# CUNNING JAVASCRIPTTIPS

by Tatiana - August 2014 Cohort



separate Window / mobile / play with CSS  $\,$ 



equivalent of p in ruby useful for debugging

console.dir

console.table

console.time

console.memory

console.trace

#### console.dir

Displays all the properties of an object.

console.table

Formats arrays and objects as tables, and allows sorting on columns.

console.time

Starts and stops a timer, logs time (in ms) of intermediate code.

console.memory

Returns heap information for this process.

console.trace

Returns a stack trace for the function where it is called.

#### Question: What is logged?

```
    var doSomething = function(){
    console.log(arguments);
    }
    doSomething("Sunny", 23);
```

```
Question: What is logged?
            var doSomething = function(){
                console.log(arguments);
        4.
            doSomething("Sunny", 23);
        5.
Answer
       1. > ["Sunny", 23]
```

The type of arguments is either an array or an "array-like object"

### **JSHINT**

http://www.jshint.com/

```
demo code
module.exports = function (grunt) {
    "use strict";

var sourceFiles = grunt.file.readJSON("sourceFiles.json");
var testSpecs = grunt.file.readJSON("testSpecs.json");
var glslSourceFiles = [ "src/video/*.glsl" ];

// Project configuration.
grunt.initConfig({
    pkg : grunt.file.readJSON("package.json"),
    path : {
        main : "build/<%= pkg.name %>-<%= pkg.version %>.js",
        min : "build/<%= pkg.name %>-<%= pkg.version %>-min.js
    },
```



Sometimes, it's nice to rip off code. <a href="http://mikekus.com/portfolio">http://mikekus.com/portfolio</a>

```
function getMeBeers(count){
       if(count){
         return count;
       else{
         return 1;
```

```
function getMeBeers(count){
       if(count){
         return count;
        }
       else{
         return 1;
function getMeBeers(count){
  return count || 1;
```

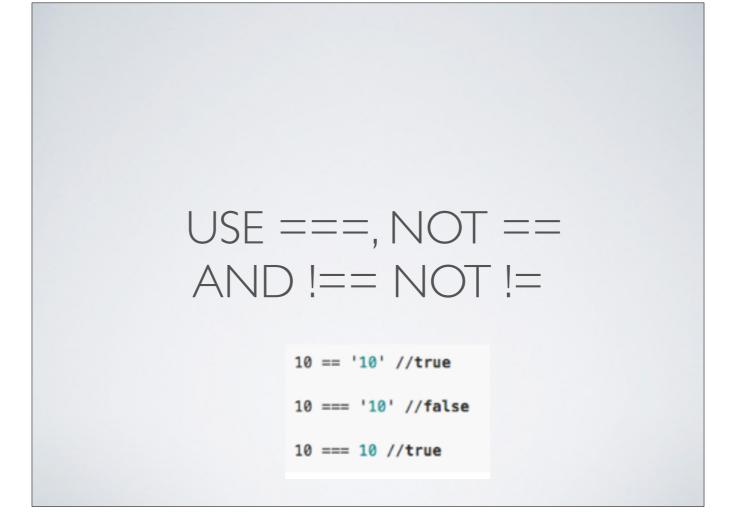
```
function getMeBeers(age,count){
    if(age>=18){
        return count || 1;
    }
}
```



```
function getMeBeers(age,count){
    if(age>=18){
        return count || 1;
    }
}
```

#### &&

```
function getMeBeers(age,count){
    return (age>=18) && (count || 1);
}
```



=== and !== will consider both value and type equality/non-equality always use === and !==.

while comparing and won't do any automatic type conversion. So, to reliably compare two values for

# FALSE EQUALS 0 TRUE EQUALS 1

# ANYTIPS YOU'D LIKE TO SHARE?