GENERAL ASSEMBLY

FUNCTIONS & OBJECTS

Felix Cohen

AGENDA

- **Review**
- **JavaScript Objects**
  - Creating Objects and attributes.
  - JSON: Saving objects in strings.
- **Functions**
  - Understand how functions take input & create output
  - Write a Function
  - Call a Function

REVIEW

- Is there homework or topics you would like to review?

JAVASCRIPT OBJECTS

OBJECTS

- Objects help us to build maintainable, understandable code.

- Programming is the art of describing real world problems in code

- Objects let us model real world things in a useful way.

- Like the car.

EXAMPLE OF AN "OBJECT"

- Person
  - Has a name, age, location
  - Can speak, eat.

- Lightbulb
  - Number of watts, brand
  - Can turn on and off

OBJECTS - FORMAL DEFINITION

- An “object” in computer science is a collection of data and functions that work with that data.

- Objects allow us to organize similar data effectively

JAVASCRIPT OBJECTS – SYNTAX

Empty object

```
{}
```

Person Object

```
var person = {
  age: 20,
  name: "Kevin Bacon"
};
```

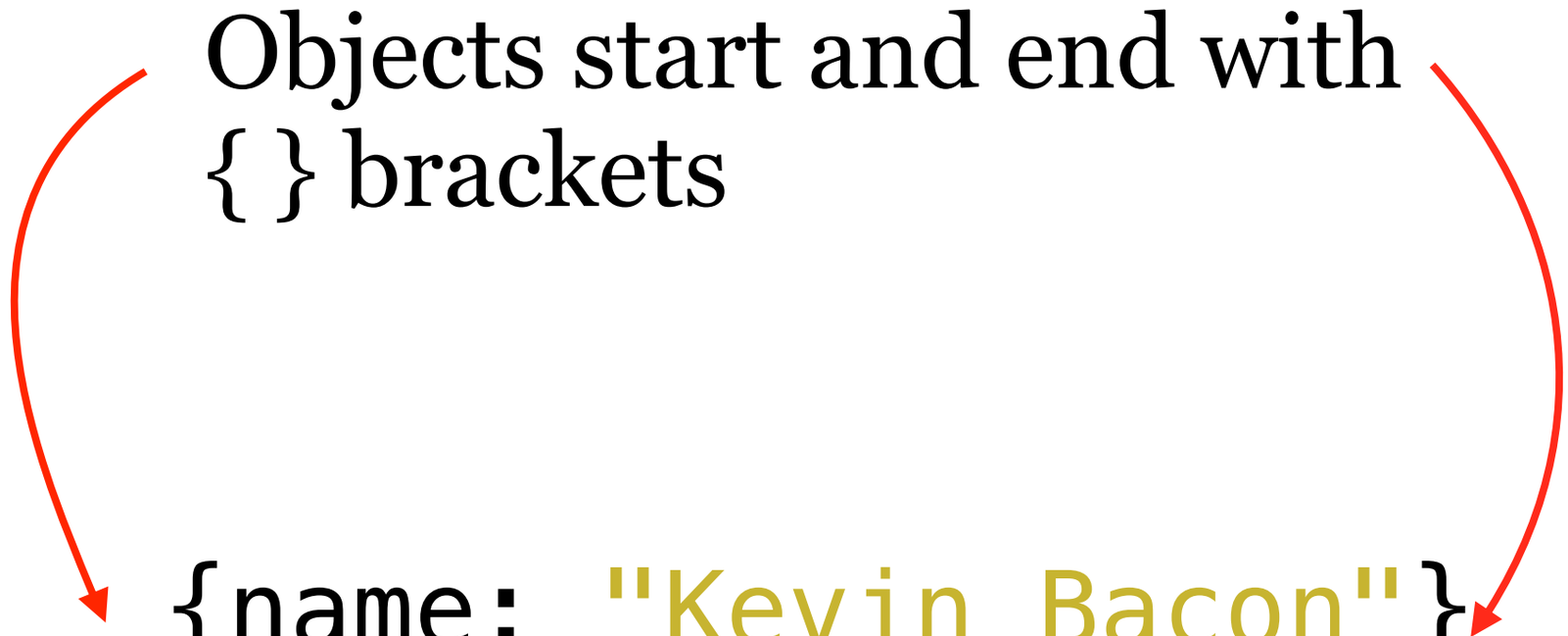
SYNTAX OF THE "KEY: VALUE" PAIRS

```
{key: "Value"}
```

SYNTAX OF THE “KEY: VALUE” PAIRS

```
{
  age: 20,
  name: “Kevin Bacon”
  Profession: “Actor”
};
```

SYNTAX OF THE "KEY: VALUE" PAIRS



SYNTAX OF THE "KEY: VALUE" PAIRS

The key is similar to
a variable name.

```
{name: "Beyonce Knowles"}
```

SYNTAX OF THE "KEY: VALUE" PAIRS

```
{name: "Beyonce Knowles"}
```

The value can be anything!
String, number, boolean..
even function or another
object!

SYNTAX OF THE "KEY: VALUE" PAIRS

```
{name: "Beyonce Knowles", age:
20};
```

Multiple key/value pairs
are separated by commas,
like array values.

INDEX METHOD – ACCESSING DATA

Creating

```
var test = {a: "hi"};
```

Accessing

```
test["a"];
// returns "hi"
```

Assigning

```
test["a"] = "bye";
// test["a"] now
   stores “bye”
```

DOT METHOD – ACCESSING OBJECTS

Creating

```
var test = {a: "hi"};
```

Accessing

```
test.a;
// returns “hi”
```

Assigning

```
test.a = "bye";
// test[“a”] now
   stores “bye”
```

CAR EXAMPLE

```
var car = {
  make: "Ford",
  model: "Focus",
  year: 2013,
  mileage: 89000
}
```

CODE ALONG

Code_Along_Objects

JSON

- JavaScript Object Notation
- Way of representing objects we create as “Strings”
- Safe way of saving objects, and passing them between places

- Things like Facebook and Twitter use JSON to pass data around.
  - When you tweet, JSON is sent to the twitter servers

SYNTAX - JSON

Normal Object `{name: "John Smith"}`

JSON `'{"name":"John Smith"}'`

It's a string!

JSON - TURING OBJECTS INTO JSON

```
JSON.stringify(obj);
// turns objects into JSON strings

JSON.parse(string);
// turns JSON strings into objects
```

LAB TIME

Exercise JSON

FUNCTIONS

FUNCTIONS

- Functions are ways to save little bits of behaviour.

- If you find you're doing something similar with lots of different bits of data, write a function.

- DRY - Don't repeat yourself. This is really important.

- For instance, a function for deposit and withdrawal in the ATM code.

FUNCTIONS - SYNTAX

```
function Keyword.
Like the “var” keyword.


function functionName(arg1, arg2) {
  //Body of function
}
```

The name of your function

FUNCTIONS - SYNTAX

Arguments let you pass data into the function

```
function functionName(arg1, arg2) {
    //Body of function
}
```

The functions executed code goes between the { } brackets. Much like an “if” statement.

FUNCTIONS - EXAMPLE

```
function helloWorld() {
  console.log("Hello Functions");
}

helloWorld(); //Prints “Hello Functions to the
console.
```


The brackets execute the function. Try calling the function without them to see what happens.

FUNCTIONS - EXAMPLE

```
function addAndPrint(num1, num2) {
  var sum = num1 + num2;
  console.log(sum);
}

addAndPrint(1, 2); // Result is 3

addAndPrint(8, 2); // Result is 10
```

CODE ALONG

- Code_Along_Functions

RETURNING DATA FROM FUNCTIONS

- What if we want to use the data the function creates?
- The “return” method allows us to do that.

FUNCTIONS - EXAMPLE

```
function add(num1, num2) {
   var sum = num1 + num2;
   return sum;
}

add(1, 2); // 3 is the result

var answer = add(20, 10);
```

ORGANIZING FUNCTIONS

- Keep all your functions together in code.

- You can create a ‘functions.js’ file if you need (or like).

- When using $(document).ready( ) it’s very useful to ‘namespace’ our functions.

ORGANIZING FUNCTIONS

```
ATM = {
    createAccount: function(name, balance) {
      account = {name: name, balance: balance, createdAt: Date.now()}
      return account;
    },
    withdraw: function(account, amount) {
        account.balance = account.balance - amount;
        return account;
      },
    deposit: function(account, amount) {
        account.balance = account.balance + amount;
        return account;
      }
    }

felixAccount = ATM.createAccount('Felix','1000');
felixAccount = ATM.withdraw(felixAccount,100);
felixAccount = ATM.deposit(felixAccount,500);
console.log(felixAccount.balance);
```

LAB TIME

- Exercise_Return_Functions
- ATM_Functions

GENERAL ASSEMBLY