JavaScript Introduction

http://goo.gl/PbYDp9

Why JavaScript?

- Widely used in web development
- Supported by almost all browsers (e.g. Chrome), to make websites respond to user interaction
- Also used by other web technologies, on the server side (e.g. Node.js, Express)

A Brief History

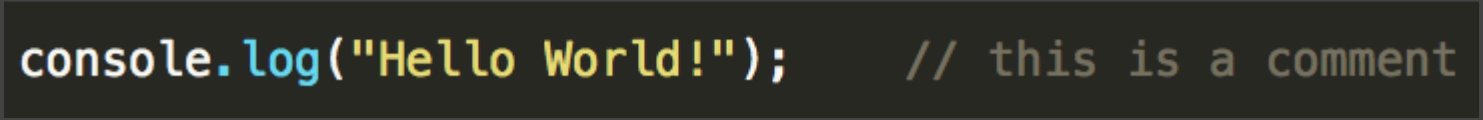
- Developed in 10 days by one person (Brendan Eich, working for Netscape) in 1995
- Not related to Java! Naming was a marketing move
- The language is often criticized...but it’s here to stay (for awhile)

What is JavaScript?

- Dynamically typed
  → you don’t specify types of variables
- Object-oriented
  → (almost) everything is an object!
- Supports first-class functions
  → functions are objects

Basics

- “print”: console.log(“Hello world”)
- comments: //, /* */



```
console.log("Hello World!");    // this is a comment
```

Data types

- Primitive types
  - immutable
  - number, string, boolean can have methods (are object-like)
- Everything else is an object

Primitive data types

- number
- string
- boolean
- `null`
- `undefined`

Number

- typeof(5)
- you can do math operations
- no integer/float distinction (all floats)
- parseInt(“123”, 10)
- parseFloat(“1.1”) (always base 10)
- NaN value (typeof(NaN) = ?)
- isNaN(10), isNaN(“hello”)

String

- no difference between string and character
- single or double quotes
- “hello”.length
- “hello”.charAt(0)

- “hello”.concat(“ world”)
- “hello” + “ world”

Variables

- a = 10;
- var a = 10;
- var a; // value of a is undefined

- always initialize variables with “var”

undefined, null

- **undefined**: type of a variable that has been defined, but does not have a value assigned to it
- **null**: type of a variable that has been defined and has as value of “no value”

CHECKPOINT

```
var foo;
foo;        // (1) value of foo?
foo = “hello”
foo;        // (2) value of foo?
foo = null;
foo;        // (3) value of foo?
```

Boolean

- has 2 values: true, false.

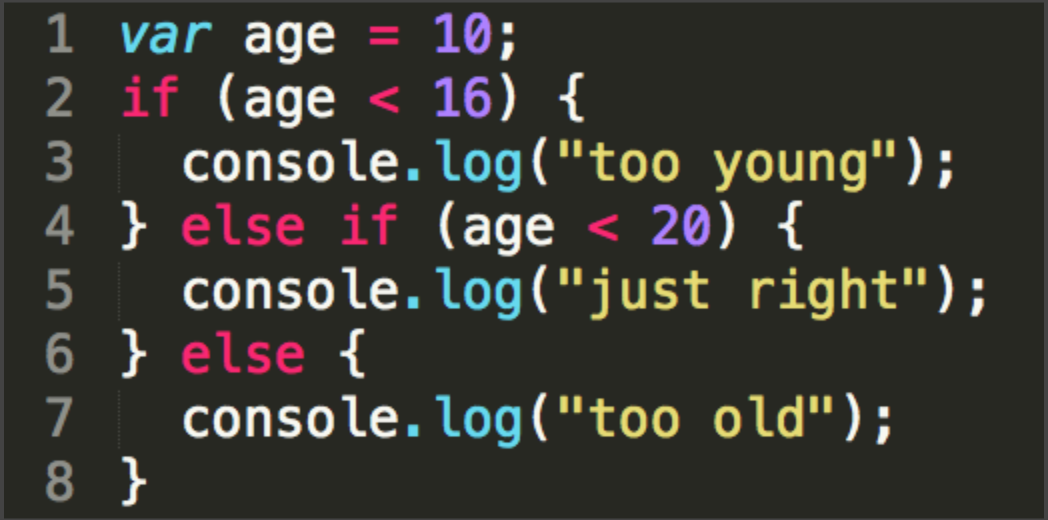
- falsy: false, 0, “”, NaN, null, undefined
- truthy: everything else

Comparison, logical operators

- >, >=, <, <=
- == performs type coercion (1 == true)
- Use === for equality comparison!

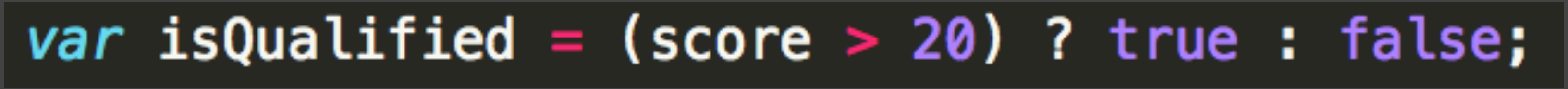
- ||, &&, !

if/else



```
var age = 10;
if (age < 16) {
    console.log("too young");
} else if (age < 20) {
    console.log("just right");
} else {
    console.log("too old");
}
```

Ternary operator



```
var isQualified = (score > 20) ? true : false;
```

CHECKPOINT (1)

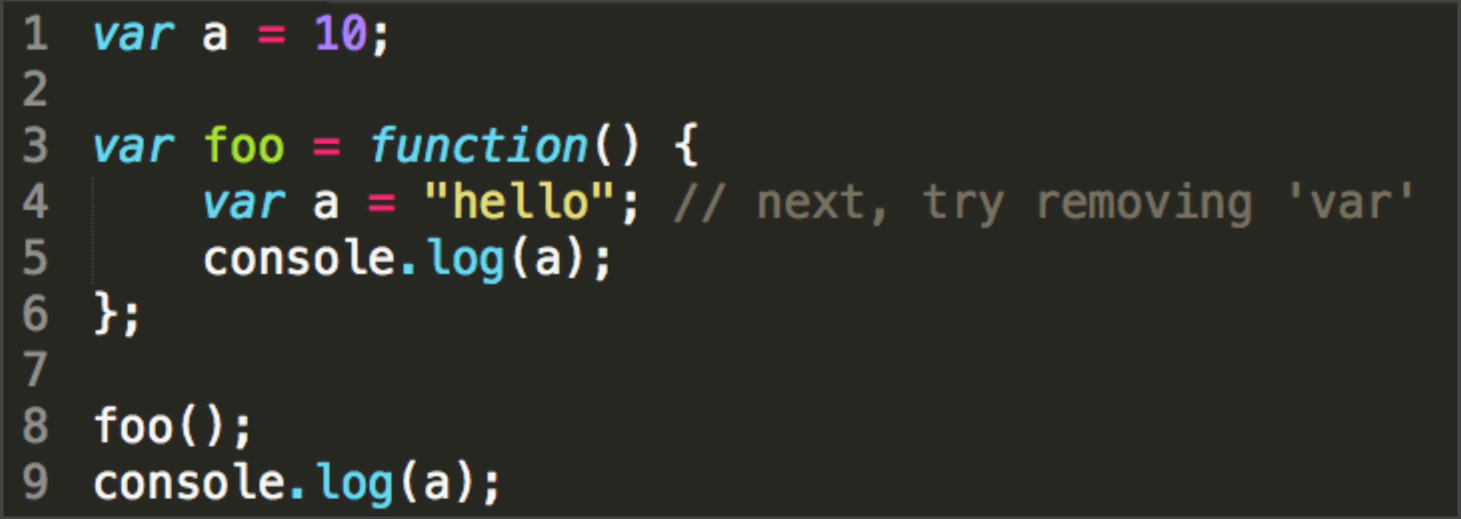
```
0 ? console.log(“truthy!”) : console.log(“falsy!”)      // (1)

1 ? console.log(“truthy!”) : console.log(“falsy!”)      // (2)

foo ? console.log(“truthy!”) : console.log(“falsy!”)  // (3)
```

CHECKPOINT (2)

```
var foo;
foo ? console.log(“truthy!”) : console.log(“falsy!”)      //(4)
var foo = 10;
foo ? console.log(“truthy!”) : console.log(“falsy!”)      //(5)
var foo = null;
foo ? console.log(“truthy!”) : console.log(“falsy!”)      //(6)
```

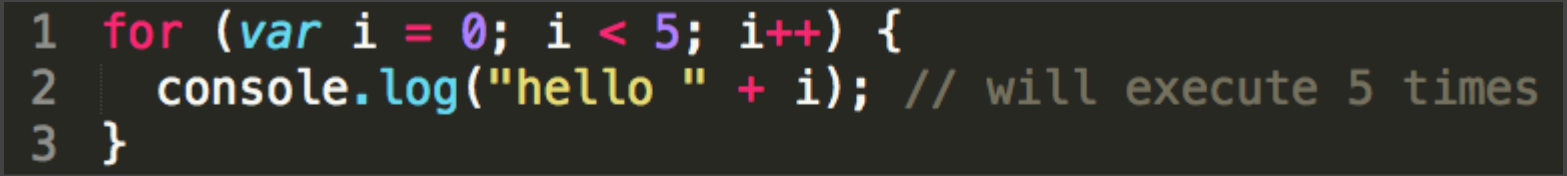
Variable Scope

- variables have global scope by default
- use keyword var so that variable is local
- (always try to minimize scope!)



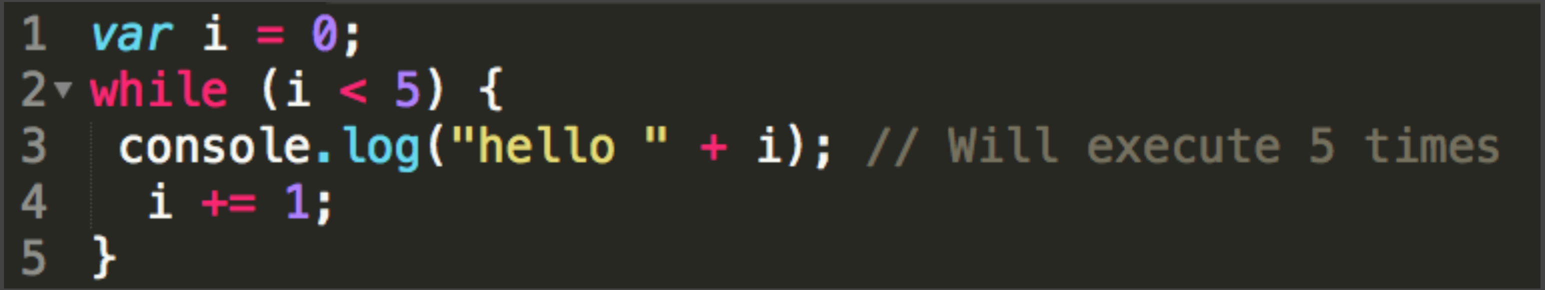
```javascript
var a = 10;

var foo = function() {
    var a = "hello"; // next, try removing 'var'
    console.log(a);
};

foo();
console.log(a);
```

for loop



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

while loop

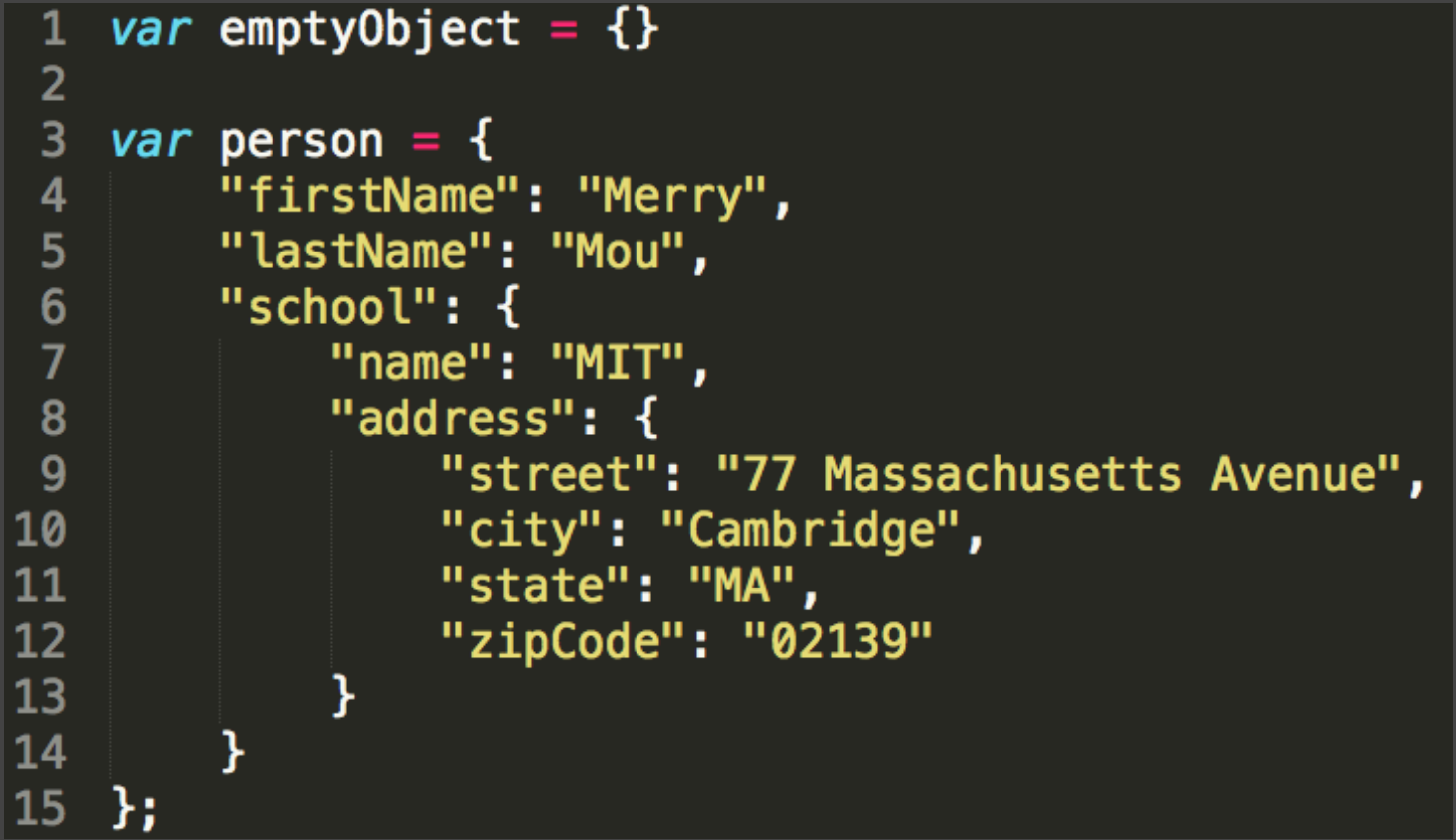


```javascript
var i = 0;
while (i < 5) {
  console.log("hello " + i); // Will execute 5 times
    i += 1;
}
```

Objects (1)

- Everything except primitive types are objects
- An object is a collection of properties
- A property has a name and a value

Objects (2)



```javascript
var emptyObject = {}

var person = {
    "firstName": "Merry",
    "lastName": "Mou",
    "school": {
        "name": "MIT",
        "address": {
            "street": "77 Massachusetts Avenue",
            "city": "Cambridge",
            "state": "MA",
            "zipCode": "02139"
        }
    }
};
```

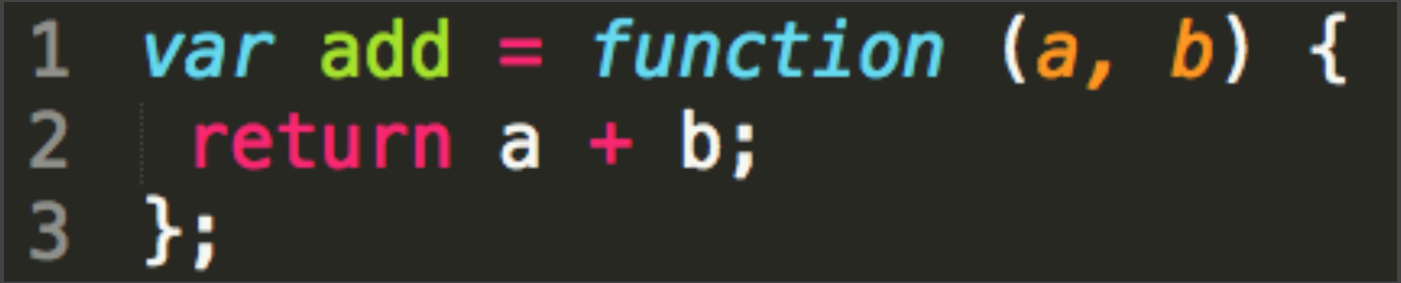
Objects (3)

- person.firstName
- person["firstName"]
- person.school.name
- person.age // undefined
- person.age = 20

Arrays

- typeof([1,2,3]) // “object”
- var myList = [“a”, “b”, 10, 20]
- myList[2]
- myList[10] // undefined
- myList[10] = “hi”
- myList
- myList.length

Functions (1)



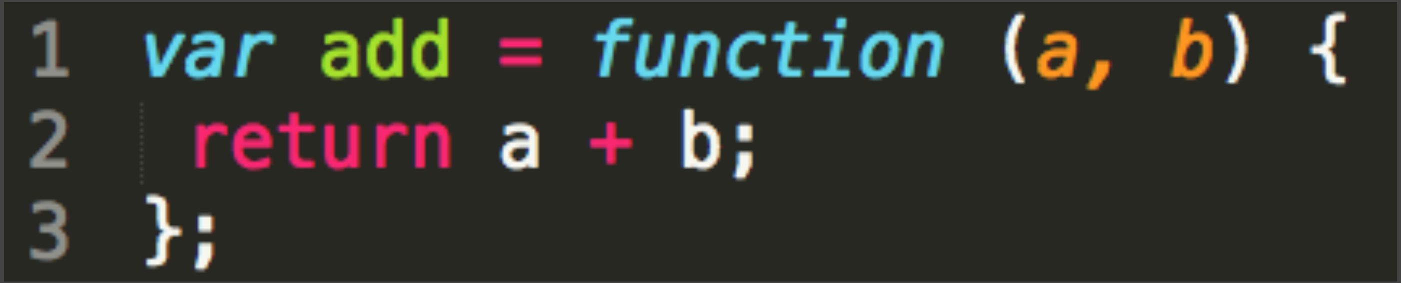
```
var add = function (a, b) {
  return a + b;
};
```

- var sum1 = add(1,2)
- var sum2 = add(add(1,2), add(5,5))

Functions (2)

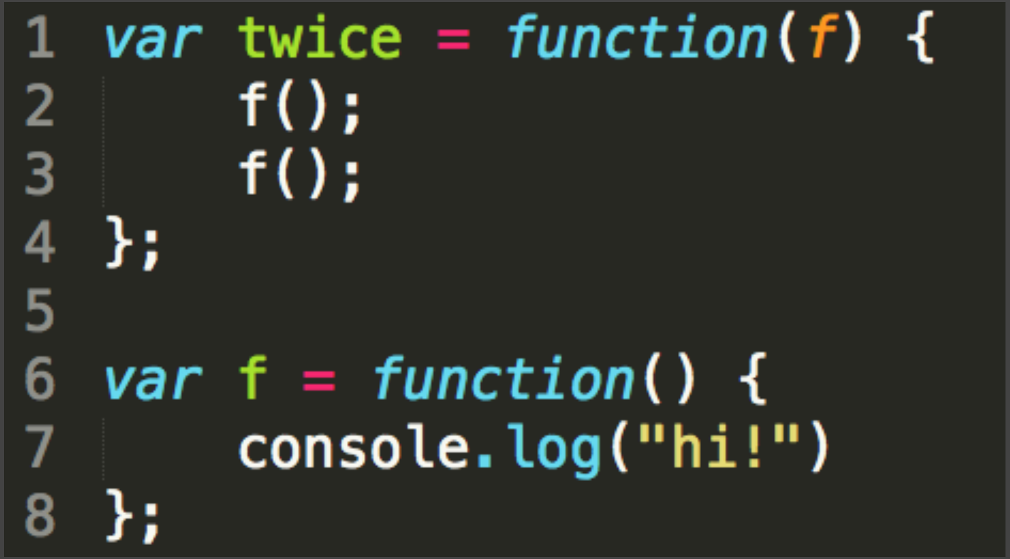
- functions are first-class objects: **treat them like objects**
- you can
  - store in variables
  - pass as arguments to functions (callbacks!)
  - create within functions
  - return from functions

First-Class Functions: store in variables



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

First-Class Functions: pass as arguments



```
var twice = function(f) {
	f();
	f();
};

var f = function() {
	console.log("hi!")
};
```

- f() vs f
- twice(f)

CHECKPOINT

Here is a JSON object with user information: https://gist.github.com/mmou/2435dac036c7b1dc02a7

(1) What is the average age of all the users?
(2) How many users are active and female?
(3) How many users have a balance >= $2000?
(4) What is the most common favorite fruit?

JSON Introduction

- JavaScript Object Notation.
- JSON is a syntax for easy storing and exchanging data.
    - easy to convert between string and JavaScript object
- **It is basically a JavaScript object.**

List comprehensions

- Let’s avoid using for loops (functional programming!)

- arr.forEach(callback[, thisArg])
- arr.map(callback[, thisArg])

CHECKPOINT

Example: How many users have <u>blue eyes</u>?

References

Mozilla Introduction

https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

JavaScript: The Good Parts

https://www.acmi.net.au/media/12347/javascript_the_good_parts.pdf

Google style guide https://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml

For fun

https://www.destroyallsoftware.com/talks/wat
(start 1:20)

How to create a new object

avoid new/this keywords