Fixing Common JavaScript Bugs

Expressions & Operators

Elijah Manor
http://elijahmanor.com
@elijahmanor

pluralsight
hardcore developer training

Crude Computation Bug

Examine

```
var i;
for (i = 0; i !== 1; i += 0.1) {
  console.log("Hello: " + i);
}
```

Crude Computation Bug

Expose

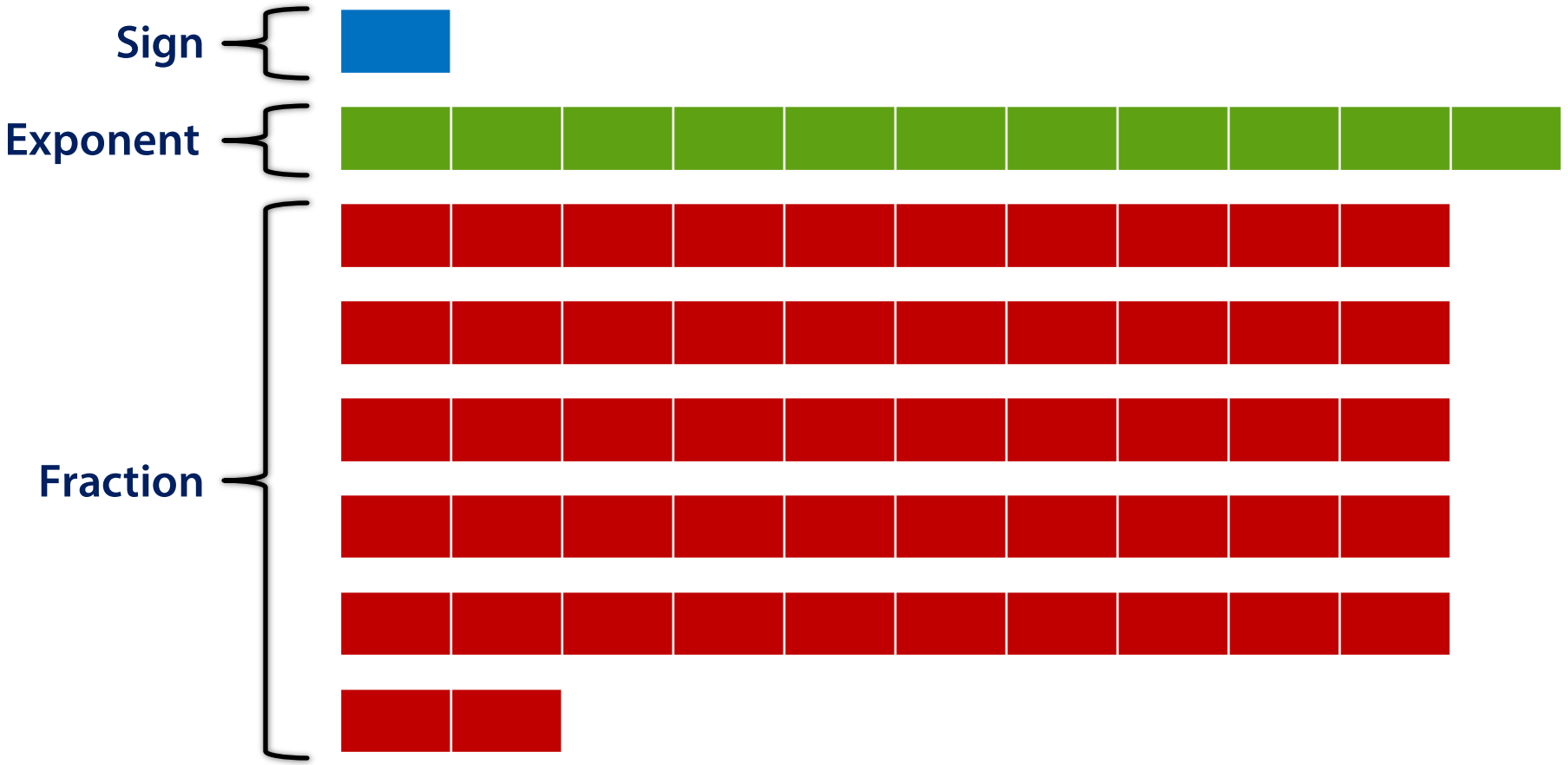
```
var i;
for (i = 0; i !== 1; i += 0.
  console.log("Hello: " + i)
}
```

```
Hello: 0
Hello: 0.1
Hello: 0.2
Hello: 0.30000000000000004
Hello: 0.4
Hello: 0.5
Hello: 0.6
Hello: 0.7
Hello: 0.7999999999999999
Hello: 0.8999999999999999
Hello: 0.9999999999999999
Hello: 1.0999999999999999
Hello: 1.2
Hello: 1.3
Hello: 1.4000000000000001
... infinite loop ...
```

Crude Computation Bug

Explain

Number type is IEEE 754 Double Precision floating point



Crude Computation Bug

Explain

```
console.log(0.1 + 0.2);          // 0.30000000000000004
console.log(999999999999999); // 1000000000000000

var money = [1.01, 2.52, 0.77, 1.50, 4.28], sum = 0;
money.forEach(function(i) { sum += i; });
console.log(sum);                // 10.080000000000002

(function sumMoney(money) {
  var result = 0;
  money = money.map(function(i) { return i * 100 });
  money.forEach(function(i) { result += i; });
  return result / 100;
}(money));                       // 10.08
```

Crude Computation Bug

```
var i;
for (i = 0; i < 1; i += 0.1) {
  console.log("Hello: " + i);
}
```

```
Hello: 0
Hello: 0.1
Hello: 0.2
Hello: 0.30000000000000004
Hello: 0.4
Hello: 0.5
Hello: 0.6
Hello: 0.7
Hello: 0.7999999999999999
Hello: 0.8999999999999999
Hello: 0.9999999999999999
```

Crude Computation Bug

```
var i;
for (i = 0; i < 10; i++) {
  console.log("Hello: " + i / 10);
}
```

```
Hello: 0
Hello: 0.1
Hello: 0.2
Hello: 0.3
Hello: 0.4
Hello: 0.5
Hello: 0.6
Hello: 0.7
Hello: 0.8
Hello: 0.9
```

Crude Computation Bug

Extra

Math Libraries

- Big - https://github.com/MikeMcl/big.js
- BigNumber - https://github.com/MikeMcl/bignumber.js/

```
// Native floating point type
console.log(0.3 - 0.1); // 0.19999999999999998

// BigNumber type
console.log(Big(0.3).minus(0.1).toString()); // "0.2"
```

Behaves more like you might expect

Crude Compuation Bug

Examine

DEMO

NOT REQUIRED

NOT COMPLETE

Mistaken Mold Bug

Examine

```
function getResource(url, callbacks) {
  reqwest(url, function (response) {
    if (typeof callbacks === "array") {
      callbacks.forEach(function (cb) { cb(response) });
    } else {
      callbacks(response);
    }
  });
}

function cb1(data) { console.log("callback1", data) }
function cb2(data) { console.log("callback2", data) }
getResource("data.json", cb1);
getResource("data.json", [cb1, cb2]);
```

Mistaken Mold Bug

Expose

```
function getResource(url, callbacks) {
  reqwest(url, function (response) {
    if (typeof callbacks === "array") {
      callbacks.forEach(function (cb) { cb(response) });
    } else {
      callbacks(response);
    }
  });
}
```

```
callback 1 Object { n: "1" }
Uncaught TypeError: object is not a function
```

```
function cb1(data) { console.log("callback1", data) }
function cb2(data) { console.log("callback2", data) }
getResource("data.json", cb1);
getResource("data.json", [cb1, cb2]);
```

Mistaken Mold Bug

Explain

Typeof	Value
`true`	`"boolean"`
`10`	`"number"`
`"Elijah"`	`"string"`
`function () {}`	`"function"`
`undefined`	`"undefined"`
`{}`	`"object"`
`{ name: "John" }`	`"object"`

Typeof	Value
`null`	`"object"`
`new Error()`	`"object"`
`[]`	`"object"`
`[{ name: "John" }]`	`"object"`
`new Date()`	`"object"`
`/^\w$/`	`"object"`

Mistaken Mold Bug

Explain

jQuery	Value
`$.type(null)`	`"null"`
`$.type(new Error())`	`"error"`
`$.type([])`	`"array"`
`$.type([{x:"y"}])`	`"array"`
`$.type(new Date())`	`"date"`
`$.type(/^\w$/)`	`"regexp"`

Underscore Lo-Dash	Value
`_.isNull(null)`	`true`
~~`new Error()`~~	
`_.isArray([])`	`true`
`_.isArray([{x:"y"}])`	`true`
`_.isDate(new Date())`	`True`
`_.isRegExp(/^\w$/)`	`true`
`_.isEmpty({})`	`True`
`_.isArguments(arguments)`	`True`
`_.isFinite(5)`	`true`
`_.isNaN(NaN)`	`true`

Mistaken Mold Bug


```
function getResource(url, callbacks) {
  reqwest(url, function (response) {
    if (_.isArray(callbacks)) {
      callbacks.forEach(function (cb) { cb(response) });
    } else {
      callbacks(response);
    }
  });
}
```

```
callback 1 Object { n: "1" }
callback 1 Object { n: "1" }
callback 2 Object { n: "2" }
```

```
function cb1(data) { console.log("callback1", data) }
function cb2(data) { console.log("callback2", data) }
getResource("data.json", cb1);
getResource("data.json", [cb1, cb2]);
```

Mistaken Mold Bug

Examine

DEMO

REQUIRED

NOT COMPLETE

Twisted Truth Bug

Examine

```
function sell(item, price) {
  if (price) {
    price = price === 0 ?
      "Free" : "$" + price.toFixed(2);
    console.log("Selling " + item + " for " + price);
  } else {
    console.log("Please provide a price");
  }
}

sell("New Things", 0.50);
sell("Old Things", 0);
sell("Whatchamacallit");
```

Twisted Truth Bug

Expose

```
function sell(item, price) {
  if (price) {
    price = price === 0 ?
      "Free" : "$" + price.toFixed(2);
    console.log("Selling " + item + " for " + price);
  } else {
    console.log("Please provide a price");
  }
}

sell("New Things", 0.50);
sell("Old Things", 0);
sell("Whatchamacallit");
```

```
Selling New Things for $0.50
Please provide a price
Please provide a price
```

Twisted Truth Bug

Explain

The ToBoolean Method (Truthy/Falsey Rules)

Type	Values	Equality
Undefined	undefined	False
Null	null	False
Boolean	false	False
Number	+0, -0, NaN	False
String	"" (Empty)	False
Otherwise		True

FALSEY: `false, 0, -0, null, undefined, NaN, ""`

TRUTHY: `true, 5, "John", {}, [], /^\w+$/, etc...`

Twisted Truth Bug

Exterminate

```
function sell(item, price) {
  if (price !== undefined) {
    price = price === 0 ?
      "Free" : "$" + price.toFixed(2);
    console.log("Selling " + item + " for " + price);
  } else {
    console.log("Please provide a price");
  }
}
sell("New Things", 0.50);
sell("Old Things", 0);
sell("Whatchamacallit");
```

```
Selling New Things for $0.50
Selling Old Things for Free
Please provide a price
```

Twisted Truth Bug

Examine

DEMO

REQUIRED

NOT COMPLETE

Crafty Convert Bug

Examine

```
var bacon = {
  slices: 0,
  buy: function (quantity, chocolate) {
    if (quantity == 0) { console.log("WAT?"); }
    if (chocolate == true) { console.log("Adding Joy") }
    this.slices += quantity;
    console.log(this.slices + " total slices of bacon!");
  }
};
bacon.buy(0);
bacon.buy(5);
bacon.buy(10, true);
bacon.buy("", "1");
bacon.buy("!", { toString: function() { return "1" } });
```

Crafty Convert Bug

Expose

```
var bacon = {
  slices: 0,
  buy: function (quantity, chocolate) {
    if (quantity == 0) { console.log("WAT?"); }
    if (chocolate == true) { console.log("Adding Joy") }
    this.slices += quantity;
    console.log(this.slices +
  }
};
bacon.buy(0);
bacon.buy(5);
bacon.buy(10, true);
bacon.buy("", "1");
bacon.buy("!", { toString: fu
```

```
WAT?
0 total slices of bacon!
5 total slices of bacon!
Adding Joy
15 total slices of bacon!
WAT?
Adding Joy
15 total slices of bacon!
Adding Joy
15! Total slices of bacon!
```

Crafty Convert Bug

Explain

The Strict Equality Comparison Algorithm (===)

Type	Values	Equality
Different Types		False
Undefined	Undefined	True
Null	Null	True
Number	Same values (except NaN)	True
String	Same characters	True
Boolean	Both true or both false	True
Object	Both refer to same object	True
Otherwise		False

Note: +0 and -0 are technically different values, but are equal to each other

Crafty Convert Bug

Explain

The Abstract Equality Comparison Algorithm (==)

Type X	Type Y	Equality
Same Types		Strict Equality Comparison Algorithm
Null or Undefined	Null or Undefined	True
Number	String	X == ToNumber(Y)
String	Number	ToNumber(X) == Y
Boolean		ToNumber(X) == Y
	Boolean	X == ToNumber(Y)
String or Number	Object	X == ToPrimitive(Y)
Object	String or Number	ToPrimitive(X) == Y
Otherwise		False

Crafty Convert Bug

Explain

ToNumber Method

Type	Value	Result
Undefined		NaN
Null		+0
Boolean	True	1
Boolean	False	+0
Number		No Conversion
String	""	0
String	"3.2"	3.2
String	"a3.2" or "3.2a"	NaN
Object		ToPrimitive(input); ToNumber(primValue);

Note: ToPrimitive(input) – return valueOf if returns primitive, or toString if returns primitive, otherwise throw and error

Crafty Convert Bug

Explain

The Addition Operator (+)

- Both sides will be toPrimitive(). If either is a String then both sides will be toString'ed and concatenated, otherwise both will be toNumber'ed and added together

```
console.log(4 + 2);           // 6
console.log("4" + 2);         // "42"
console.log("WAT" + 42);      // "WAT42"
console.log("WAT" + "42");    // "WAT42"
console.log("" + 42);         // "42"
```

Crafty Convert Bug

Errors:

- Line 9: `if (quantity == 0) { console.log("WAT?"); }`

  Expected '===' and instead saw '=='.

- Line 10: `if (chocolate == true) { console.log("Adding Joy"); }`

  Expected '===' and instead saw '=='.

```
var
  bu
                                                          ) }
    console.log(this.slices +  total slices of bacon!");
  }
};
bacon.buy(0);
bacon.buy(5);
bacon.buy(10, true);
bacon.buy("", "1");
bacon.buy("!", { toString: function () { return "1" } });
```

```
WAT?
0 total slices of bacon!
5 total slices of bacon!
Adding Joy
15 total slices of bacon!
```

Crafty Convert Bug

Extra

```
var bacon = {
  slices: 0,
  buy: function (quantity, chocolate) {
    if (typeof quantity == "number") {
      if (quantity == 0) { console.log("WAT?") }
      if (typeof chocolate == "boolean" && chocolate) {
        console.log("Adding Joy");
      }
      this.slices += quantity;
      console.log(this.slices +
        " total slices of bacon!");
    }
  }
};
```

Crafty Convert Bug

Examine

DEMO

REQUIRED

NOT COMPLETE

Problematic Pause Bug

Examine

```
function max() {
  var max = 0, i, len, arg;
  for (
    i = 0, len = arguments.length;
    i < len, arg = arguments[i];
    console.log("i: " + i), i++
  ) {
    max = arg > max ?
      (console.log("new max: " + arg), arg) :
      max;
  }
  return max, arg;
}
console.log(max(1, 3, 5, 6, 8, 4, 3, 5, 7, 8, 9, 2, 5));
```

Problematic Pause Bug

Expose

```
function max() {
  var max = 0, i, len, arg;
  for (
    i = 0, len = arguments.length;
    i < len, arg = arguments[i];
    console.log("i: " + i), i++
  ) {
    max = arg > max ?
      (console.log("new max: " + arg), arg) :
      max;
  }
  return max, arg;
}
console.log("max: " + max(1, 3, 5, 6, 8, 4, 9, 2));
```

```
new max: 1
arg: 1
i: 0
new max: 3
arg: 3
i: 1
...
max: undefined
```

Problematic Pause Bug

Explain

Comma Operator

"The comma operator evaluates both of its operands(from left to right) and returns the value of the second operand." --MDN

```
var single, double, wat;

wat = (single = 3, double = 3 * 2);

console.log("single", single); //3
console.log("double", double); //6
console.log("wat", wat);       //6
```

Problematic Pause Bug


```
function max() {
  var max = 0, i, len, arg;
  for (
    i = 0, len = arguments.length;
    i < len, arg = arguments[i];
    i++
  ) {
    max = arg > max ? arg : max;
  }
  return max;
}

console.log("max: " + max(1, 3, 5, 6, 8, 4, 9, 2));
```

```
max: 9
```

Problematic Pause Bug

Examine

DEMO

NOT REQUIRED

NOT COMPLETE

Ignored Invention Bug

Examine

```
function Cat(name, breed, color) {
  this.name = name || "Unknown";
  this.breed = breed || "Unknown";
  this.color = color || "Unknown";
}

var fluffy = new Cat("Fluffy", "Ragamuffin", "White"),
  midnight = Cat("Midnight", "Bombay", "Black");

console.log(JSON.stringify(fluffy));
console.log(JSON.stringify(midnight));
```

Ignored Invention Bug

Expose

```
function Cat(name, breed, color) {
  this.name = name || "Unknown";
```

```
{"name":"Fluffy","breed":"Ragamuffin","color":"White"}
undefined
Midnight Unknown Black
```

```
var fluffy = new Cat("Fluffy", "Ragamuffin", "White"),
  midnight = Cat("Midnight", "Bombay", "Black");

console.log(JSON.stringify(fluffy));
console.log(JSON.stringify(midnight));
console.log(window.name, window.breed, window.color);
```

Ignored Invention Bug

Explain

The new Operator

- Creates a new object
- Sets the object's prototype to the constructor function's prototype
- Executes the constructor function passing the new object as its this context
- [illegible] object if constructor doesn't return an object

Refers to global object

```
function Person(name) {
  this.name =
}

var person1 = [illegible]

var person2 = Person();
```

Creates new object

Object not created

Considered best practice to upper-case your constructor function to signify to developer that it needs to be new'ed up

Ignored Invention Bug

Explain

Make Your Constructor Function Smarter

- If didn't use new, then call it manually

```
function Person(name) {
    if (!(this instanceof Person)) {
        return new Person(name);
    }
    this.name = name;
}

var person1 = new Person();

var person2 = Person();
```

This technique is meant to protect against accidental creation of objects without using new

Ignored Invention Bug

Exterminate

Errors:

- Line 12: `midnight = Cat("Midnight", "Bombay", "Black");`

  Missing 'new' prefix when invoking a constructor.

```
funct
  if
    r
  }
```

```
{"name":"Fluffy","breed":"Ragamuffin","color":"White"}
{"name":"Midnight","breed":"Bombay","color":"Black"}
result undefined undefined
```

```
var fluffy = new Cat("Fluffy", "Ragamuffin", "White"),
  midnight = new Cat("Midnight", "Bombay", "Black");

console.log(JSON.stringify(fluffy));
console.log(JSON.stringify(midnight));
console.log(window.name, window.breed, window.color);
```

Ignored Invention Bug

Examine

DEMO

REQUIRED

NOT COMPLETE

Inaccurate Increase Bug

Examine

```
// Desired Output: ["apple-1", "orange-2", "banana-3"]

var fruit = ["apple-1"], index = 0, count = 1;

fruit[index++] = "orange-" + ++count;
fruit[index++] = "banana-" + ++count;

console.log(JSON.stringify(fruit));
```

Inaccurate Increase Bug

Expose

```
// Desired Output: ["apple-1", "orange-2", "banana-3"]

var fruit = ["apple-1"], index = 0, count = 1;

fruit[index++] = "orange-" + ++count;
fruit[index++] = "banana-" + ++count;

console.log(JSON.stringify(fruit));

//  Actual Output: ["orange-2", "banana-3"]
```

Where did apple-1 go?

Inaccurate Increase Bug

Explain

Arithmetic Operator (++)

Operator	Syntax	Result
Prefix	++myVar	Returns operand after adding one
Postfix	myVar++	Return operand before adding one

```
var x = 0;
x++;
console.log(x); // 1

var myArray = [], y = 0;
myArray[y] = "test" + ++y;
console.log(myArray, y); // ["test1"] 1
myArray[y] = "test" + y++;
console.log(myArray, y); // ["test1", "test1"] 2
```

Not so confusing on one line

Somewhat confusing when part of a complex statement.

Inaccurate Increase Bug

Exterminate

Errors:

- Line 9: `fruit[index++] = "orange-" + ++count;`
  Confusing pluses.
- Line 10: `fruit[index++] = "banana-" + ++count;`
  Confusing pluses.

```
// Desired                                                    na-3"]

var fruit

index += 1
count += 1
fruit[index] = "orange-" + count;

index += 1;
count += 1;
fruit[index] = "banana-" + count;

console.log(JSON.stringify(fruit));
```

Same as ++myVar, but less prone to error

Inaccurate Increase Bug

Extra

```
// Desired Output: ["apple-1", "orange-2", "banana-3"]

var fruit = ["apple-1"];

fruit.push("orange-" + (fruit.length + 1));

fruit[fruit.length] = "banana-" + (fruit.length + 1);

console.log(JSON.stringify(fruit));
```

Alternate ways to append item to array

Inaccurate Increase Bug

Examine

DEMO

NOT REQUIRED

NOT COMPLETE

Summary

- Be careful with floating point arithmetic
- Typeof operator doesn't always get what you want
- You should learn the truthy/falsey rules
- The double equals can be confusing
- The comma operator can be helpful, but be careful
- Remember to new up your constructors or make them smarter
- Be mindful of the difference between the prefix and postfix increment operator