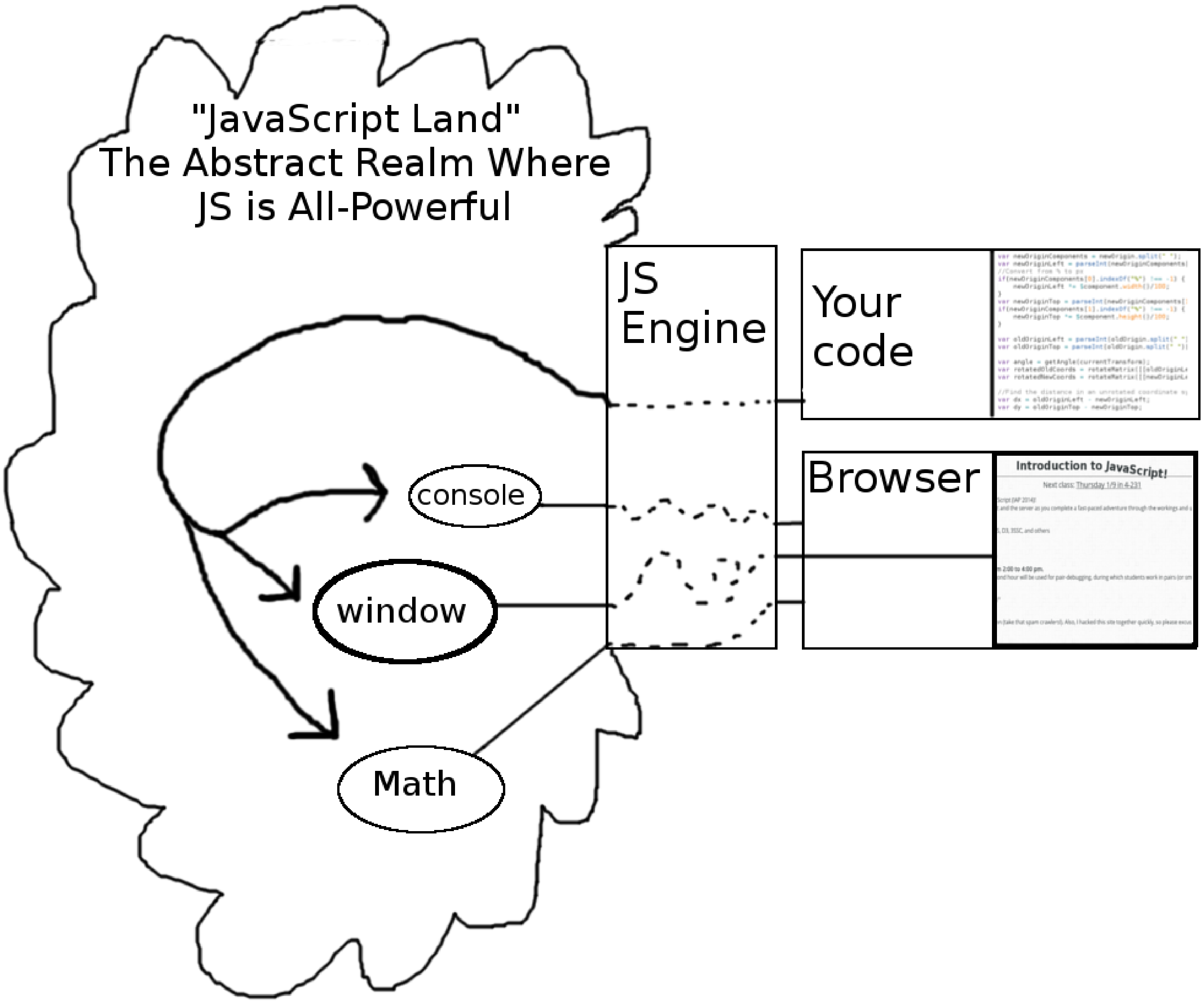
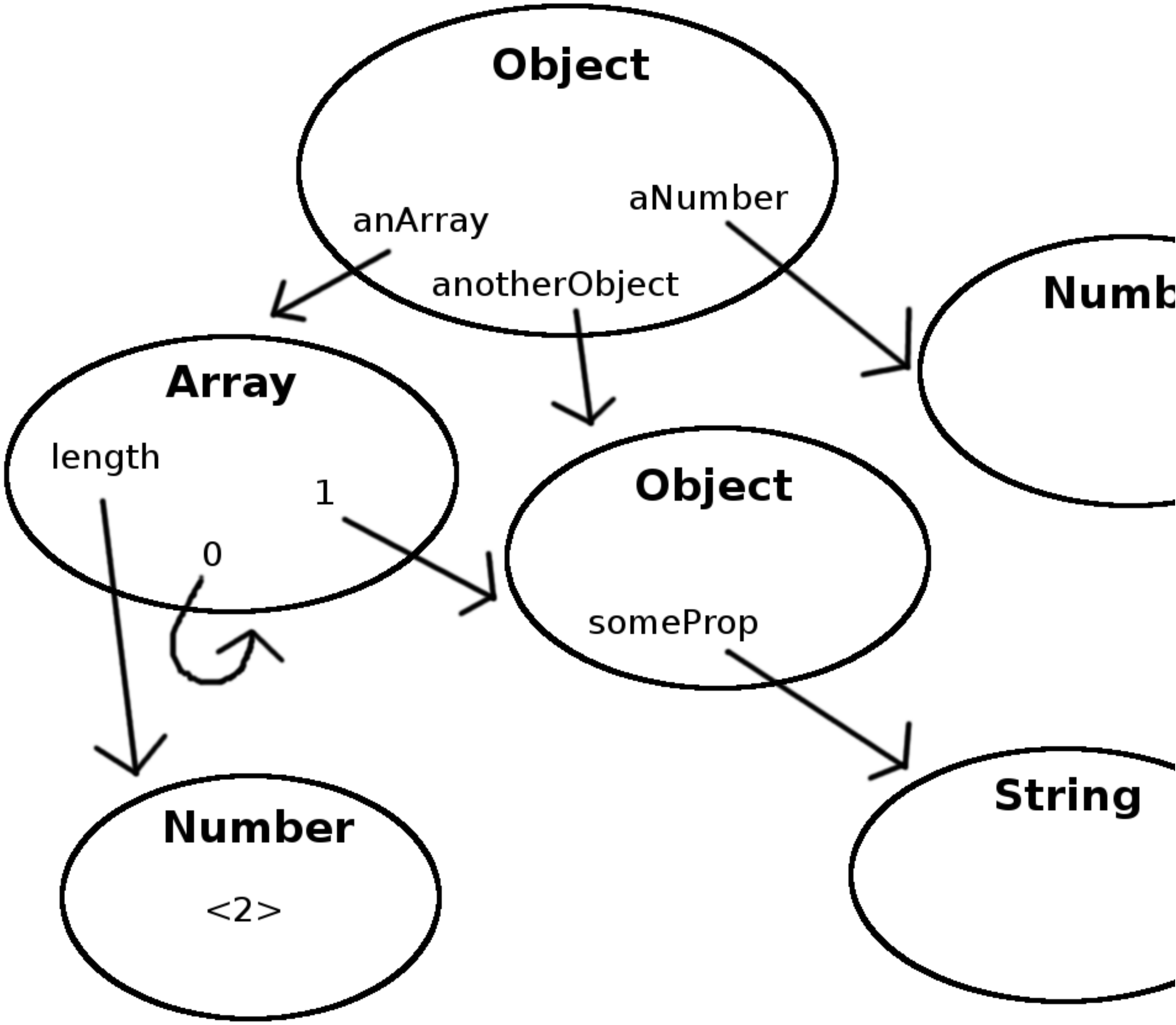
Introduction to JavaScript!

A visual metaphor for JS



```
var newOriginComponents = newOrigin.split(" ");
var newOriginLeft = parseInt(newOriginComponents[
//Convert from % to px
if(newOriginComponents[0].indexOf("%") !== -1) {
    newOriginLeft *= $component.width()/100;
}
var newOriginTop = parseInt(newOriginComponents[
if(newOriginComponents[1].indexOf("%") !== -1) {
    newOriginTop *= $component.height()/100;
}

var oldOriginLeft = parseInt(oldOrigin.split(" ")
var oldOriginTop = parseInt(oldOrigin.split(" "))

var angle = getAngle(currentTransform);
var rotatedOldCoords = rotateMatrix([[oldOriginL
var rotatedNewCoords = rotateMatrix([[newOriginL

//Find the distance in an unrotated coordinate s
var dx = oldOriginLeft - newOriginLeft;
var dy = oldOriginTop - newOriginTop;
```

Objects


Object
anArray
aNumber
anotherObject
Array
length
0
1
Object
someProp
Number
<2>
String

Numbers

```
42                  //A Number (an integer)
6.28318530717959    //Also a Number (a double)
1e-7                //A double in scientific notation

2i                  //Not a Number
notnumber           //Not a Number
```

Operations on Numbers

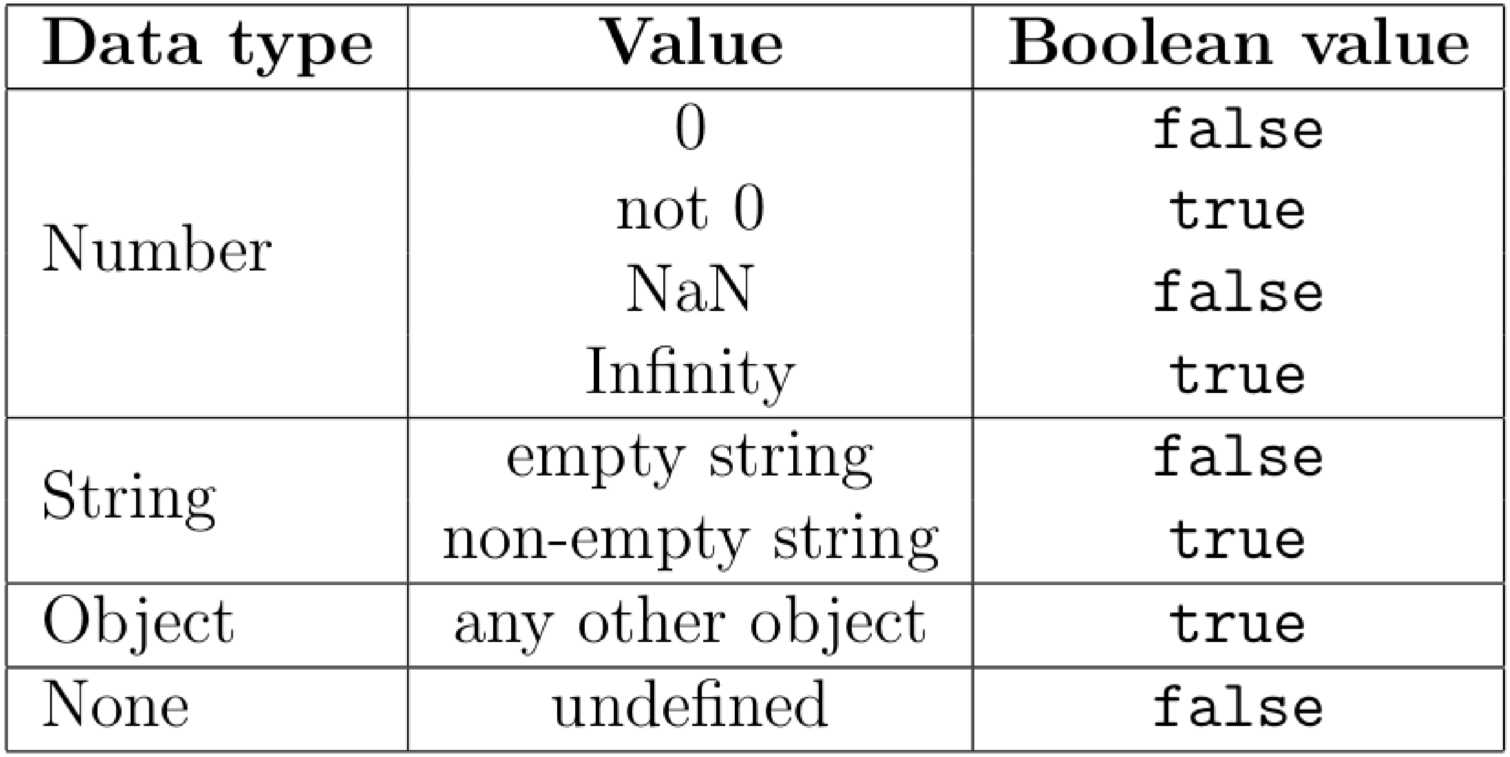
```
1 + 1 = 2

1.6666666667 - .3333333333 = 1.3333333335 //Rounding error!

21 * 2 = 42

-1024 / 32 = -32

5 % 3 = 2 //%, the modulus (remainder) operator
```

Weird Numbers

```
Infinity            //The result of dividing by zero
NaN                 //The result of something crazy like √-1
```

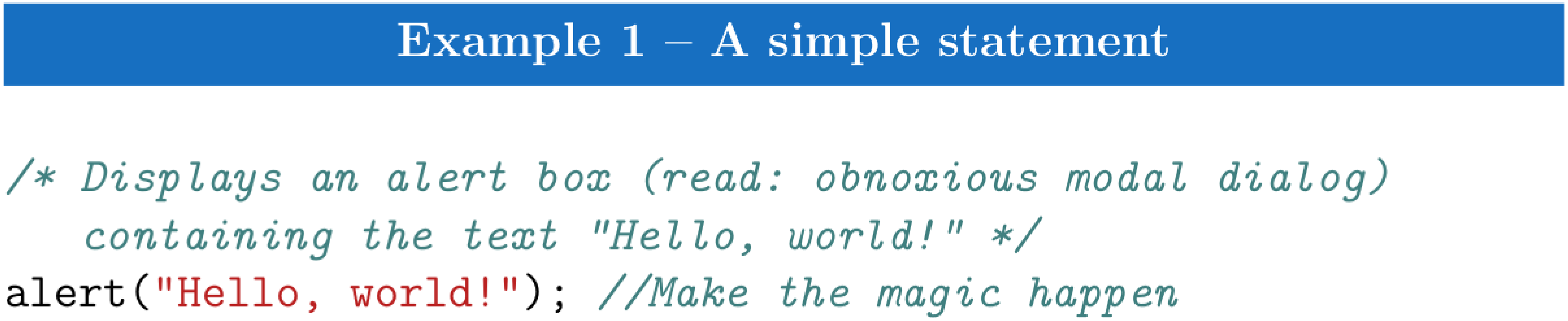
```
Infinity (-,*,/) real number = Infinity
Infinity % any number = NaN
Infinity + Infinity = Infinity
Infinity - Infinity = NaN
Infinity * Infinity = Infinity
Infinity / Infinity = NaN

NaN (-,*,/,%) any number = NaN

Any number (-,*,/,%) a number = NaN //+ is special
```

Casting (converting) to Booleans



Data type	**Value**	**Boolean value**
Number	0	`false`
	not 0	`true`
	NaN	`false`
	Infinity	`true`
String	empty string	`false`
	non-empty string	`true`
Object	any other object	`true`
None	undefined	`false`

Hello, world!



Example 1 – A simple statement

```
/* Displays an alert box (read: obnoxious modal dialog)
   containing the text "Hello, world!" */
alert("Hello, world!"); //Make the magic happen
```

Creating and assigning variables

```
var variableName = Object
```

Example 2 – Veritable variables

```
var aNumber = 42;          //camelCase
var aWord = "beautiful";
```

Scoping

Example 3 – Visible Variables

```
var var1 = "I’m a global variable!";

function assumeThisIsWhatFunctionsLookLike () {
    var local = "I’m a local variable.";
    var2 = "I’m a global variable, too!";

    console.log(local); //Outputs "I’m a local variable."
}

console.log(var1);  //"I’m a global variable!"
console.log(var2);  //"I’m a global variable, too!"
console.log(local); //undefined
```

Objects and their properties

Example 4 – The key to accessing an Object's values

```
var anObject = {
    aKey: "a value",
}

console.log(anObject.aKey); //"a value"

anObject["anotherKey"] = "another value";
console.log(anObject.anotherKey); //"another value"
```

Numbers

Example 5 – Mathematical!

```
var angle = Math.PI/4.30355158;

console.log(Math.sin(angle)); //0.6668696350365613
```

Strings

Example 6 – Letting the concat out of the bag

```
var pieStr = "I have pie";
var happyStr = 'everything is great';

console.log(pieStr + "; " + happyStr);
//"I have pie; everything is great"
```

Example 7 – JavaScript from the future

```
console.log(2000 + 14 + " is this year");
//"2014 is this year"

console.log("The year is " + 2000 + 14);
//"The year is 200014"
```

Arrays

Example 8 – Arrays make sense!

```
var boringArray = [1, 2, 3, 4];

console.log(boringArray[0]); //1 (zero-indexed)
console.log(boringArray.length); //4
```

Functions

```
function functionName(argument1, argument2, ...) {
    /*code*/
    return value; //This is optional
}
```

```
Function.prototype.property = value;
```

Example 9 – First-class functions

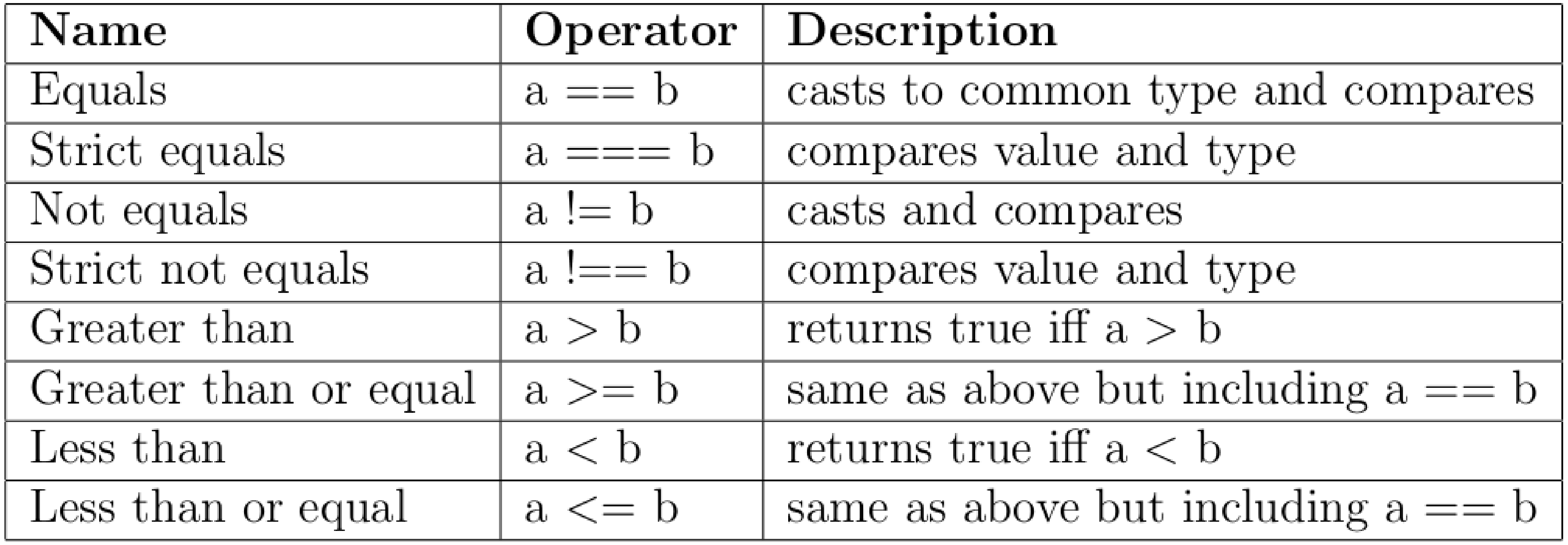
```
function stringReturner() {
    return "string";
}

function stringPrinter(stringGenerator) {
    console.log(stringGenerator());
}

stringPrinter(stringReturner); //"string"
```

Functions

Example 10 – 300: JavaScript edition

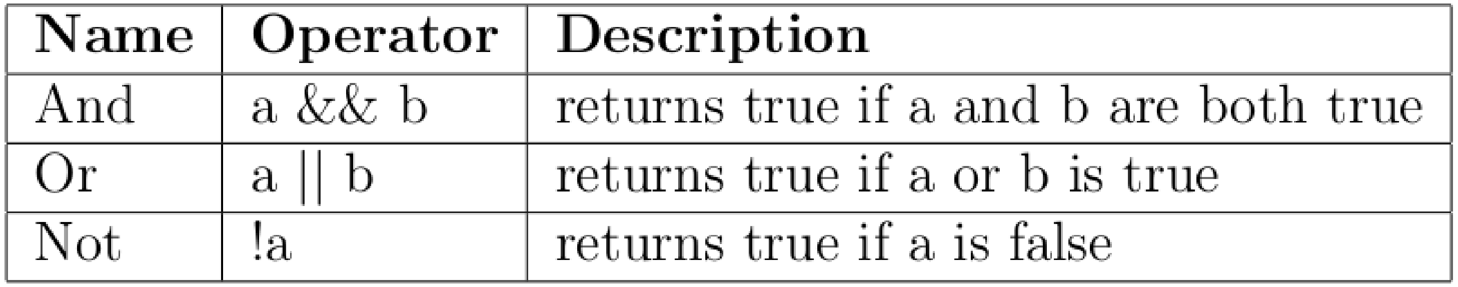
```javascript
/**
 * Represents a Movie about a topic
 *
 * @param about the topic of the movie
 */
function Movie(about) { //Function declaration
    this.about = about;
}

//Creates a new Object, {}, and calls Movie() with it as this
//then assigns the new, initialized object to boxOfficeHit
var boxOfficeHit = new Movie("JavaScript");

/**
 * Plays a movie about the topic being madness
 */
Movie.prototype.play = function() { //Function expression
    console.log("Madness? This is " + this.about + "!");
}

//Notice that the new object now has the play property
boxOfficeHit.play(); //"Madness? This is JavaScript!"
```

Conditional Operators



Name	Operator	Description
Equals	a == b	casts to common type and compares
Strict equals	a === b	compares value and type
Not equals	a != b	casts and compares
Strict not equals	a !== b	compares value and type
Greater than	a > b	returns true iff a > b
Greater than or equal	a >= b	same as above but including a == b
Less than	a < b	returns true iff a < b
Less than or equal	a <= b	same as above but including a == b

Comparison operators



Name	Operator	Description
And	a && b	returns true if a and b are both true
Or	a || b	returns true if a or b is true
Not	!a	returns true if a is false

Logical operators

Conditionals

Example 11 – Conditional number guessing

```
var randomNumber = "4"; //chosen by fair dice roll
                        //guaranteed to be random
if(a === 4) {
    var response = "The number is 4!";
} else if(a == 4) {
    var response = 'The "number" is a "4"!';
} else {
    var response = "I give up! What was the number?";
}

console.log(response); //"The "number" is a "4"!"
```

Switch statement

Example 12 – Switching it up using switch

```
var requestedPage = window.location.hash;
//the part of the URL including and after the #

switch(requestedPage) { //This can be any expression
    case "#home": //Like Python, but whitespace not required
        showHomePage();
        break;
    case "#about":
        showAboutPage();
        break; //Don't forget to include this or
               //execution will fall through to the next case
    default:
        showWittyErrorPage(); //I don't have an example.
                              //That's an error.
                              //That's all I know.
}

//This translates directly to:
if(requestedPage == "#home") {
    showHomePage();
} else if(requestedPage == "#about") {
    showAboutPage();
} else {
    showWittyErrorPage();
}
```

Ternary Operator

```
(expression) ? ifTrue : ifFalse
```

Example 13 – The *Ternary* Operator

```
var isRealisticExample = false;

console.log("This is " +
  (isRealisticExample ? "a " : "an un") +
  "realistic example."); //"This is an unrealistic example."
```

for loop

```
for(Number; Test; Iterator) { /*do stuff*/ }
```

Example 14 – Syntax for for

```
var problems = [];
problems[99] = "too lazy to fill array";

for(var i = 0; i < problems.length; i++) {
    if(problems[i] === "can't use profanity in lecture") {
        console.log("I feel you, bro");
        break; //Exits the loop
    }
}
if(i === problems.length) { //the loop finished/didn't break
    console.log("I've got 99 problems.");
    console.log("Most of them are undefined, though...")
}
```

for...in loop

Example 15 – An example to make for...in less foreign

```
var arrayObject = [];
arrayObject.push(1);
arrayObject[-1] = "wtf?";
arrayObject["-1"] = "stahp pls"; //Overwrites "wtf?"

for(var key in arrayObject) {
    console.log(key);
}
//Outputs: "0", "-1"
```

while loop

```
while(Test) { /*stuff*/ }
```

Example 16 – Whiling the time away

```
while(true) {
    yoloSwag();
}
cureCancer();
//To get to the magical land of all numbers big and small,
//go down this road for infinity and then make a left
```

do...while loop

Example 17 – A real life do...while!

```
var reasons = 0;
do {
    reasons++;
} while (false === true)

console.log("I can think of " +
             reasons + " reason to use do...while loops");
//"I can think of 1 reason to use do...while loops";
```

Wrap-up

```
var variableName = value; //Can be any object
variableName = value; //Global variable
```

```
theObject.propertyName  //key must be single word String
theObject[propertyName] //can be numerical/space-containing key
```

```
function aFunction([arguments]) { //can be prototyped
    /*code*/
    return value; //optional
}

//can not be prototyped
var anonymous = function() { alert('pass me around!'); }
```

```
var newObject = new TheFunction([arguments]);
```

```
TheFunction.prototype.newProperty = value;
//accessed using theObject.newProperty
```