Fixing Common JavaScript Bugs

Objects

Elijah Manor
http://elijahmanor.com
@elijahmanor

pluralsight
hardcore developer training

Pregnable Property Bug

Examine

```
<!DOCTYPE html>
<html>
<head><script src="./mootools.js"></script></head>
<body>
  <script>
    var customers = ["John Smith", "Susan Smith"], key;

    customers[10] = "Jane Smith";
    for (key in customers) {
      console.log(key, customers[key]);
    }
  </script>
</body>
</html>
```

Pregnable Property Bug

Expose

```
<!DOCTYPE html>
<html>
<head><script                                  /head>
<body>
  <script>
    var custom                                 ith"], key;

    customers[
    for (key i
      console.
    }
  </script>
</body>
</html>
```

```
0 John Smith
1 Susan Smith
10 Jane Smith
$family function()
@constructor [ undefined ]
each function()
clone function()
clean function()
invoke function()
associate function()
link function()
contains function()
...
```

Explain

Pregnable Property Bug

- `for-in` Iterates over [[Enumerable]] properties, including those inherited from the object's prototype chain
- MooTools & Prototype.js add custom prototype methods to Array, String, and others

```
var myArray = ["test1"], name;
myArray[3] = "test2";

Array.prototype.wat = "WAT!?!";
for (name in myArray) { console.log(name, myArray[name]) }
```

```
0 test1
3 test2
wat WAT!?!
```

```
for (var i = 0, len = myArray.length; i < len; i++) {
  console.log(i, myArray[i]);
}
```

```
0 test1
1 undefined
2 undefined
3 test2
```

Explain

Pregnable Property Bug

- `for-in` works with Arrays and other types, however, it is most valuable when iterating over an object, but be mindful of the prototype

```
function Person(name) { this.name = name; };
Person.prototype.married = false;

var john = new Person("John Smith"), k;
john.married = true;
for (k in john) { console.log(k, john[k]); }
```

```
name John Smith
married true
```

```
var susan = new Person("Susan Smith");
for (k in susan) {
  if (susan.hasOwnProperty(k)) {
    console.log(k, susan[k]);
  }
}
```

```
name Susan Smith
```

Pregnable Property Bug

Exterminate

```
var customers = ["John Smith", "Susan Smith"], key;
customers[3] = "Jane Smith";
for (key in customers) {
  if (customers.hasOwnProperty(key)) {
```

```
0 John Smith
1 Susan Smith
```

Errors:

- Line 4: `for (item in customers) {`

  The body of a for in should be wrapped in an if statement to filter unwanted properties from the prototype.

```
var i, len;
for (i = 0,
     len = customers.length; i < len;
     i++) {
  console.log(i, customers[i]);
}
```

```
0 John Smith
1 Susan Smith
2 undefined
3 Jane Smith
```

Pregnable Property Bug

Examine

DEMO

REQUIRED - 1

NOT COMPLETE

Accidental Ancestry Bug

Examine

```
function Animal(name) { this.name = name; }
Animal.prototype.eat = function () {
    console.log(this.name + " is eating");
};
Animal.prototype.sleep = function () {
    console.log(this.name + " is sleeping");
};

function Cat(name) { this.name = name; }
Cat.prototype = Animal.prototype;
Cat.prototype.eat = function () {
    console.log(this.name + " is eating");
    this.sleep();
};
```

Accidental Ancestry Bug

Expose

```
function
Animal
    co
};
Animal
    co
};

functi
Cat.pr
Cat.pr
    co
    thi
};
```

```
function Dog(name) { this.name = name; }
Dog.prototype = Animal.prototype;
Dog.prototype.sleep = function () {
    console.log("Attack the humans!");
};

var cat = new Cat("Fluffy");
cat.eat();

cat instanceof Animal;        // true
cat instanceof Cat;           // true
cat.constructor === Animal;   // true
cat.constructor === Cat;      // false
```

```
Fluffy is eating
Attack the humans!
```

Accidental Ancestry Bug

Explain

Break the connection

```
SubType.prototype = Object.create(SuperType.prototype);
```

Set the constructor

```
SubType.prototype.contructor = SubType;

var subType = new SubType();
console.log(subType instanceof SuperType);      // true
console.log(subType instanceof SubType);        // true
console.log(subType.constructor === SuperType); // false
console.log(subType.constructor === SubType);   // true
```

Accidental Ancestry Bug

Exterminate

```
function Animal(name) { this.name = name; }
Animal.prototype.eat = function () {
  console.log(this.name + " is eating");
};
Animal.prototype.sleep = function () {
  console.log(this.name + " is sleeping");
};

function Cat(name) { this.name = name; }
Cat.prototype = Object.create(Animal.prototype);
Cat.prototype.constructor = Cat;
Cat.prototype.eat = function () {
  console.log(this.name + " is eating");
  this.sleep();
};
```

Accidental Ancestry Bug

Extra

```
function Cat(name) {
  Animal.prototype.constructor.call(this, name);
}

Cat.prototype = Object.create(Animal.prototype);
Cat.prototype.constructor = Cat;

Cat.prototype.eat = function () {
  Animal.prototype.eat.apply(this);
  this.sleep();
};
```

Accidental Ancestry Bug

Examine

DEMO

REQUIRED - 2

NOT COMPLETE

Eccentric Envelope Bug

Examine

```
var contestants = ["John Smith", "Jane Smith"];
function isWinner(person) {
  var winner = contestants.some(function (contestant) {
    return person.name === contestant && person.winner;
  });
  if (winner) { console.log(person.name + " :)"); }
  else { console.log(person.name + " :("); }
}

isWinner({ name: new String("Elijah Manor"), winner: new
Boolean(false) });

isWinner({ name: new String("John Smith"), winner: new
Boolean(true) });
```

Expose

Eccentric Envelope Bug

```
var contestants = ["John Smith", "Jane Smith"];
function isWinner(person) {
  var winner = contestants.some(function (contestant) {
    return person.name === contestant && person.winner;
  });
  if (winner) { cons                    e + " :)"); }
  else { console.log                    ("); }
}
```

```
Elijah Manor :(
John Smith :(
```

```
isWinner({ name: new String("Elijah Manor"), winner: new
Boolean(false) });

isWinner({ name: new String("John Smith"), winner: new
Boolean(true) });
```

Eccentric Envelope Bug

Explain

JavaScript has 5 Primitive Types

- boolean, number, string, `null`, & `undefined`

JavaScript also has 3 Constructor Wrappers

- `Boolean`, `Number`, & `String`

```
typeof true // boolean
typeof new Boolean(true) // object
new Boolean(true) === new Boolean(true) // false
true == new Boolean(true) // true

typeof "Hello" // string
typeof new String("Hello") // object
new String("Hello") === new String("Hello") // false
new String("Hello") == "Hello" // true
```

Eccentric Envelope Bug

Exterminate

```
var contestants = ["John Smith", "Jane Smith"];
function isWinner(person) {
  va                                                    {
                                                        ;
  })
  if
  el
}

isWi
```

Errors:

- Line 12: `isWinner({ name: new String("Elijah Manor"), winner: new Boolean(false) });`
  Do not use String as a constructor.
- Line 12: `isWinner({ name: new String("Elijah Manor"), winner: new Boolean(false) });`
  Do not use Boolean as a constructor.
- Line 14: `isWinner({ name: new String("John Smith"), winner: new Boolean(true) });`
  Do not use String as a constructor.
- Line 14: `isWinner({ name: new String("John Smith"), winner: new Boolean(true) });`
  Do not use Boolean as a constructor.

```
isWinner({ name: "John Smith", winner: true });
```

Eccentric Envelope Bug

Extra

To Boolean

```
console.log(!!"false");          // true
console.log(Boolean("false"));   // true
```

Number To String

```
console.log(String(42));         // "42"
console.log(42 + "");            // "42"
console.log(42..toString());     // "42"
```

String To Number

```
console.log(+"42");              // 42
console.log(Number("42"));       // 42
console.log(parseInt("42", 10)); // 42
```

Eccentric Envelope Bug

Examine

DEMO

REQUIRED - 3

NOT COMPLETE

Translate Time Bug

Examine

```
var dataFromServer = [{
    "Name": "John Smith",
    "Birthday": "\/Date(1330848000000-0800)\/"
  },
  /* ... more items ... */
];

var dataFromServer = dataFromServer.map(function (item) {
  return {
    name: item.Name,
    birthday: new Date(item.PublishedAt)
  };
});
```

Translate Time Bug

Expose

```
var dataFromServer = [{
    "Name": "John
    "Birthday":                                  \/"
  },
  /* ... more i
];

var dataFromSer                                  ction (item) {
  return {
    name: item.
    birthday: n
  };
});
```

```
▼ [Object, Object] i
  ▼ 0: Object
    ▶ birthday: Invalid Date
      name: "John Smith"
    ▶ __proto__: Object
  ▼ 1: Object
    ▶ birthday: Invalid Date
      name: "Jane Smith"
    ▶ __proto__: Object
    length: 2
  ▶ __proto__: Array[0]
```

Explain

Translate Time Bug

Older Versions of .NET would return serialize dates as an escaped Date initializer with a Unix Epoch value… `"/Date(1320825600000-0800)/"`

```
new Date("/Date(1320825600000-0800)/") // Invalid Date
```

Thankfully the moment.js library knows how to convert this for us ☺

```
moment("/Date(1320825600000-0800)/")
```

Translate Time Bug

Exterminate

```
var dataFromServer = [{
    "Name": "John Smith",

  },
  /*
];

var                                                    ) {
  re

    birthday: moment(item.PublishedAt).toDate()
  };
});
```

```
▼ [Object, Object]
  ▼ 0: Object
    ▶ birthday: Sun Mar 04 2012 02:00:00 GMT-0600 (Central Standard Time)
      name: "John Smith"
    ▶ __proto__: Object
  ▼ 1: Object
    ▶ birthday: Wed Nov 09 2011 02:00:00 GMT-0600 (Central Standard Time)
      name: "Jane Smith"
    ▶ __proto__: Object
    length: 2
  ▶ __proto__: Array[0]
```

Translate Time Bug

Extra

```
// Formatting dates
moment().format('dddd');       // Thursday
moment().format("MMM Do YY"); // Jul 25th 13
moment().format();             // 2013-07-25T23:33:26-05:00

// Timeago
moment("20111031", "YYYYMMDD").fromNow(); // 2 years ago
moment().startOf('day').fromNow();        // a day ago
moment().endOf('day').fromNow();          // in 28 minutes
moment().startOf('hour').fromNow();       // 32 minutes ago

// Calendar Time
moment().subtract('days', 10).calendar(); // 07/15/2013
moment().add('days', 10).calendar();      // 08/04/2013
```

Translate Time Bug

Examine

DEMO

NOT REQUIRED - 1

NOT COMPLETE

Perpetual Property Bug

Examine

```
Object.defineProperty(window, "MEANING_OF_LIFE", {
  writable: false,
  value: 42
});

console.log(window.MEANING_OF_LIFE);
window.MEANING_OF_LIFE = 24;
console.log(window.MEANING_OF_LIFE);
```

Expose

Perpetual Property Bug

```
Object.defineProperty(window, "MEANING_OF_LIFE", {
  writable: false,
  value: 42
});

console.log(window.MEANING_OF_LIFE);
window.MEANING_OF_LIFE = 24;
console.log(window.MEANING_OF_LIFE);
```

WAT!?!

```
42
42
```

Perpetual Property Bug

Explain

EcmaScript 5 does not throw an error when trying to redefine an immutable property…

```
undefined = true; // ignore
```

However, it will throw an error if you are in `"use strict"`; mode!

```
(function () {
  "use strict";
  undefined = true;
}());
```

Uncaught TypeError: Cannot assign to read only property 'undefined' of [object Object]

Perpetual Property Bug

Exterminate

```
(function () {
  "use strict";

  Object.defineProperty(window, "MEANING_OF_LIFE", {
    writable: false,
    value: 42
  });

  consol
  window
  consol
}());
```

```
42
Uncaught TypeError: Cannot assign to
read only property 'MEANING_OF_LIFE' of
[object Object]
```

Perpetual Property Bug

Examine

DEMO

NOT REQUIRED - 2

NOT COMPLETE

Examine

Strange Set Bug

```
var easyCombination = new Array(13),
  hardCombination   = new Array(42, 16, 21),
  combined = easyCombination.concat(hardCombination);

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

Expose

Strange Set Bug

```
var easyCombination = new Array(13),
  hardCombination   = new Array(42, 16, 21),
  combined = easyCombination.concat(hardCombination);

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

```
[null, null, null, null, null,
null, null, null, null, null,
null, null, null, 42, 16, 21]
```

Strange Set Bug

Explain

Array Constructor is Overloaded

```
new Array() // []
```

```
new Array(1, 2, 3) // [1, 2, 3]
new Array("1", "2", "3") // [ "1", "2", "3" ]
```

```
new Array(3) // [undefined, undefined, undefined]
```

Strange Set Bug

Exterminate

```
    // JSHint: The array literal notation [] is preferrable.
    // JSLint: Use the array literal notation []
var myArray1 = new Array(),

    // No Warnings
    myArray2 = new Array(5),

    // JSLint: Use the array literal notation []
    myArray3 = new Array(1, 2, 3),

    // JSLint: Use the array literal notation []
    myArray4 = new Array("a", "b", "c");
```

Strange Set Bug

Examine

DEMO

NOT REQUIRED - 3

NOT COMPLETE

Malformed Message Bug

Examine

```
var dataFromServer = "{ name: 'John Smith', phone: [
'555-123-4567', '123-456-7890' ], age: 28 }";

var parsed = JSON.parse(dataFromServer);

console.log(parsed);
```

Expose

Malformed Message Bug

```
var dataFromServer = "{ name: 'John Smith', phone: [
'555-123-4567', '123-456-7890' ], age: 28 }";

var parsed = JSON.parse(dataFromServer);

console.log(parsed);
```

```
Uncaught SyntaxError: Unexpected token n
```

Explain

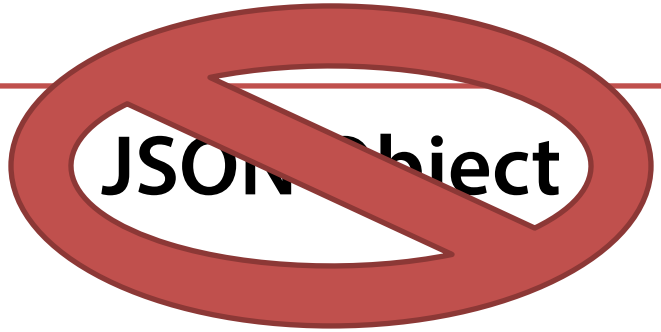
Malformed Message Bug

Differences between JSON (string) and Object Literal (object)

- JSON: Keys are double quoted
- JSON: Strings are double quoted

JSON is a subset of the object literal notation of JavaScript

```
var objl = { b: 'c', d: [1, '2', 3], e: 4 };
var json = '{ "b": "c", "d": [1, "2", 3], "e": 4 }';
```



JSON is a string, it's not an object.
It's serialized data

Exterminate

Malformed Message Bug

```
var fromServer = '{ "name": "John Smith", "phone": [
"555-123-4567", "123-456-7890" ], "age": 28 }';

var parsed = JSON.parse(fixServer);

console.l
```

```
▼ Object {name: "John Smith", phone: Array[2], age: 28}
    age: 28
    name: "John Smith"
  ▼ phone: Array[2]
      0: "555-123-4567"
      1: "123-456-7890"
      length: 2
    ▶ __proto__: Array[0]
  ▶ __proto__: Object
```

Malformed Message Bug

Examine

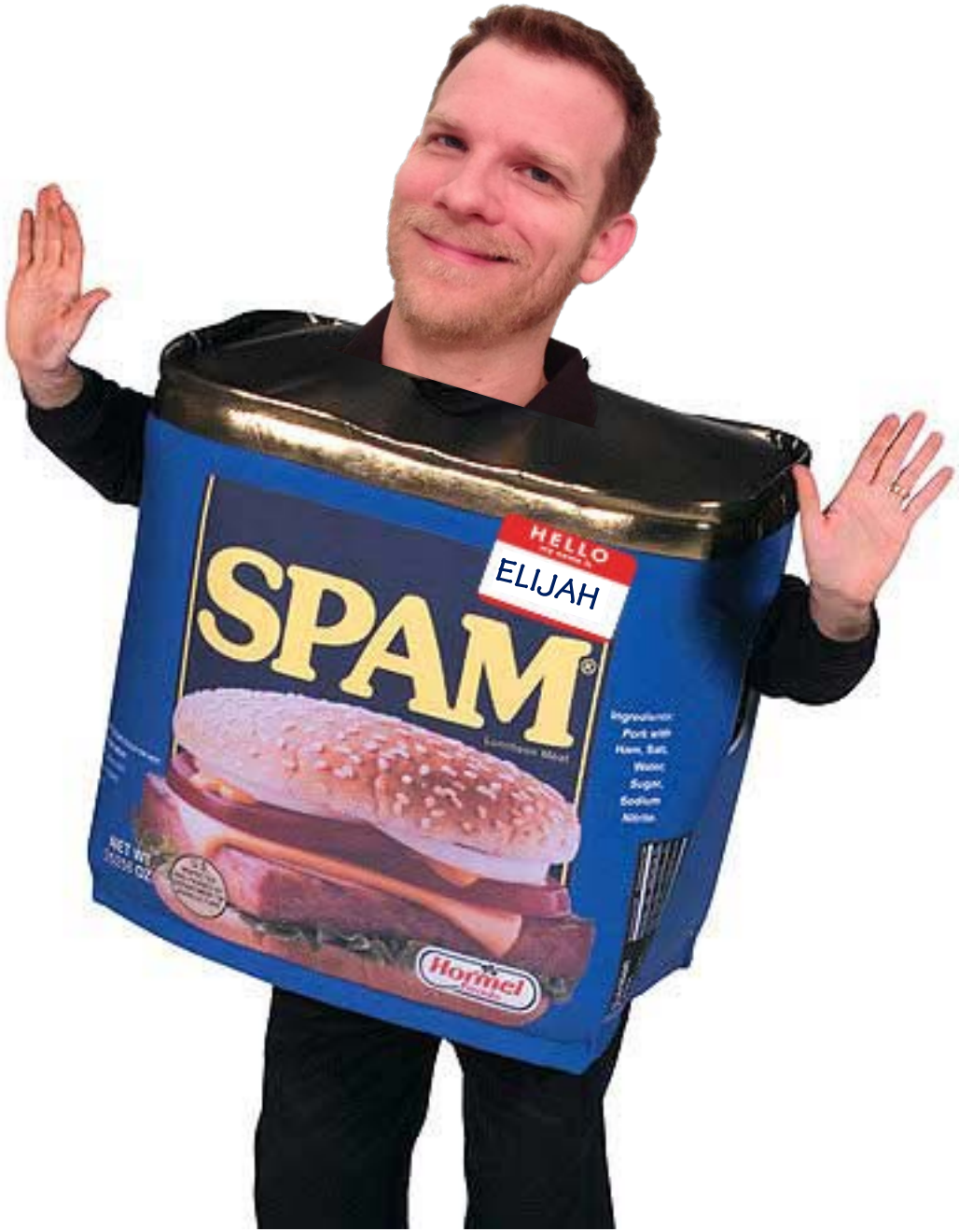
DEMO

REQUIRED - 4

NOT COMPLETE

Summary

- Use for...in on objects and check hasOwnProperty and to use a standard for loop on arrays
- Remember to set the constructor on subclasses and Object.create the prototype
- Don't use the wapper constructors (Boolean, Number, or String) unless it is for conversion
- Be careful converting dates. Consider using a helper library such as moment.js
- Use strict mode to provide exceptions when setting read-only properties
- Stay away from the array constructor and use the literal syntax instead
- Make sure you are using valid JSON


HELLO
ELIJAH
SPAM
Hormel