RICE®

Web Development

COMP 431 / COMP 531

Scope

Scott E Pollack, PhD

February 2, 2016

Recap

- HTML
- JavaScript
- Forms
- CSS
- Events
- Homework Assignment 3 (Draft Front-End)
  - Due Tuesday 2/9

~~9/3 CSS~~
~~9/8 Events~~
~~9/10 HTML5~~
9/15 Scope

Homework Assignment 4 (JavaScript Game)
Due Thursday 2/18

Functions

```
var namedFunction = function aName() {
    // This is a comment
    return 9;
}

var unnamedFunction = function () {
    return 7;
}

function globalFunction() {
    return 33;
}
```

```
> namedFunction
< function aName()
> namedFunction.name
< "aName"
> unnamedFunction
< function unnamedFunction()
> unnamedFunction.name
< ""
> globalFunction
< function globalFunction()
> globalFunction.name
< "globalFunction"
```

Functions

```
var namedFunction = function aNam
    // This is a comment
    return 9;
}

var unnamedFunction = function ()
    return 7;
}

function globalFunction() {
    return 33;
}
```

```
> gf = globalFunction
< function globalFunction()
> gf.name
< "globalFunction"
> window.gf
< function globalFunction()
> window.namedFunction.toString()
< "function aName() {
      // This is a comment
      return 9;
  }"
```

Scope

where a name is known

```
private void go() {
    Random r = new Random();
    int sum = 0;
    int value = 0;
    int prevValue = 1;
    for (int ii = 0; ii < 100; ++ii) {
        value = r.nextInt(10);
        int product = value * prevValue;
        prevValue = value;
        sum += product;
    }
    System.out.println("The product was " + product);
    System.out.println("The sum is " + sum);
}
```

Java has block scope

Scope

where a name is known

```
ScopeExample.java:18: error: cannot find symbol
                System.out.println("The product was " + product);
                                                       ^
  symbol:   variable product
  location: class ScopeExample
1 error
```

```
        private void go() {
            int product = value * prevValue;
            prevValue = value;
            sum += product;
        }
        System.out.println("The product was " + product);
        System.out.println("The sum is " + sum);
    }
```

JavaScript has Function Scope

```
function go() {
    var sum = 0;
    var value = 0;
    var prevValue = 1;
    for (var ii = 0; ii < 100; ++ii) {
        value = Math.floor(Math.random()*10);
        var product = value * prevValue;
        prevValue = value;
        sum += product;
    }
    console.log('The product was ' + product)
    console.log('The sum is ' + sum)
}
```

```
The product was 7
The sum is 2482
```

Function Scope

Changing outer scope

Declared in global scope

```
function innerOuter() {
    var outer1 = "In Outer Scope"
    var outer2 = "Also in Outer Scope"

    var internal = function() {
        var inner = "In Inner Scope"
        outer1 = "Foo"
        var outer2 = "Redeclared"
        outer3 = "Bar"
        console.log([inner, outer1, outer2, outer3])
    }
    internal()
    console.log([outer1, outer2, outer3])
    console.log(inner)
```

```
> outer3
< "Bar"
```

```
["In Inner Scope", "Foo", "Redeclared", "Bar"]
["Foo", "Also in Outer Scope", "Bar"]
▼ Uncaught ReferenceErr
    innerOuter
    (anonymous function) @ scope.js:33
```

if it is used without declaring , then it will be Window scope
ie. global scope

Block Scope with Let

maybe good for “for loop”

strict mode!

```
function innerOuterBlock() {
    var outer1 = "In Outer Scope"
    var outer2 = "Also in Outer Scope"

    var internal = function() {
        'use strict'
        // block scope with let
        {
            let inner = "In Inner Scope"
            outer1 = "Foo"
        }
        //console.log("let does not hoist: " + outer2)
        let outer2 = "Redeclared"
        var outer3 = "Bar" // no auto-global scope
        console.log([inner, outer1, outer2, outer3])
    }
    internal()
}
```

Uncaught ReferenceError: inner is not defined

Variable Hoisting

```
function go() {
    var sum = 0;
    var value = 0;
    var prevValue = 1;
    for (var ii = 0; ii < 100;
        value = Math.floor(Mat
        var product = value *
        prevValue = value;
        sum += product;
    }
    console.log('The product was ' + product)
    console.log('The sum is ' + sum)
}
```

```
function go() {
    var product, ii
    var sum = 0;
    var value = 0;
    var prevValue = 1;
    for (ii = 0; ii < 100; ++ii) {
        value = Math.floor(Math.random()*10);
        product = value * prevValue;
        prevValue = value;
        sum += product;
    }
    console.log('The product was ' + product)
    console.log('The sum is ' + sum)
}
```

Variable Hoisting

```
hoist()
function hoist() {
    console.log("Inside hoist()", a)
    var a = "Hoist Me!"
}
hoist()

hoist2()
var hoist2 = function() {
    hoist()
}
```

a is hoisted, but has no value

hoist2 is hoisted, but has no value and therefore is not a function yet

```
Inside hoist() undefined
Inside hoist() undefined
▶ Uncaught TypeError: hoist2 is not a function
```

Closures

Closing over scope

```
function closed() {
    var i = 0;
    return function() {
        return ++i
    }
}
```

```
> plusOne = closed()
< function anonymous()
> plusOne.name
< ""
> plusOne()
< 1
> plusOne()
< 2
> plusOne()
< 3
```

Closures

```
function incrementerFactory(increment) {
    var value = 0
    var inc = increment ? increment : 1;
    return function() {
        value += inc
        return value;
    }
}

var plusOne = incrementerFactory()

var plusTwo = incrementerFactory(2)
```

```
> plusOne()
<· 1
> plusOne()
<· 2
> plusOne()
<· 3
> plusTwo()
<· 2
> plusTwo()
<· 4
> plusTwo()
<· 6
```

A Closure Approach to Privacy

```
> var obj = { i:1,
      increment: function() { return ++this.i } }
< undefined
> obj.increment()
< 2
> obj.increment()
< 3
> obj.increment()
< 4
> obj.i = 10
< 100
> obj.increment()
```

That's why we favor function over obejct: you wont be able to change internal value

```
function incrementerFactory(increment) {
    var value = 0;
    var inc = increment ? increment : 1;
    return {
        getValue: function() {
            return value
        },
        increment: function() {
            value += inc
            return value;
        }
    }
}
```

A Closure Approach to Privacy

```
> obj = incrementerFactory(3)
<· ▼Object {} i
     ▶getValue: function ()
     ▶increment: function ()
     ▶__proto__: Object
> obj.increment()
<· 3
> obj.increment()
<· 6
> obj.getValue()
<· 6
> obj.getValue() = 10
⊗ ▶Uncaught ReferenceError: Invalid
  left-hand side in assignment
```

```
function incrementerFactory(increment) {
    var value = 0;
    var inc = increment ? increment : 1;
    return {
        getValue: function() {
            return value
        },
        increment: function() {
            value += inc
            return value;
        }
    }
}
```

Functions and Constructors

Capitalize for class name

```
function Person(name) {
    var i = 0
    this.name = name
    this.increment = function() {
        return ++i
    }
}
```

Call as a function

```
> fn = Person("Max")
< undefined
> fn.name
✕ ▶ Uncaught TypeError: Cannot read property 'name' of undefined
```

Call as a constructor

```
> cons = new Person("Leo")
< ▶ Person {name: "Leo"}
> cons.name
< "Leo"
```

Functions and Constructors

```
function Person(name) {
    var i = 0
    this.name = name
    this.increment = function() {
        return ++i
    }
}
```

```
> cons = new Person("Leo")
< ▶ Person {name: "Leo"}
> cons.name
< "Leo"
```

```
> cons.increment()
< 1
> cons.name = "Scott"
< "Scott"
> cons
< ▶ Person {name: "Scott"}
> cons.i
< undefined
```

Because i is a variable rather than a field

Scope Gotcha!

```
function showHelp(help) {
  document.getElementById('help').innerHTML = help;
}

function setupHelp() {
  var helpText = [
      {'id': 'email', 'help': 'Your e-mail address'},
      {'id': 'name', 'help': 'Your full name'},
      {'id': 'age', 'help': 'Your age (you must be over 16)'}
    ];

  for (var i = 0; i < helpText.length; i++) {
    var item = helpText[i];
    document.getElementById(item.id).onfocus = function() {
      showHelp(item.help);
    }
  }
}

window.onload = setupHelp
```

Because item is hoisted!

Will always point to the last one

Your age (you must be over 16)

E-mail:

Name:

Age:

Context: What is *this*?

```
function Person(name) {
    var i = 0
    this.name = name
    this.increment = function() {
        return ++i
    }
    console.log(this)
}
```

Call as a function

```
> fn = Person('Max')

  ▶ Window {top: Window, Locat
  document: document, window:
  Object…}
```

Call as a constructor

```
> cons = new Person('Leo')
  ▶ Person {name: "Leo"}
<· ▶ Person {name: "Leo"}
```

Function Binding

```
function Person(name) {
    var i = 0
    this.name = name
    this.increment = function() {
        return ++i
    }
    //console.log(this)
    this.status = function(postfix) {
        return this.name + ' counted to '
            + i + (postfix ? postfix : '')
    }
}
```

```
> leo = new Person('Leo')
<· ▶ Person {name: "Leo"}
> leo.status()
<· "Leo counted to 0"
> scott = new Person('Scott')
<· ▶ Person {name: "Scott"}
> scott.increment()
```

Function Binding

```
function Person(name) {
    var i = 0
    this.name = name
    this.increment = function() {
        return ++i
    }
    //console.log(this)
    this.status = function(postfix) {
        return this.name + ' counted to '
            + i + (postfix ? postfix : '')
    }
}
```

```
> scott.status()
<- "Scott counted to 2"
> scott.status.call(leo, '?')
<- "Leo counted to 2?"
> leo.status()
<- "Leo counted to 0"
```

Immediately Invoked Function Expression (IIFE)

```
function incrementerFactory(incre
    var value = 0;
    var inc = increment ? increme
    return {
        getValue: function() {
            return value
        },
        increment: function() {
            value += inc
            return value;
        }
    }
}
```

```
> var incrementer = incrementerFactory(3)
< undefined
> incrementer.increment()
< 3
> var incrementer = (incrementerFactory)(3)
< undefined
> incrementer.increment()
< 3
```

Immediately Invoked Function Expression (IIFE)

```
function incrementerFactory(incre
    var value = 0;
    var inc = increment ? increme
    return {
        getValue: function() {
            return value
        },
        increment: function() {
            value += inc
            return value;
        }
    }
}
```

```
> var incrementer = incrementerFactory(3)
< undefined
> incrementer.increment()
< 3
> var incrementer = (incrementerFactory)(3)
< undefined
> incrementer.increment()
< 3
> var incrementer = (function() { return 5 } )(3)
< undefined
> incrementer
< 5
> var incrementer = (function(a) { return a+1 } )(3)
< undefined
> incrementer
< 4
```

IIFE

ASI+IIFE Gotcha!

```
var testIFFE = 'foo'

(function() {
    console.log('This is an IFFE!')
})()
```

```
▶ Uncaught TypeError: "foo" is not a function
```

IIFE is good since they wont pollute the name space and will get garbage collected right away

```
var testIFFE = 'foo'

;(function() {
    console.log('This is an IFFE!')
})()
```

```
This is an IFFE!
```

In-Class Exercise:
Get Started with Heroku

I have provided instructions for this assignment here:

https://www.clear.rice.edu/comp431/data/heroku.html

***Turnin* README.json *to* COMP431-S16:inclass-7**