don't copy the answer.
- Ask for help as soon as you start to get frustrated.

Exercises

- Infamous padleft - together
- String Reverse - write a function that takes a word and outputs its reverse
- Sum of a list - given an array of numbers, output the sum of the numbers

External Learning Material

- https://www.codecademy.com/ - Good for Beginners
- http://www.w3schools.com/js/ - Great Reference
- https://egghead.io/ - Intermediate to Advanced free-ish training
- https://www.udemy.com/learn-javascript-for-beginners/learn/v4/overview