1. Numbers, Strings, and Operators

Floating point numbers are approximations

- `0.1 + 0.2 // = 0.30000000000000004`
  - 0.1 + 0.2 is not 0.3, but 0.30000000000000004 because floating numbers are approximations.
  - This is not a particular JavaScript issue, but the floating point number issue in general.

Shifting a number is multiplying or dividing by 2

- `1 << 2; // = 4`
  - It is 4, as 1 is multiplied by 2 (shift left) two times (the 2 in 1 « 2). It is the same as 2^2.
  - 1 « 3 is 8, because $2^3 = 8$, and 1 « 4 is 16.

JavaScript is a weakly typed language.

JavaScript will change the type if necessary

- `"1, 2, " + 3; // = "1, 2, 3"`
  - JavaScript changes the number 3 into a string "3"
- `"Hello " + ["world", "!"]; // = "Hello world,!"`
  - JavaScript changes the array into a sequence of strings.
- `13 + !0;` // 14
  - !0 is interpreted 1 because it is added to a number
- "13" + !0; // '13true'
  - !0 is interpreted true as it is concatenated to a string.

Use === not ==

- `"5" == 5`; // = true
  - This is dangerous, use === instead.
  - `"5" === 5`; // = false
- `null == undefined`; // = true
  - This is also dangerous, use === instead
  - `null === undefined`; // = false

JavaScript copies some Java methods with a twist

- "This is a string".charAt(0); // = 'T'
- "Hello world".substring(0, 5); // = "Hello"
- "Hello".length; // = 5
  - `length` is a property in JavaScript, so don't use () like Java.

Falsy and Truthy with a twist

- false, null, undefined, NaN, 0 and "" are falsy; everything else is truthy.
- Note that 0 is falsy and "0" is truthy, even though 0 == "0".

2. Variables, Arrays and Objects

shift (delete first) and unshift (add last)

```
// ['Hello', 45, true, 'Hello']
myArray.shift(); // shift is delete the first element
// => [45, true, 'Hello']
myArray.unshift(3); // Add as the first element
// => [3, 45, true, 'Hello']
someVar = myArray.shift(); // Remove first element and return it
// => [45, true, 'Hello']
```

push (add last) and pop (delete last)

```
myArray.push(3); // Add as the last element
// => [45, true, 'Hello', 3]
someVar = myArray.pop(); // Remove last element and return it
// => [45, true, 'Hello']
```

JavaScript list can have hybrid values

- var myArray0 = [32,false,"js",12,56,90];
- myArray0.join(";"); // = "32;false;js;12;56;90"

slice and splice

- myArray0.slice(1,4); // = [false,"js",12] <- 1,2,3, not 4
- myArray0.splice(2,4,"hi","wr","ld");
  - [32, false, "js",12,56,90] => [32,false,"hi","wr","ld"]
  - remove elements from 2 to 2 + 4, and add three more elements.

JavaScript Object is not exactly the same as JSON

Two ways to access the value in JavaScript Object

- Keys are strings, but quotes aren't required if they're a valid JavaScript identifier. Values can be any type.
- To access values, we can use the bracket [...] or the " notation.

```
var myObj = {key1: "Hello", "key2": "World"};
console.log(myObj["key1"]) // give string as a key
console.log(myObj.key2) // give name as a property
```

- When the key has a space in it, it should be a string.

```
var myObj = {myKey: "myValue", "my other key": 4};
myObj["my other key"]; // = 4
```

There is no exception when accessing unset values

- myObj.myKey; // = "myValue"
- console.log(myObj['myKey'])
- myObj.myFourthKey; // = undefined
  - If you try to access a value that's not yet set, you'll get undefined.
  - Compare Python and JavaScript.

```
// Python
>>> a['a']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'a'
// JavaScript
> a = {}
{}
> a['a']
undefined
```

3. Logic and Control Structures

- The for loop is similar to Java's for loop.

```
for (var i = 0; i < 5; i++){
    // will run 5 times
    console.log(i)
}
```

- It's not a good coding habit, but we can break out of the multiple loop using a label. It is recommended to use a function and return from the function, instead of break using a label.

```
outer:
for (var i = 0; i < 10; i++) {
    for (var j = 0; j < 10; j++) {
        if (i == 5 && j ==5) {
            break outer;
            // breaks out of outer loop instead of only the inner one
        }
    }
}
```

- We can use for each loop.

```
for (var x in person){
    description += person[x] + " ";
} // description = 'Paul Ken 18 '
```