Mobile Application Development

Introduction to JavaScript

Interpreted Language

- Each browser has its own JavaScript engine, which either interprets the code, or uses some sort of lazy compilation
  - V8: Chrome and Node.js
  - SpiderMonkey: Firefox
  - JavaScriptCore: Safari
  - Chakra: Microsoft Edge/IE

- They each implement the ECMAScript standard, but may differ for anything not defined by the standard

Syntax

- Declarations
  - Var
  - Let
  - Const
- Variables
  - Same as most other languages
- Declaring variables
  - Global and local scope
- Literals {}, [], ‘ ’, 1, 2

Datatypes

- Dynamic typing
- Primitive types
  - undefined
  - null
  - boolean
  - number
  - string
  - symbol
- Objects

Typecasting? Coercion.

- Explicit vs. Implicit Coercion

- == vs ===

Primitives Vs. Objects

- Primitives are immutable
- Objects are mutable and stored by reference
  - Passing by reference vs. passing by value

Scoping

- Lexical
- Block

Array Functions

- Filter
  - The filter() method creates a new array with all elements that pass the test implemented by the provided function

```
const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];
const result = words.filter(word => word.length > 6);
```

- Map
  - The map() method creates a new array populated with the results of calling a provided function on every element in the calling array.

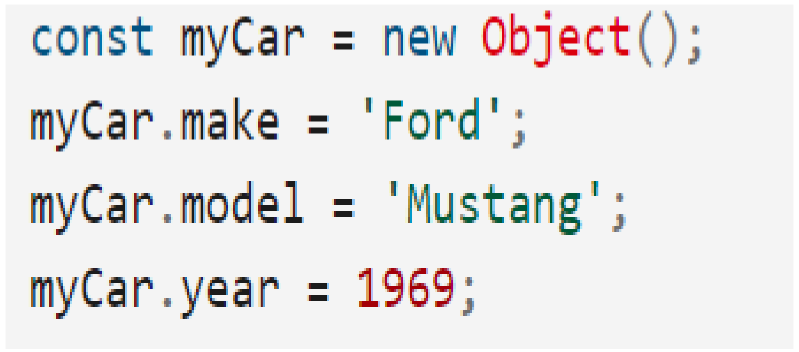
```
const array1 = [1, 4, 9, 16
map1 = array1.map(x => x * 2);
```

Array Functions

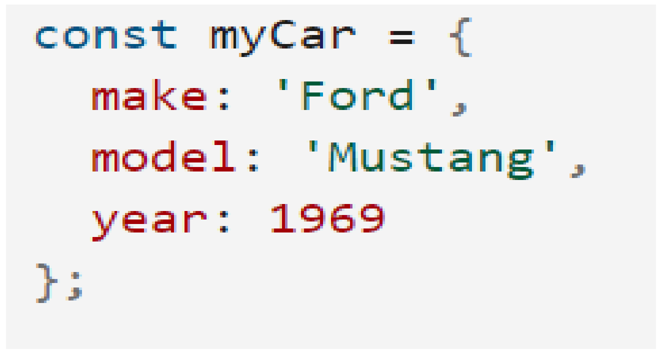
- Reduce
  - Arr.reduce(function, initial value)
  - Return a single

Objects

- An object is a collection of properties, and a property is an association between a name (or *key*) and a value.
- Creation



```
const myCar = new Object();
myCar.make = 'Ford';
myCar.model = 'Mustang';
myCar.year = 1969;
```



```
const myCar = {
  make: 'Ford',
  model: 'Mustang',
  year: 1969
};
```

Objects

- Constructor
  - Define the object type by writing a constructor function. use a capital initial letter(convention).
  - Create an instance of the object with new.

```
function Car(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
}
```

```
const myCar = new Car('Eagle', 'Talon TSi', 1993);
```

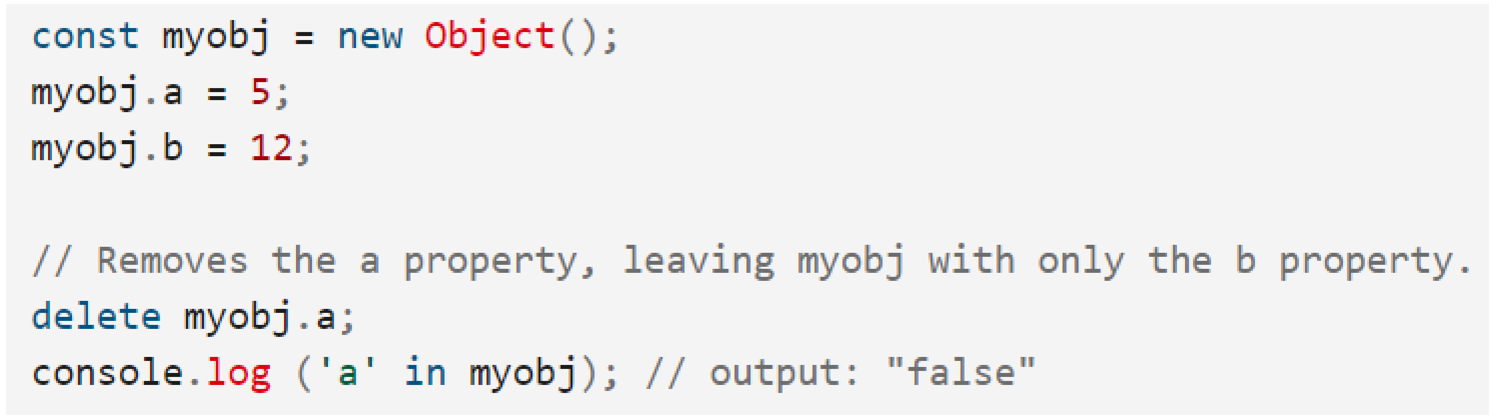
Objects

- Use of this
    - use within a method to refer to the current object.

```javascript
const Manager = {
  name: "John",
  age: 27,
  job: "Software Engineer"
}
const Intern = {
  name: "Ben",
  age: 21,
  job: "Software Engineer Intern"
}

function sayHi() {
  console.log(`Hello, my name is ${this.name}`)
}

// add sayHi function to both objects
Manager.sayHi = sayHi;
Intern.sayHi = sayHi;

Manager.sayHi(); // Hello, my name is John'
Intern.sayHi(); // Hello, my name is Ben'
```

Objects

- Deleting properties
  - can remove a non-inherited property



```
const myobj = new Object();
myobj.a = 5;
myobj.b = 12;

// Removes the a property, leaving myobj with only the b property.
delete myobj.a;
console.log ('a' in myobj); // output: "false"
```

Objects

- Comparing objects
  - objects are a reference type.
  - distinct objects are never equal, even if they have the same properties.
  - Only comparing the same object reference with itself yields true.

```
const fruit = {name: 'apple'};
const fruitbear = fruit;  // Assign fruit object reference to fruitbear

// Here fruit and fruitbear are pointing to same object
fruit == fruitbear; // return true
fruit === fruitbear; // return true

fruit.name = 'grape';
console.log(fruitbear); // output: { name: "grape" }, instead of { name: "apple" }
```

```
const fruit = {name: 'apple'};
const fruitbear = {name: 'apple'};

fruit == fruitbear; // return false
fruit === fruitbear; // return false
```

References

- https://www.w3schools.com/js/default.asp
- https://developer.mozilla.org/en-US/docs/Web/JavaScript