Fixing Common JavaScript Bugs

Functions

Elijah Manor
http://elijahmanor.com
@elijahmanor

pluralsight
hardcore developer training

Raised Resource Bug

Examine

```
var score = 1000;

function play() {
  console.log("begin: " + score);
  if (!score) {
    console.log("setting default");
    var score = 100;
  }
  console.log("end: " + score);
}

console.log(score); play(); console.log(score);
```

Rai

```
1000
begin: undefined
setting default
end: 100
1000
```

```
var score = 1000;

function play() {
  console.log("begin: " + score); // undefined
  if (!score) {
    console.log("setting default");
    var score = 100;
  }
  console.log("end: " + score);  // 100
}

console.log(score); play(); console.log(score);
```

Raised Resource Bug

Explain

Block Scope

- The following is a C# example showing block scope

```
public class Test
{
    public static void Main()
    {
        if (true)
        {
            var secret = "all the things";
            Console.WriteLine(secret);
        }
        Console.WriteLine(secret);
    }
}
```

Test.cs(11,27): error CS0103:
The name `secret` does not exist
in the current context

Raised Resource Bug

Explain

Functional Scope

- The following is a JavaScript example showing functional scope

No Error

```
var test = (function() {
    var main = function () {
        if (true) {
            var secret = "all the things";
            console.log(secret); // all the things
        }
        console.log(secret);     // all the things
    }

    return { main: main };
}());
```

Raised Resource Bug

Explain

Functional Scope

- JavaScript hoists variable declarations to top of function

```
var test = (function() {
    var main = function () {
        var secret = undefined;
        if (true) {
            secret = "all the things";
            console.log(secret); // all the things
        }
        console.log(secret);     // all the things
    }

    return { main: main };
}());
```

Explain

Rai

```
1000
begin: undefined
setting default
end: 100
1000
```

```
var score = 1000;

function play() {
  var score = undefined;
  console.log("begin: " + score); // undefined
  if (!score) {
    console.log("setting default");
    score = 100;
  }
  console.log("end: " + score);  // 100
}

console.log(score); play(); console.log(score);
```

Rais

```
1000
begin: 1000
end: 1000
1000
```

```
var score = 1000;

function play() {
  console.l
  if (!scor
    console
    score =
  }
  console.l
}

console.log(score); play(); console.log(score);
```

Errors:

- Line 10: `var score = 100;`

  'score' is already defined.

- Line 12: `console.log("end: " + score);`

  'score' used out of scope.

Raised Resource Bug

Extra

Best Practice

- Declare all of your variables at the top of the function score

```
var vendingMachine = function() {
  var dayOfWeek = new Date().getDay(), items = [];

  items.push("Coffee");
  if (dayOfWeek === 1) {
    items.push("Sushi");
  } else if (dayOfWeek === 5) {
    items.push("Pizza");
  }

  return items;
}
```

Declare all variables at top of function

Raised Resource Bug

Examine

DEMO

REQUIRED - 1

NOT COMPLETE

Early Execution Bug

Examine

```
try { sayHello()  } catch (e) { console.log(e.message) }

try { sayGoodbye() } catch (e) { console.log(e.message) }

var sayHello = function () { console.log("Hello") };

function sayGoodbye() { console.log("Goodbye") }

sayHello();

sayGoodbye();
```

Early Execution Bug

Expose

```
try { sayHello()  } catch (e) { console.log(e.message) }

try { sayGoodbye() } catch (e) { console.log(e.message) }

var sayHello = function () { console.log("Hello") };

function sayGoodbye() { console.log("Goodbye") }

sayHello();

sayGoodbye();
```

```
Undefined is not a function
Goodbye
Hello
Goodbye
```

Early Execution Bug

Explain

Functional Scope

- Function expressions hoist declaration and keeps assignment the same
- Function statements hoists both declaration and assignment

```
(function () {
  var test1 = undefined;
  var test2 = function () { /* test 2 */ };

  test1(); test2();

  test1 = function() { /* test 1 */ };
  function test2() { /* test 2 */ };
}());
```

Early Execution Bug

Explain

```
var sayHello = undefined;

var sayGoodbye = function () { console.log("Goodbye") }

try { sayHello()  } catch (e) { console.log(e.message) }

try { sayGoodbye() } catch (e) { console.log(e.message) }

sayHello = function () { console.log("Hello") };

sayHello();

sayGoodbye();
```

```
Undefined is not a function
Goodbye
Hello
Goodbye
```

Early Execution Bug

Exterminate

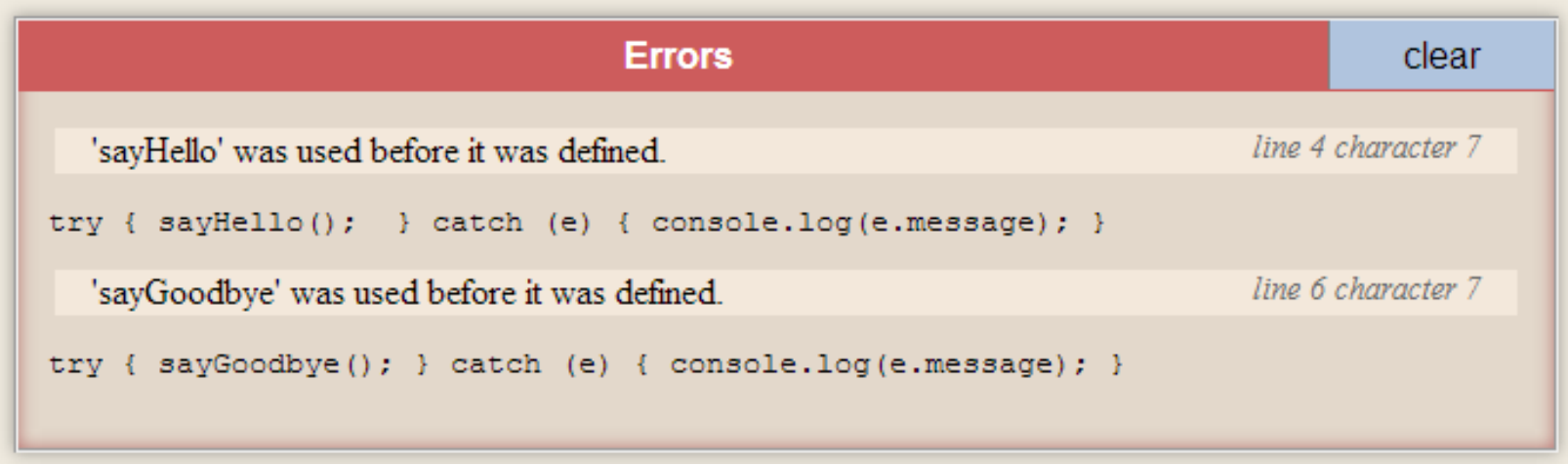
```
var sayHello = function () { console.log("Hello") };

sayHello();

sayGoodbye();

fu
```

Function expressions before usage



Early Execution Bug

Examine

DEMO

NOT REQUIRED

NOT COMPLETE

Morphed Method Bug

Examine

```
var element = document.getElementById("greeting");

function text() { return element.innerText; }

function text(value) { element.innerText = value; }

function text(callback) {
  element.innerText = callback(element.innerText);
}

text();
text("Hello");
text(function (text) { return text + " World!"; });
```

Morphed Method Bug

Expose

```
var element = document.getElementById("greeting");

function text() { return element.innerText; }

function text(value) { element.innerText = value; }

function text(callback) {
  element.innerText = callback(element.innerText);
}

text();
text("Hello");
text(function (text) { return text + " World!"; });
```

Uncaught TypeError:
undefined is not a function

Morphed Method Bug

Explain

```
function overloaded(parm1, parm2) {
  if (arguments.length === 0) {
    console.log("NOTHING");
  } else if (typeof parm1 === "string") {
    console.log("Hello " + parm1);
  }
  console.log(typeof parm2);
  console.log(typeof arguments[2]);
}

overloaded();
overloaded("World!");
overloaded("World!", 5);
overloaded("World!", 5, function () { });
```

Morphed Method Bug

Exterminate

Errors:

- Line 8: `function text(value) { element.innerText = value; }`

  'text' is already defined.

- Line 10: `function text(callback) {`

  'text' is already defined.

```
va

fu
        element.innerText = value(element.innerText);
    }
}

text();
text("Hello");
text(function (text) { return text + " World!"; });
```

Overload with parameter type checking

Extra

Morphed Method Bug

	typeof	jQuery.type()	Underscore.js
true	boolean	boolean	_.isBoolean()
10	number	number	_.isNumber()
"Elijah"	string	string	_.isString()
function() {}	function	function	_.isFunction()
undefined	undefined	undefined	_.isUndefined()
{ name:"Elijah" }	object	object	_.isObject()
null	object	null	_.isNull()
new Error()	object	error	
[{ name:"Elijah" }]	object	array	_.isArray()
new Date()	object	date	_.isDate()
/^\w+$/	object	regexp	_.isRegExp()

Morphed Method Bug

Examine

DEMO

REQUIRED - 2

NOT COMPLETE

Confounding Context Bug

Examine

```
var student = {
  name: "John Smith",
  resume: [],
  study: function (item) {
    console.log(this.name + " is studying " + item);
    function addToResume(item) { this.resume.push(item) }
    addToResume(item);
  }
}, memorize = student.study;

student.study("chemistry");
console.log(student.resume);
memorize("history");
console.log(student.resume);
```

Confounding Context Bug

Expose

```
var student = {
  name: "John",
  resume: [],
  study: function (item) {
    console.log(this.name + " is studying " + item);
    function addToResume(item) { this.resume.push(item) }
    addToResume(item);
  }
}, memorize = student.study;

student.study("chemistry");
console.log(student.resume);
memorize("history");
console.log(student.resume);
```

```
John is studying chemistry
```

Uncaught TypeError: Cannot call method 'push' of undefined

Confounding Context Bug

Explain

```
function hiWindow() { console.log(this); }
hiWindow(); // [object Window] => global

var person = {
    name: "John Smith",
    greet: function () {
        console.log(this);
    }
};
person.greet(); // [object Object] => person

var hello = person.greet;
hello(); // [object Window] => global
```

Confounding Context Bug

Explain

```
function hiWindow() { console.log("Hello " + this.name) }
hiWindow.call({ name: "John" }); // Hello John

var person = {
  name: "Jane",
  greet: function() { console.log("Hello " + this.name) }
};
var hello = person.greet.bind({ name: "Jake" });
hello(); // Hello Jake

var Person = function(name) { this.name = name; };
Person.prototype.greet =
  function() { console.log("Hello " + this.name) }
new Person("Jane").greet(); // Hello Jane
```

hiWindow.apply({ name: "John" })

Confounding Context Bug

Explain

Code	this
`this`	`window`
`myFunction()`	`window`
`myObject.method()`	`myObject`
`myFunction.call(context, arg1, arg2)`	`context`
`myFunction.apply(context, [arg1, arg2])`	`context`
`var f = myFunction.bind(context); f();`	`context`
`var myObject = new Object();`	`myObject`

Confounding Context Bug

Exterminate

```
var student = {
  name: "John", resume: [],
  study: function (item) {
    var that = this;
    console.log(this.name + " is studying " + item);
    function addToResume(item) { that.resume.push(item) }
    addToResume(item);
  }
}, memorize = student.study;

student.study("chemistry");
console.log(student.resume);
memorize.call(student, "history");
console.log(student.resume);
```

```
John is studying chemistry
["chemistry"]
John is studying history
["chemistry", "history"]
```

Confounding Context Bug

Extra

(func
  "us
  fun
  hil

Errors:

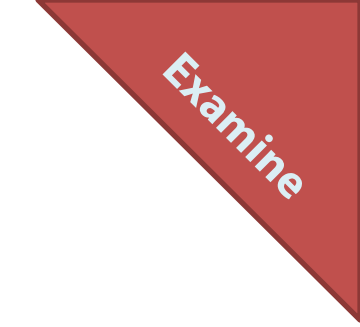
- Line 9: `function addToResume(item) { this.resume.push(item); }`

  Possible strict violation.

```
  var person = {
    name: "Jane",
    greet: function () {
      console.log(this);
    }
  };
  person.greet.call(null); // null
}());
```

"use strict"; eliminates this coercion to the global object

Confounding Context Bug



DEMO

REQUIRED - 3

NOT COMPLETE

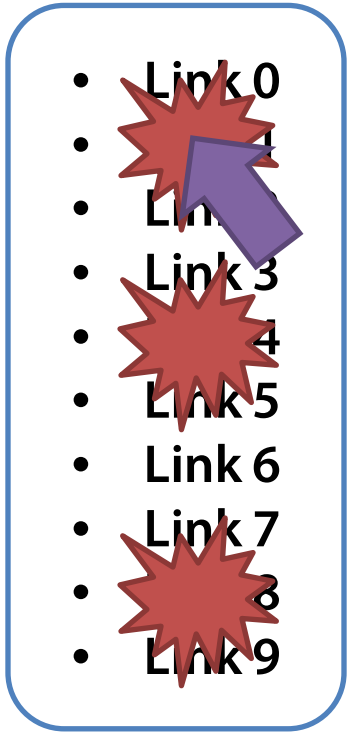
Escaped Environment Bug

Examine

```
var ul = document.querySelector("ul"), i, li;
for (i = 0; i < 10; i++) {
  li = document.createElement("li");
  li.innerHTML = "Link " + i;
  li.addEventListener("click", function () {
    console.log("You've clicked " + i);
  }, false);
  ul.appendChild(li);
}
```

Escaped Environment Bug

```
var ul = document.querySelector("ul"), i, li;
for (i = 0; i < 10; i++) {
  li = document.createElement("li");
  li.innerHTML = "Link " + i;
  li.addEventListener("click", function () {
    console.log("You've clicked " + i);
  }, false);
  ul.appendChild(li);
}
```




```
You've clicked 10
You've clicked 10
You've clicked 10
```

Escaped Environment Bug

Explain

- We need to introduce the concept of a closure

"A closure is a special kind of object that combines two things: a function, and the environment in which that function was created." --MDN

```
function makeAdder(x) {
    return function (y) { return x + y; };
}

var add5  = makeAdder(5);
var add10 = makeAdder(10);

console.log(add5(2));  // 7
console.log(add10(2)); // 12
```

Escaped Environment Bug

Exterminate

```
var ul = document.querySelector("ul"), i, li;
for (i = 0; i < 10; i++) {
  li = document.createElement("li");
  li.innerHTML = "Link " + i;
  li.addEventListener("click", (function (index) {
    return function () {
      console.log("You've clicked " + index);
    };
  }(i)), false);
  ul.appendChild(li);
}
```

JSHint: Line 9: `}(i)), false);`
Don't make functions within a loop.

```
You've clicked 1
You've clicked 4
You've clicked 8
```

Escaped Environment Bug

Extra

```
var ul = document.querySelector("ul"), i, li;
for (i = 0; i < 10; i++) {
  li = document.createElement("li");
  li.innerHTML = "Link " + i;
  li.addEventListener("click", clickHandler(i), false);
  ul.appendChild(li);
}

function clickHandler(index) {
  return function() {
    console.log("You've clicked " + index);
  };
}
```

Escaped Environment Bug

Examine

DEMO

REQUIRED - 4

NOT COMPLETE

Peculiar Parameter Bug

Examine

```
function sum() {
  var result = 0;
  arguments.forEach(function (number) {
    result += number;
  });
  return result;
}

function average() {
  return sum.apply(null, arguments) / arguments.length;
}

console.log(sum(1, 2, 3, 4, 5));
console.log(average(10, 9.5, 8, 9.5, 10));
```

Peculiar Parameter Bug

Expose

```
function sum() {
  var result = 0;
  arguments.forEach(function (number) {
    result += number;
  });
  return result;
}

function average() {
  return sum.apply(null, arguments) / arguments.length;
}

console.log(sum(1, 2, 3, 4, 5));
console.log(average(10, 9.5, 8, 9.5, 10));
```

Uncaught TypeError: Object #<Object> has no method 'forEach'

Peculiar Parameter Bug

Explain

- Arguments is not an array

"An Array-like object corresponding to the arguments passed to a function. " --MDN

```
function myFunction() {
  console.log(arguments.length);
  console.log(arguments[0]);
  console.log(arguments[1]);
  console.log(arguments[2]);
  console.log(arguments[3]);
}

myFunction(1, "Hello", true, { name: "John" });
```

```
4
1
Hello
true
Object { name="John" }
```

Peculiar Parameter Bug

Exterminate

```
function sum() {
  var result = 0, length = arguments.length, i;
  for (i = 0; i < length; i++) {
    result += arguments[i];
  }
  return result;
}

function average() {
  return sum.apply(null, arguments) / arguments.length;
}

console.log(sum(1, 2, 3, 4, 5));
console.log(average(10, 9.5, 8, 9.5, 10));
```

```
15
9.4
```

Peculiar Parameter Bug

Extra

```
function sum() {
  var result = 0, args = [].slice.call(arguments);
  args.forEach(function (number) {
    result += number;
  });
  return result;
}

function average() {
  return sum.apply(null, arguments) / arguments.length;
}

console.log(sum(1, 2, 3, 4, 5));
console.log(average(10, 9.5, 8, 9.5, 10));
```

```
15
9.4
```

Peculiar Parameter Bug

Examine

DEMO

NOT REQUIRED

NOT COMPLETE

Condemned Criterion Bug

Examine

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

  var min = 0, max = 100, random =
    Math.floor(Math.random() * (max - min + 1) + min);

  console.log("Random Number: " + random);

  setTimeout(arguments.callee, 5000);
}());
```

Condemned Criterion Bug

Expose

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

  var min = 0, max = 100, random =
    Math.floor(Math.random() * (max - min + 1) + min);

  console.log("Random Number: " + random);
  setTimeout(arguments.callee, 5000);
}());
```

```
Random Number: 3
Uncaught TypeError: 'caller', 'callee', and 'arguments'
properties may not be accessed on strict mode functions
or arguments objects for calls to them
```

Condemned Criterion Bug

Explain

- A little bit of history…

"Early versions of JavaScript did not allow named function expressions…" --MDN

```
var fibonacci =
  [0, 1, 2, 3, 4, 5, 6, 7].map(function algorithm(number) {
    return n >= 2 ?
      algorithm(number - 1) + algorithm(number – 2):
      number;
  });

console.log(fibonacci);
```

Condemned Criterion Bug

Exterminate

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

  var min = 0, max = 100, random =
    Math.f                                  min);

  console.
  setTimeo
}());
```

Errors:

- Line 9: `setTimeout(arguments.callee, 5000);`

  Avoid arguments.callee.

```
Random Number: 3
Random Number: 35
Random Number: 87
Random Number: 71
```

Condemned Criterion Bug

Examine

DEMO

NOT REQUIRED

NOT COMPLETE

Summary

- Declare your variables at the top of a function
- Make sure your function is defined before calling it
- To "overload" a function you have to check the arguments
- Be care of the value of the "this" implicit parameter
- Be aware of what a closure is and when you might need it
- Remember that the arguments parameter is array-like
- Don't use arguments.callee or arguments.caller in strict mode