INTRODUCTION TO JAVASCRIPT
var empty_object = {};

var student = {
    "first name": "Daniel",
    "last name": "Graham",
    title: "PHD",
}

It is ok to include or exclude trailing commas.
Object can even contain other objects

```javascript
var teacher = {
    name: "Daniel Graham",
    student: {
        age: 21,
        name: "John Stewart",
        grade: 86.2,
    }
}
```

JSON (JAVASCRIPT OBJECT NOTATION)
JavaScripit Object Retrieval

```javascript
console.log(student["first name"])  
console.log(student.title)  
console.log(teacher.student.name)
```
What happens if you reference a property that is apart of the object

```javascript
console.log(teacher.salary)
```

**THROWS AN UNDEFINED ERROR**

```javascript
console.log(teacher.salary.raise)
```

```javascript
console.log(teacher.salary.raise)
^  
TypeError: Cannot read property 'raise' of undefined
  at Object.<anonymous> (/Users/dgg6b/Documents/Classes/MobileApplicationDevelopment/00-Lecture/Code/objects.js:26:28)
  at Module._compile (internal/modules/cjs/loader.js:721:30)
  at Object.Module._extensions..js (internal/modules/cjs/loader.js:732:10)
  at Module.load (internal/modules/cjs/loader.js:620:32)
  at tryModuleLoad (internal/modules/cjs/loader.js:563:12)
```
JAVASCRIPT SYNTAX OBJECTS

• Updating properties: Properties can be updated by assignment.

```javascript
teacher['name'] = 'Daniel Graham PhD'
```

• New properties can also be added.

```javascript
teacher['purpose'] = 'Thinking & Creating'

teacher.affiliation = 'UVA'
```
It is also possible to update nested objects:

```javascript
teacher.student = {
    age: 22,
    grade: 97.2,
    year: 2,
    name: "John Stewart"
}

teacher.student.rank = 1
```
Objects are passed by reference not value. Objects are never copied.

```javascript
person1 = {
    name: 'John',
    age: 32
}

person2 = person1
person2.name = 'Tom'

console.log(person1.name)
```

WHAT GETS PRINTED?

TOM  JOHN
```javascript
/** Copies all the the properties from person 1 into person3 **/
person3 = Object.assign({}, person1)

person3.name = 'Jill'
```

**Does a swallow copy**
```javascript
husband = {
    name: 'Daniel',
    child: {
        name: 'Ruth',
        sex: 'Female',
        age: 1,
    }
}

// Shallow Copy
wife = Object.assign({}, husband)
wife.name = 'Shea'
wife.child.name = 'George'

console.log(husband.child.name)
```

RUTH

GEORGE
Sallow copies only copy references to nested objects
Correct way to deep copy in javascript

```javascript
wife = JSON.parse(JSON.stringify(husband));
```

Write your own

Part of Project 0 homework
JAVASCRIPT SYNTAX OBJECTS

- Javascript is a prototype based language (aka Classless) while Java or C++ are class based languages
- In Class-based languages new methods cannot be added at run-time or when the object is declared
- Each object is Javascript is created from a prototypical object (Object.prototype). This means that methods can be added or removed at run-time.
“Objects inherit from objects. What could be more object oriented than that?”

–DOUGLAS CROCKFORD
JAVASCRIPT SYNTAX OBJECTS

• Why are there no constructors?

• It is like the objects are created ex nihilo ("from nothing")

• It javascript there is a root object (Object.prototype) which comes standard with javascript. Objects are recreated by inheriting from Object.prototype
JAVASCRIPT SYNTAX OBJECTS

- The {...} notation creates a new object by inheriting from Object.prototype standard Javascript object.

```javascript
superClass = {
    language: "Javascript",
    version: 2.1,
}

subClass = {
    fork: "UVAscript",
}

Object.setPrototypeOf(subClass, superClass)

console.log(subClass.fork + ' ' + subClass.language)
```

PRINTS: UVASCRPT JAVASCRIPT
```javascript
/**
 * Demonstrate the class hierarchy
 */
oneUp = Object.getPrototypeOf(subClass)
oneUpList = Object.getOwnPropertyNames(oneUp)
console.log('One Up List
' + oneUpList)

//Get the prototype of superClass
twoUp = Object.getPrototypeOf(oneUp)
twoUpList = Object.getOwnPropertyNames(twoUp)
console.log('Two Up List
' + twoUpList)
```

**NOTE THE ROOT CLASS CONTAINS THE CONSTRUCTOR**
JAVASCRIPT SYNTAX OBJECTS

• Enumerating of the properties

```javascript
/** Enumerating all the properties of the subClass **/
for (prop in subClass) {
    console.log(prop)
}
```

Prints:
fork
language
version

INCLUDES THE PROPERTIES OF SUPERCLASS

There are no guarantees on the order
JAVASCRIPT DELETING PROPERTIES

• The delete operator can be used to delete properties from an object.

```javascript
delete subClass.fork
```

DELETES FORK PROPERTY FROM SUBCLASS

```javascript
delete subClass.language
```

DOES NOTHING BECAUSE LANGUAGE IS A PROPERTY OF SUPERCLASS
```javascript
superClass = {
  language: "Javascript",
  version: 2.1,
}

subClass = {
  fork: "UVAscript",
}

subSubClass = {
  fork: "UVAsubscript"
}

Object.setPrototypeOf(subSubClass, subClass)
console.log(subSubClass.fork)
delete subSubClass.fork
// Allows the superclass property to shine through
console.log(subSubClass.fork)
```

PRINTS UVASCRIP
PRINTS UVASUBSCRIPT
GLOBAL ABATEMENT

• Recall that variables declared with the var key word are available outside of the blocks in which it was defined.

• Nesting object inside of global object is one way to reduce unwanted interacts between variables. (Libraries etc)
NODE JS HAS A GLOBAL A OBJECT CALL GLOBAL THAT HOLDS ALL OFF THE VAR VARIABLES
JAVASCRIPT CLASSES

- The Classes Keywords was introduced in ECMAScript 2015. But has not induced a new inheritance model the prototype inheritance model still holds
FUNCTIONS IN JAVASCRIPT
“The Best thing about Javascript is its implementation of functions.”

–DOUGLAS CROCKFORD
FUNCTIONS ARE OBJECTS

• In Javascript functions are objects.

• These objects inherit from Function.prototype

• Function.prototype inherits from object.prototype
CREATING A SIMPLE FUNCTION

```javascript
function grow (last, increase)
{
    return last *(1 + increase)
}
```

```
console.log(grow(0.7, 0.1))
```

NAMED FUNCTION

PRINTS 0.77

WHEN FUNCTIONS ARE DECLARED OUTSIDE OF THE SCOPE OF CLASS WE NEED THE FUNCTION KEYWORD
FUNCTIONS ARE OBJECTS

SINCE FUNCTIONS ARE OBJECTS THEY CAN BE USED LIKE OTHER OBJECTS

career = function grow (lastyear, increase){
   return last*(1+increase)
};

console.log(career(0.7, 0.1))

Since this is an assignment statement you may sometimes see ; at the end
But ; aren’t necessary
ANONYMOUS FUNCTIONS

career = function(last, increase){
    return last*(1+increase)
}

console.log(career(0.7, 0.1))
ARROW FUNCTIONS
ARROW FUNCTIONS

let func = (arg1, arg2, ...argN) => expression

General Form of

let func = function(arg1, arg2, ...argN) {
  return expression;
}
ARROW FUNCTIONS

```javascript
multiply = (a, b) => a * b

// Equivalent to
multiply2 = function(a, b){
  return a * b
}

square = a => multiply(a, a)
```

SINGLE LINE ARROW FUNCTION
THE VALUE OF THE EXPRESSION IS RETURNED BY DEFAULT

Single parameters don’t require brackets
MULTIPLE LINE ARROW FUNCTIONS

\[(a, b) =>\{
  if (a >= b) {
    return a
  } else {
    return b
  }
}\]

bigger = (a, b) =>{
  return a >= b ? a : b
}\`

console.log(bigger(1, 3))

ARROW FUNCTIONS CAN BE ANONYMOUS
IIFE (IMMEDIATELY INVOKED FUNCTION EXPRESSION)

```javascript
!(function () {
    console.log('I called myself'
})()
```

IIFE ARE FUNCTIONS THAT CALL THEMSELVES

```javascript
(function() {
    console.log('I called myself too')
})();
```

RUNS IN V8 ENGINE BUT IN NODE

Older syntax for IIFEs
WEB API & REQUESTS
WHAT IS AN WEB API

- Application Programming Interface

Typical implemented
Using a MVC (Model View Control) framework
Django
ASP.net MVC

Alternative Lambda function (AWS)
Serverless solutions
WEB REQUEST & RESPONSE

CLIENT  

GET  POST

REQUEST  RESPONSE

SERVER

JSON
LET’S MAKE THIS CLEAR BY LOOKING AT SOME SAMPLE WEB APIS

https://github.com/toddmotto/public-apis

END POINT

https://catfact.ninja/#!/Facts/fact

DOCUMENTATION ON CAT FACTS

https://catfact.ninja/fact
CURL & BROWSER EXAMPLE

curl https://catfact.ninja/fact

HTTP/1.1 200 OK
SERVER: APACHE/2.2.14 (WIN32)
LAST-MODIFIED: WED, 22 JUL 2009 19:15:56 GMT
CONTENT-LENGTH: 88
CONTENT-TYPE: TEXT/HTML
CONNECTION: CLOSED
CURL & BROWSER EXAMPLE

curl https://catfact.ninja/fact

RESPONSE

GET /HELLO.HTM HTTP/1.1
USER-AGENT: MOZILLA/4.0
HOST: WWW.TUTORIALSPOINT.COM
ACCEPT-LANGUAGE: EN-US
ACCEPT-ENCODING: GZIP, DEFLATE
CONNECTION: KEEP-ALIVE
ASYNC FUNCTIONS

JAVASCRIPT
ASYNC/AWAIT

SPECIAL SYNTAX FOR PROMISES
ASYNC FUNCTIONS

ASYNC KEYWORD

```javascript
async function simpleFunction() {
    return 1;
}
```

Async keyword means that function returns a promise.
Async keyword means that function returns a promise

```javascript
async function simpleFunction() {
    return 1;
}
```

But this does not return a promise

Async keyword means that function returns a promise
ASYNC FUNCTION WEBREQUEST FETCH
ARRAYS IN JAVASCRIPT
MAP, REDUCE, FILTER

FUN WITH ARROW FUNCTIONS
PRINT ALL THE IDS

```javascript
students = [{grade:93, name: 'DeShea'},
            {grade: 96, name: 'Devin'},
            {grade: 90.0, name: 'Phylicia'}]

ids = []
for(let i =0; i < students.length; i++){
  ids.push(students[i].grade)
}

console.log(ids)
```

INTRODUCED THIS NEW SYNTAX OF ARRAY.push
CONCEPT OF MAP

Apply some function F to every value in the array
Apply some function $F$ to every value in the array $\{1, 4, 9, 16\}$.
```
var grades = students.map(function (student) {
  return student.grade
});

console.log("map grades "+ grades)
```

```
var grades = students.map(student => student.grade)

console.log("map grades "+ grades)
```
REDUCE VISUALIZATION

(((D+1)+2)+3)
REDUCE

```javascript
average = grades.reduce((sum, grade) => sum += grade, 0)/grades.length
console.log(average)
```

THE LAST PARAMETER IS ALWAYS THE VALUE ARRAY
REDUCE QUIZ

WRITE A REDUCTION FINDS THE HIGHEST SCORER

```javascript
highestGrade = students.reduce((highScorer, student) => {
    return (highScorer.grade || 0) > student.grade ? highScorer : student;
});

console.log(highestGrade)
```
FILTER

FUNCTION (X) => X > 2

Old Array

1 → F
2 → F
3 → F
4 → F

New Array

3
4

APPLY SOME FUNCTION F TO EVERY VALUE IN THE ARRAY
ONLY KEEP VALUES WHERE FUNCTION IS TRUE
FILTER

```javascript
above90 = students.filter(student => {
  return student.grade > 90
})
console.log(above90)

[ { grade: 93, name: 'DeShea' }, { grade: 96, name: 'Devin' } ]
```

**IF A MULTIPLE LINE ARROW FUNCTION IS USE IT MUST INCLUDE A RETURN STATEMENT**