ADVANCED JAVASCRIPT WORKSHOP

WHO AM I?

DAMJAN VUJNOVIC

damjan@samuraiprinciple.com

@returnthis

INTRODUCTION TO JASMINE

- JavaScript unit testing framework
-
- will be used throughout this course (test to learn)
- asynchronous code
- continuous integration

SUITES, SPECS AND EXPECTATIONS

```
describe('Arithmetic operators', function () {
  it('should add numbers using + operator', function () {
    expect(1 + 2).toBe(3);
  });
  it('should subtract numbers using - operator', function () {
    expect(7 - 2).toBe(5);
  });
  it('should fail', function () {
    expect(4 * 2).toBe(42);
  });
});
```

MORE MATCHERS

```
describe('Matchers', function () {
  it('should use toBe matcher when expecting same object', function () {
    var samurai = ['Myamoto', 'Hattori', 'Dangereous Dave'];
    expect(samurai).toBe(samurai);
    expect(samurai).not.toBe(['Myamoto', 'Hattori', 'Dangereous Dave']);
  });
  it('should use toEqual matcher when expecting equivalent object', function () {
    var samurai = ['Myamoto', 'Hattori', 'Dangereous Dave'];
    expect(samurai).toEqual(samurai);
    expect(samurai).toEqual(['Myamoto', 'Hattori', 'Dangereous Dave']);
  });
  it('should use toBeDefined matcher when not expecting undefined', function () {
    var nullSamurai = null,
    undefinedSamurai = undefined,
    evenMoreUndefinedSamurai;
    expect(nullSamurai).toBeDefined();
    expect(undefinedSamurai).not.toBeDefined();
    expect(evenMoreUndefinedSamurai).not.toBeDefined();
  });
  it('should use toBeNull matcher when expecting null', function () {
    var samurai = null, undefinedSamurai;
    expect(samurai).toBeNull();
    expect(undefinedSamurai).not.toBeNull();
  });
});
```

EVEN MORE MATCHERS

```
describe('Truthy and falsy values and expectations', function () {
  it('should use toBeTruthy/Falsy to check if something is truthy', function () {
    var result = '';
    if (1)
      result += 'number 1 ';
    expect(1).toBeTruthy();
    if (0)
      result += 'number 0';
    expect(0).toBeFalsy();
    if (NaN)
      result += 'not a number';
    expect(NaN).toBeFalsy();
    if ('false')
      result += 'false ';
    expect('false').toBeTruthy();
    if ('')
      result += 'empty string';
    expect('').toBeFalsy();
    if (null)
      result += 'null ';
    expect(null).toBeFalsy();
    if (undefined)
      result += 'undefined ';
    expect(undefined).toBeFalsy();
    expect(result).toBe('number 1 false ')
  });
});
```

EXAMPLE: OPERATOR == AND TYPE COERCION

```
describe('Operator == and type coercion', function () {
  it('Operator == behaves oddly because of type coercion', function () {
    expect(NaN == NaN).toBe(false);//not reflexive! hmmm...
    expect(Infinity == Infinity).toBe(true);
    expect('      ' == 0).toBe(true);
    expect('  \t \r \n' == 0).toBe(true);
    expect('' == 0).toBe(true);
    expect(0 == '0').toBe(true);
    expect('' == '0').toBe(false);//not transitive! hmmm...
    expect(false == '0').toBe(true);
    expect(null == undefined).toBe(true);
    expect(false == 'false').toBe(false);
    expect(false == null).toBe(false);
    expect(false == undefined).toBe(false);
  });
});
```

MORAL OF THE STORY:

- use === instead of ==

ASYNCHRONOUS SPECS

```
describe('Testing asynchronous code', function () {
  it('should wait until model.value is not undefined and invoke\
  function passed as runs parameter', function () {
    var model = {
      value: undefined,
      fetchFromServer: function () {
        setTimeout(function () {
          model.value = 123;
        }, 1);
      }
    };
    expect(model.value).not.toBeDefined();
    model.fetchFromServer();
    waitsFor(function () {
      return model.value !== undefined;
    }, 'value was never fetched from the server');
    expect(model.value).not.toBeDefined();//Explain this!!!
    runs(function () {
      expect(model.value).toBe(123);
    });
    expect(model.value).not.toBeDefined();
  });
});
```

OBJECTS

OBJECTS

Objects are mutable collection of properties

Each property has

- name
- value
- attributes

OBJECT PROPERTIES

property name can be any string, even empty string

property value can be any JavaScript value except undefined

CREATING NEW OBJECT VALUES

Using object literals

```
var person = {
  name: "John Doe",
  age: 23
};
```

Using `new` operator (constructor function)

```
var person = new Person("John Doe", 23);
```

OBJECT LITERALS

```
describe('Objects literals', function () {
  it('should create new object using object literals', function () {
    var samurai = {
      name: 'Myamoto',
      age: 32,
      address: {
        street: 'Ninja Way',
        postcode: 'PG 18+'
      },
      'funny-property name': 12345
    };
    expect(typeof(samurai)).toBe('object');
  });
});
```

RETRIEVING PROPERTY VALUE

```
describe('Retrieving object property values using [] and .', function () {
  it('should return property value', function () {
    var samurai = {
      name: 'Yamamoto',
      age: 12,
      address: {
        street: 'Ninja Way',
        postcode: 'PG 333'
      }
    };
    expect(samurai.name).toBe('Yamamoto');
    expect(samurai['name']).toBe('Yamamoto');
    expect(samurai.address.street).toBe('Ninja Way');
  });
});
```

RETRIEVING PROPERTY VALUE

```
describe('Retrieving object property values using [] and .', function () {
  it('should use [] when properties are not legal names', function () {
    var samurai = {
      '1stName': 'Myamoto',
      'second name': 'Musashi',
      'home-telephone': '555-12345'
    };
    expect(samurai['1stName']).toBe('Myamoto');
    expect(samurai['second name']).toBe('Musashi');
    expect(samurai['home-telephone']).toBe('555-12345');
  });
});
```

DEFAULT OPERATOR

Don't do this (awful)

```
if (person.address === undefined) {
  address = "N/A";
} else {
  address = person.address;
}
```

or this (a little bit better)

```
address = person.address === undefined ? "N/A" : person.address;
```

DEFAULT OPERATOR

```
describe('Default operator ||', function () {
  it('should return default if left operand is null or undefined', function () {
    var samurai1 = {
      name: 'Myamoto',
      mobilePhoneNumber: '555-1234'
    }, samurai2 = {
      name: 'Hattori'
    };
    expect(samurai1.mobilePhoneNumber || 'N/A').toBe('555-1234');
    expect(samurai2.mobilePhoneNumber || 'N/A').toBe('N/A');
  });
});
```

DEFAULT OPERATOR - BE CAREFUL WITH 0

```
describe('Default operator ||', function () {
  it('be careful with other falsy values (0, empty string, ...)', function () {
    var account = {
      balance: 0
    };
    expect(account.balance || 'N/A').toBe('N/A');
  });
});
```

GUARD OPERATOR

don't do this (awful)

```
if (person !== undefined && person.address !== undefined) {
  postcode = person.address.postcode;
}
```

or this (still awful):

```
postcode = person !== undefined && person.address !== undefined ?
  person.address.postcode :
  undefined;
```

also, we (potentially) have to check for nulls too

GUARD OPERATOR

```
describe('Guard operator &&', function () {
  it('should guard you against null and undefined', function () {
    var s0, s1 = {
      name: 'Yamamoto'
    }, s2 = {
      name: 'Hattori',
      address: {
        street: 'Samurai Way'
      }
    }, s3 = {
      name: 'Samyaki',
      address: {
        street: 'Ninja Avenue',
        postcode: 'XX1Y22'
      }
    };
    expect(s0 && s0.address && s0.address.postcode).toBe(undefined);
    expect(s1 && s1.address && s1.address.postcode).toBe(undefined);
    expect(s2 && s2.address && s2.address.postcode).toBe(undefined);
    expect(s3 && s3.address && s3.address.postcode).toBe('XX1Y22');
  });
});
```

UPDATING PROPERTY VALUE

```
describe('Updating property value', function () {
  it('should update existing property value', function () {
    var samurai = {
      name: 'Myamoto'
    };
    samurai.name = 'Hattori';
    expect(samurai.name).toBe('Hattori');
  });
});
```

ADDING NEW PROPERTIES

```
describe('Adding new properties', function () {
  it('should add new properties', function () {
    var samurai = {
      name: 'Myamoto'
    };
    samurai.age = 20;
    samurai['is-male'] = true;
    expect(samurai.age).toBe(20);
    expect(samurai['is-male']).toBe(true);
  });
});
```

ADDING NEW PROPERTIES

```
describe('Adding new properties', function () {
  it('should add new property, value is an object', function () {
    var samurai = {
      name: "Myamoto"
    };
    samurai.address = {
      street: "Elm Street",
      postcode: "XX1Y22"
    };
    expect(samurai.address.postcode).toBe("XX1Y22");
  });
});
```

ENUMERATION

```
describe('Enumerating properties', function () {
  it('should use for in to enumerate properties of an object', function () {
    var samurai = {
      name: 'Myamoto',
      age: 32
    }, propertyName, result = '',
    optionA = 'name=Myamoto;age=32;',
    optionB = 'age=32;name=Myamoto;';
    for (propertyName in samurai) {
      result += propertyName + '=' + samurai[propertyName] + ';';
    }
    expect(result === optionA || result === optionB).toBe(true);
  });
});
```

DELETE

```
describe('Delete operator', function () {
  it('should remove property from the object', function () {
    var samurai = {
      name: 'Myamoto'
    };
    expect(samurai.name).toBe('Myamoto');
    expect(true).toBe(delete samurai.name);
    expect(samurai.name).toBe(undefined);
  });
});
```

PROPERTY ATTRIBUTES

- ReadOnly
- DontEnum
- DontDelete
- Internal

EXECUTION CONTEXTS

- global
- function
- eval

Don't confuse execution context with variable scope!

DELETE OPERATOR

In script block:

```
var x = 1;
expect(false).toBe(delete x);
expect(1).toBe(x);
```

In (older) Firebug console (or when using eval):

```
var x = 1;
expect(true).toBe(delete x);
expect(undefined).toBe(x);
```

In script block:

```
window.x = 1;
expect(true).toBe(delete window.x);
expect(undefined).toBe(window.x);
```

FUNCTIONS

- PART 1 -

FUNCTIONS ARE 'FIRST CLASS'

- can be stored in variables, objects & arrays
- can be passed as a parameter to a function
- can be returned as result of a function
- can have methods
- plus, they can be invoked
- are linked to Function.prototype (linked to Object.prototype)

FUNCTION LITERAL

- function fName (param1, param2) { /.../ }
- reserved word function
- optional name (otherwise anonymous; debugging)
- parameters (between ())
- body (between {})

FUNCTION LITERAL: EXAMPLE

```
describe('Function Literal', function () {
  it('should create a function', function () {
    var greet = function (name) {
      return 'Hello ' + name;
    };
    expect(greet('Myamoto')).toBe('Hello Myamoto');
  });
  it('should create a function too', function () {
    function greet (name) {
      return 'Hello ' + name;
    };
    expect(greet('Myamoto')).toBe('Hello Myamoto');
  });
});
```

FUNCTION DECLARATION HOISTING

```
describe('Function Declaration hoisting', function () {
  it('should be undefined', function () {
    var f;
    expect(f).toBe(undefined);
    f = function (name) {
      return 'Hello ' + name;
    };
  });
  it('should be defined even though it appears to be unreachable', function () {
    expect(f('Myamoto')).toBe('Hello Myamoto');
    return;
    function f(name) {
      return 'Hello ' + name;
    };
  });
});
```

FUNCTION DECLARATION HOISTING (CONTD.)

```
describe('Function Declaration hoisting', function () {
  it('should not be possible to have conditional declarations', function () {
    var flag = true;
    if (flag) {
      function f (name) {
        return 'Hi ' + name;
      }
    } else {
      function f (name) {
        return 'Hello ' + name;
      }
    }
    expect(f('Myamoto')).toBe('Hello Myamoto');//passes in Chrome, fails in FF
  });
});
```

FUNCTION PARAMETERS

```
describe('Parameters', function () {
  var returnParameters = function (first, second) {
    return [first, second];
  };
  it('should return both parameters', function () {
    var parameters = returnParameters('first', 'second');
    expect(returnParameters.length).toBe(2);
    expect(parameters[0]).toBe('first');
    expect(parameters[1]).toBe('second');
  });
  it('should be able to pass less parameters', function () {
    var parameters = returnParameters('first');
    expect(parameters[0]).toBe('first');
    expect(parameters[1]).toBe(undefined);
  });
  it('should be able to pass (but not retrieve) more parameters', function () {
    var parameters = returnParameters('first', 'second', 'third');
    expect(parameters[0]).toBe('first');
    expect(parameters[1]).toBe('second');
    expect(parameters[2]).toBe(undefined);
  });
});
```

PUZZLE TIME: OVERWRITING UNDEFINED

```
describe('Overwriting undefined (same can be done with NaN)', function () {
  var isUndefined = function (value) {
    return value === undefined;
  };
  it('should behave strangely', function () {
    var x, oldUndefined = undefined;
    expect(isUndefined(x)).toBe(true);
    undefined = 3;
    expect(isUndefined(x)).toBe(false);
    expect(isUndefined(3)).toBe(true);
    undefined = oldUndefined;
  });
});
```

OVERWRITING UNDEFINED - SOLUTION

```
describe('Overwriting undefined', function () {
  var isUndefined = function (value, u) {
    return value === u;
  };
  it('should behave as expected', function () {
    var x, oldUndefined = undefined;
    expect(isUndefined(x)).toBe(true);
    undefined = 3;
    expect(isUndefined(x)).toBe(true);
    expect(isUndefined(3)).toBe(false);
    undefined = oldUndefined;
  });
});
```

TWO IMPLICIT PARAMETERS

Whenever a function is invoked, two implicit parameters are passed too:

- `arguments`
- `this`

IMPLICIT PARAMETER - ARGUMENTS

```
describe('Implicit parameter arguments', function () {
  var returnArguments = function (first, second) {
    return arguments;
  };
  it('should be array-like object', function () {
    var args = returnArguments('first', 'second');
    expect([].push).not.toBe(undefined);
    expect(args.push).toBe(undefined);
  });
  it('should contain both parameters', function () {
    var args = returnArguments('first', 'second');
    expect(args).toEqual(['first', 'second']);
  });
  it('should be able to pass less parameters', function () {
    var args = returnArguments('first');
    expect(args).toEqual(['first']);
  });
  it('should be able to pass (and retrieve) more parameters', function () {
    var args = returnArguments('first', 'second', 'third');
    expect(args).toEqual(['first', 'second', 'third']);
  });
});
```

HOW CAN WE USE IT?

```
describe('String formatter', function () {
  var format = function () {
    var result = arguments[0], i;
    for (i = 1; i < arguments.length; i += 1) {
      result = result.replace('%', arguments[i]);
    }
    return result;
  };
  it('should format string without params', function () {
    expect(format('Hello')).toBe('Hello');
  });
  it('should format string with one param', function () {
    expect(format('Hello %!', 'World')).toBe('Hello World!');
  });
  it('should format string with three params', function () {
    expect(format('x=% y=% z=%', 1, 2, 3)).toBe('x=1 y=2 z=3');
  });
});
```

WHAT IS `THIS` BOUND TO?

Depends on the invocation pattern:

- method
- function
- constructor
- call/apply

METHOD INVOCATION PATTERN

```
describe('Method Invocation Pattern', function () {
  it('should bind this to the samurai object', function () {
    var samurai = {
      setName: function (value) {
        expect(this).toBe(samurai);
        this.name = value;
      }
    };
    expect(samurai.name).toBe(undefined);
    samurai.setName('Myamoto');
    expect(samurai.name).toBe('Myamoto');
  });
});
```

FUNCTION INVOCATION PATTERN

```
describe('Function Invocation Pattern', function () {
  it('should bind this to the global object', function () {
    var setName = function (value) {
      this.name = value;
    };
    setName('Myamoto');
    expect(window.name).toBe('Myamoto');
  });
});
```

FUNCTION INVOCATION PATTERN GOTCHA

```
describe('Function Invocation Pattern Gotcha', function () {
  it('should pollute global namespace', function () {
    var samurai = {
      setName: function (value) {
        var setFirstAndLastName = function (firstName, lastName) {
          this.firstName = firstName;
          this.lastName = lastName;
        }, names = value.split(' ');
        if (names.length === 2) {
          setFirstAndLastName(names[0], names[1]);
        } else {
          setFirstAndLastName(names[1], names[2]);
        }
      }
    };
    samurai.setName('Mr. Myamoto Musashi');
    expect(window.firstName).toBe('Myamoto');
    expect(window.lastName).toBe('Musashi');
  });
});
```

SOLUTION 1

```
describe('Function Invocation Pattern Gotcha', function () {
  it('should not pollute global namespace', function () {
    var samurai = {
      setName: function (value) {
        var that = this, setFirstAndLastName = function (firstName, lastName) {
          that.firstName = firstName;
          that.lastName = lastName;
        }, names = value.split(' ');
        if (names.length === 2) {
          setFirstAndLastName(names[0], names[1]);
        } else {
          setFirstAndLastName(names[1], names[2]);
        }
      }
    };
    samurai.setName('Mr. Myamoto Musashi');
    expect(samurai.firstName).toBe('Myamoto');
    expect(samurai.lastName).toBe('Musashi');
  });
});
```

SOLUTION 2

```
describe('Function Invocation Pattern Gotcha', function () {
  it('should not pollute global namespace', function () {
    var samurai = {
      setName: function (value) {
        var setFirstAndLastName = function (firstName, lastName) {
          samurai.firstName = firstName;
          samurai.lastName = lastName;
        }, names = value.split(' ');
        if (names.length === 2) {
          setFirstAndLastName(names[0], names[1]);
        } else {
          setFirstAndLastName(names[1], names[2]);
        }
      }
    };
    samurai.setName('Mr. Myamoto Musashi');
    expect(samurai.firstName).toBe('Myamoto');
    expect(samurai.lastName).toBe('Musashi');
  });
});
```

HOW DID FUNCTION INVOCATION RUIN MY DATE?

- using window.name for session variables
- this in this.name = ... got bound to global namespace
- 100x more session state requests
- BOOM!

MORAL OF THE STORY

- monitor global namespace pollution from your unit-tests
- fail the test if anything gets added or removed

CONSTRUCTOR INVOCATION PATTERN

```
describe('Constructor Invocation Pattern', function () {
  it('should bind this to the object that will be created', function () {
    var Samurai = function (name) {
      this.name = name;
    }, samurai;
    samurai = new Samurai('Myamoto');
    expect(samurai.name).toBe('Myamoto');
  });
});
```

CONSTRUCTOR INVOCATION PATTERN GOTCHA

```
describe('Constructor Invocation Pattern Gotcha', function () {
  it('should bind this to the global object if new is omitted', function () {
    var Samurai = function (name) {
      this.name = name;
    }, samurai;
    samurai = Samurai('Myamoto');
    expect(samurai).toBe(undefined);
    expect(window.name).toBe('Myamoto');
  });
});
```

CONSTRUCTOR INVOCATION PATTERN GOTCHA

```
describe('Constructor Invocation Pattern Gotcha', function () {
  it('should bind this to the object that will be created', function () {
    var Samurai = function (name) {
      if (!(this instanceof Samurai)) {
        return new Samurai(name);
      }
      //can use arguments.callee
      this.name = name;
    }, samurai1, samurai2;
    samurai1 = new Samurai('Myamoto');
    samurai2 = Samurai('Myamoto');
    expect(samurai1.name).toBe('Myamoto');
    expect(samurai2.name).toBe('Myamoto');
  });
});
```

CALL/APPLY INVOCATION PATTERN

```
describe('Apply Invocation Pattern', function () {
  it('should bind this to the object that is passed as a parameter', function () {
    var samurai = {
      name: 'Myamoto'
    }, setName = function (value) {
      this.name = value;
    };
    setName.call(samurai, 'Hattori');
    expect(samurai.name).toBe('Hattori');
    setName.apply(samurai, ['Myamoto']);
    expect(samurai.name).toBe('Myamoto');
  });
});
```

HOW DO WE USE IT?

```
describe('Apply Invocation Pattern', function () {
  it('should demonstrate Math.max function', function () {
    expect(Math.max(10, 20, 30)).toBe(30);
  });
  it('should return maximum element in an array', function () {
    var max = function (arr) {
      return Math.max.apply(null, arr);
    };
    expect(max([10, 20, 30])).toBe(30);
  });
});
```

ANOTHER SOLUTION FOR FIRST/LAST NAME GOTCHA

```
describe('Function Invocation Pattern Gotcha', function () {
  it('should not pollute global namespace', function () {
    var samurai = {
      setName: function (value) {
        var setFirstAndLastName = function (firstName, lastName) {
          this.firstName = firstName;
          this.lastName = lastName;
        }, names = value.split(' ');
        if (names.length === 2) {
          setFirstAndLastName.call(this, names[0], names[1]);
        } else {
          setFirstAndLastName.call(this, names[1], names[2]);
        }
      }
    };
    samurai.setName('Mr. Myamoto Musashi');
    expect(samurai.firstName).toBe('Myamoto');
    expect(samurai.lastName).toBe('Musashi');
  });
});
```

RETURN

```
describe('Return', function () {
  it('should return undefined if there is no return statement', function () {
    var samurai = {
      setName: function (value) {
        this.name = value;
      }
    };
    expect(samurai.setName('Myamoto')).toBe(undefined);
  });
});
```

RETURN AND CONSTRUCTOR INVOCATION

```
describe('Return', function () {
  it('should return the created object\
    if function is invoked as a constructor', function () {
    var innerThis, Samurai = function (name) {
      innerThis = this;
      this.name = name;
    }, samurai = new Samurai('Myamoto');
    expect(innerThis).toBe(samurai);
  });
});
```

RETURN AND CONSTRUCTOR INVOCATION

```
describe('Return', function () {
  it('should return the created object if function is invoked as a constructor\
  unless function returns an object', function () {
    var innerThis, Samurai = function (name) {
      innerThis = this;
      this.name = name;
      return { name: 'Hattori' };
    }, samurai = new Samurai('Myamoto');
    expect(samurai.name).toBe('Hattori');
  });
});
```

RETURN AND CONSTRUCTOR INVOCATION

```
describe('Return', function () {
  it('should return the created object if function is invoked as a constructor\
  unless function returns an object', function () {
    var innerThis, Samurai = function (name) {
      innerThis = this;
      this.name = name;
      return 0;
    }, samurai = new Samurai('Myamoto');
    expect(samurai.name).toBe('Myamoto');
  });
});
```

PROTOTYPE

```
describe('Prototype', function () {
  var isEmpty = function (object) {
    var name;
    for (name in object) {
      return false;
    }
    return true;
  };
  it('when function is declared it gets a prototype property', function () {
    var f = function () {
    };
    expect(typeof f.prototype).toBe('object');
    expect(isEmpty(f.prototype)).toBe(true);
  });
});
```

PROTOTYPE

```
describe('Prototype', function () {
  it('instance inherits prototype properties', function () {
    var Person = function () {
    }, instance;
    Person.prototype.name = 'default name';
    instance = new Person();
    expect(instance.name).toBe('default name');

    expect(Person.prototype.isPrototypeOf(instance)).toBe(true);
    expect({}.isPrototypeOf(instance)).toBe(false);
  });
});
```

PROTOTYPE

```
describe('Prototype', function () {
  it('the link between instance and prototype is "live"', function () {
    var Person = function () {
    }, instance;
    instance = new Person();//we create instance first
    Person.prototype.name = 'default name';//and then augment the prototype
    expect(instance.name).toBe('default name');
  });
});
```

PROTOTYPE

```
describe('Prototype', function () {
  it('instance can override prototype properties', function () {
    var Person = function () {
    }, firstInstance, secondInstance;
    Person.prototype.name = 'default name';
    firstInstance = new Person();
    secondInstance = new Person();
    firstInstance.name = 'new name';
    expect(firstInstance.name).toBe('new name');
    expect(secondInstance.name).toBe('default name');
    expect(Person.prototype.name).toBe('default name');
  });
});
```

AUGMENTING BUILT-IN PROTOTYPES

```
describe('Prototype', function () {
  Object.prototype.isEmpty = function () {
    var name;
    for (name in this) {
      if (this.hasOwnProperty(name)) {
        return false;
      }
    }
    return true;
  }
  it('should be able to augment built-in prototypes', function () {
    var emptyObject = {}, nonEmptyObject = { name: 'Myamoto' };
    expect(emptyObject.isEmpty()).toBe(true);
    expect(nonEmptyObject.isEmpty()).toBe(false);
  });
});
```

DELETE

```
describe('Prototype', function () {
  it('when property is deleted, prototype property may shine through', function () {
    var Person = function (name) {
      this.name = name;
    }, instance;
    Person.prototype.name = 'default name';
    instance = new Person('Myamoto');
    expect(instance.name).toBe('Myamoto');
    delete instance.name;
    expect(instance.name).toBe('default name');
    //think of a transparent foils stack
  });
});
```

ENUMERATION

```
describe('Prototype', function () {
  it('enumeration', function () {
    var Person = function (age) {
      this.age = age;
    }, instance, properties = '', propertyName;
    Person.prototype.name = 'default name';
    Person.prototype.age = 0;
    instance = new Person(32);
    for (propertyName in instance) {
      properties += propertyName;
    }
    expect(properties === 'agename' || properties === 'nameage').toBe(true);

    expect(instance.hasOwnProperty('age')).toBe(true);
    expect(instance.hasOwnProperty('name')).toBe(false);

    expect(instance.propertyIsEnumerable('age')).toBe(true);
    expect(instance.propertyIsEnumerable('name')).toBe(false);//!!!
  });
});
```

ARRAYS

ARRAY LITERALS

```
describe('Array Literals', function () {
  var isArray = function (value) {
    // does not work if value is created in different frame or window
    return value &&
      typeof value === 'object' &&
      value.constructor === Array;
  };
  var isArray2 = function (value) {
    return toString.call(value) === "[object Array]";
  };
  it('should create an empty array', function () {
    var emptyArray = [];
    expect(isArray(emptyArray)).toBe(true);
  });
  it('should create a non-empty array', function () {
    var weapons = ['katana', 'wakizashi'];
    expect(isArray(weapons)).toBe(true);
  });
  it('should create an array with elements of different types', function () {
    var samuraiItems = [3, 'katana', { name: 'Miyamoto' }, [1, 2, 3], window.alert];
    expect(isArray(samuraiItems)).toBe(true);
  });
  it('should create a sparse array', function () {
    var friends = ['Hattori Hanzo', , 'Takeda Shingen'];
    expect(isArray(friends)).toBe(true);
  });
});
```

RETRIEVING ARRAY ELEMENTS

```
describe('Retrieving elements', function () {
  it('should retrive elements using [] operator', function () {
    var samurai = ['Hattori Hanzo', 'Date Masamune', 'Shimazo Yashihiro'];
    expect(samurai[0]).toBe('Hattori Hanzo');
    expect(samurai[1]).toBe('Date Masamune');
    expect(samurai[2]).toBe('Shimazo Yashihiro');
    expect(samurai[3]).toBe(undefined);
  });
  it('should retrive missing elements as undefined', function () {
    var samurai = ['Hattori Hanzo', , 'Date Masamune', 'Shimazo Yashihiro'];
    expect(samurai[1]).toBe(undefined);
  });
});
```

LENGTH

```
describe('Length', function () {
  it('should return number of elements in (non-sparse) array', function () {
    var samurai = ['Hattori Hanzo', 'Date Masamune', 'Shimazo Yashihiro'];
    expect(samurai.length).toBe(3);
  });
  it('should return index of last element plus one in sparse array', function () {
    var samurai = ['Hattori Hanzo', , 'Date Masamune', , , 'Shimazo Yashihiro'];
    expect(samurai.length).toBe(6);
  });
  it('should ignore trailing comma (one) at the end of the array', function () {
    var samurai = ['Hattori Hanzo', , 'Date Masamune', 'Shimazo Yashihiro', , , ];
    expect(samurai.length).toBe(6);
  });
  it('should be able to increase length of an array', function () {
    var samurai = ['Hattori Hanzo', 'Date Masamune', 'Shimazo Yashihiro'];
    samurai.length = 4;
    expect(samurai.length).toBe(4);
  });
  it('should be able to decrease length of an array', function () {
    var samurai = ['Hattori Hanzo', 'Date Masamune', 'Shimazo Yashihiro'];
    samurai.length = 2;
    expect(samurai.length).toBe(2);
    expect(samurai[2]).toBe(undefined);
  });
});
```

MODIFYING ARRAYS

```
describe('Modifying', function () {
  it('should be able to modify array element', function () {
    var samurai = ['Hattori Hanzo', 'Date Masamune', 'Shimazo Yashihiro'];
    samurai[1] = 'Miyamoto Musashi';
    expect(samurai[1]).toBe('Miyamoto Musashi');
  });
  it('should be able to modify array element', function () {
    var samurai = ['Hattori Hanzo', 'Date Masamune', 'Shimazo Yashihiro'];
    samurai[3] = 'Miyamoto Musashi';
    expect(samurai.length).toBe(4);
    expect(samurai[3]).toBe('Miyamoto Musashi');
  });
  it('should be able to modify array element if index <= 4294967295', function () {
    var samurai = ['Hattori Hanzo', 'Date Masamune', 'Shimazo Yashihiro'];
    samurai[4294967294] = 'Miyamoto Musashi';
    expect(samurai[4294967294]).toBe('Miyamoto Musashi');
    expect(samurai.length).toBe(4294967295);
    samurai[4294967295] = 'Miyamoto Musashi';
    expect(samurai[4294967295]).toBe('Miyamoto Musashi');
    expect(samurai.length).toBe(4294967295);
  });
});
```

DELETING ELEMENTS

```
describe('Delete', function () {
  it('should only change value for specified name to undefined', function () {
    var samurai = [
      'Hattori Hanzo',
      'Date Masamune',
      'Shimazo Yashihiro',
      'Miyamoto Musashi'
    ];
    delete samurai[1];
    expect(samurai[1]).toBe(undefined);
    expect(samurai.length).toBe(4);
  });
});
```

DELETING ELEMENTS (CONTD.)

```
describe('Splice', function () {
  it('should remove element and compact the array', function () {
    var samurai = [
      'Hattori Hanzo',
      'Date Masamune',
      'Shimazo Yashihiro',
      'Miyamoto Musashi'
    ];
    samurai.splice(2, 1);
    expect(samurai[2]).toBe('Miyamoto Musashi');
    expect(samurai.length).toBe(3);
  });
});
```

ITERATING - USING FOR LOOP

```
describe('Iterating arrays using for loop', function () {
  it('should use for loop', function () {
    var samurai = ['Hattori', , 'Date', 'Shimazo', 'Miyamoto'],
      i, all = '', length = samurai.length;
    for (i = 0; i < length; i += 1) {
      all += samurai[i] + ' ';
    }
    expect(all).toBe('Hattori undefined Date Shimazo Miyamoto ');
  });
});
```

ITERATING - USING FOR-IN LOOP

```
describe('Iterating arrays using for-in loop', function () {
  it('should consider using for-in for sparse arrays', function () {
    var samurai = ['Hattori', 'Date', 'Shimazo'], name, all = '';
    samurai[1000000000] = 'Miyamoto';
    for (name in samurai) {
      all += samurai[name] + ' ';
    }
    expect(all).toBe('Hattori Date Shimazo Miyamoto ');
  });
});
```

FOR-IN GOTCHAS

```
describe('Iterating arrays using for-in loop - gotchas', function () {
  it('should be aware that order is not deterministic', function () {
    var samurai = [], name, all = '';
    samurai[1000000000] = 'Shimazo';//try with 2 instead
    samurai[0] = 'Hattori';
    for (name in samurai) {
      all += samurai[name] + ' ';
    }
    expect(all).toBe('Hattori Shimazo ');//passes in Chrome, fails in FF
  });
  it('should be aware of unwanted properties', function () {
    var samurai = ['Hattori', 'Date'], name, all = '';
    samurai['name'] = 'Damjan';
    for (name in samurai) {
      all += samurai[name] + ' ';
    }
    expect(all).toBe('Hattori Date Damjan ');
  });
});
```

ARRAY METHODS - CONCAT & JOIN

```
describe('Array Methods - concat & join', function () {
  it('should join two arrays', function () {
    var array = [10, 20, 30];
    expect(array.concat([40, 50, 60])).toEqual([10, 20, 30, 40, 50, 60]);
  });
  it('should join elements of an array', function () {
    var array = [10, 20, 30];
    expect(array.join('.')).toBe('10.20.30');
  });
});
```

ARRAY METHODS - PUSH & POP

```
describe('Array Methods - push & pop', function () {
  it('should add one or more elements to the end of the array', function () {
    var array = [10, 20, 30], newLength;
    newLength = array.push(40, 50);
    expect(newLength).toBe(5);
    expect(array).toEqual([10, 20, 30, 40, 50]);
  });
  it('should remove (and return) last element from array', function () {
    var array = [10, 20, 30], lastElement;
    lastElement = array.pop();
    expect(lastElement).toBe(30);
    expect(array).toEqual([10, 20]);
  });
});
```

ARRAY METHODS - SHIFT & UNSHIFT

```
describe('Array Methods - shift & unshift', function () {
  it('should remove (and return) first element from array', function () {
    var array = [10, 20, 30], firstElement;
    firstElement = array.shift();
    expect(firstElement).toBe(10);
    expect(array).toEqual([20, 30]);
  });
  it('should add one or more elements to the front of the array\
    and return new array', function () {
    var array = [10, 20, 30], newLength;
    newLength = array.unshift(-10, 0);
    expect(newLength).toBe(5);
    expect(array).toEqual([-10, 0, 10, 20, 30]);
  });
});
```

ARRAY METHODS - SLICE & SPLICE

```
describe('Array Methods - reverse & sort', function () {
  it('should extract a section of the array and return a new array', function () {
    var array = [10, 20, 30, 40, 50], smallArray;
    smallArray = array.slice(2, 4);
    expect(array).toEqual([10, 20, 30, 40, 50]);
    expect(smallArray).toEqual([30, 40]);
  });
  it('should add and/or remove elements from array', function () {
    var array = [10, 20, 30, 40, 50];
    array.splice(1, 2, 21, 31, 39);
    expect(array).toEqual([10, 21, 31, 39, 40, 50]);
  });
});
```

ARRAY METHODS - REVERSE & SORT

```
describe('Array Methods - slice & splice', function () {
  it('should transpose the elements of an array', function () {
    var array = [10, 20, 30, 40, 50];
    array.reverse();
    expect(array).toEqual([50, 40, 30, 20, 10]);
  });
  it('should sort the elements of an array', function () {
    var array = [20, 50, 10, 30, 40];
    array.sort();
    expect(array).toEqual([10, 20, 30, 40, 50]);
    array.sort(function(first, second) {
      return second - first;
    });
    expect(array).toEqual([50, 40, 30, 20, 10]);
  });
});
```

ARRAY METHODS IN JAVASCRIPT 1.6+

```
describe('Array Methods Introduced in JavaScript 1.6+', function () {
  it('should sum the squares of all the odd array elements', function () {
    var array = [1, 2, 3, 4, 5], result;
    result = array.filter(function (item) {
      return item % 2 === 1;
    }).map(function (item) {
      return item * item;
    }).reduce(function (first, second) {
      return first + second;
    }, 0);
    expect(result).toBe(35);
  });
});
```

TWO-DIMENSIONAL ARRAYS

```
describe('Two-Dimensional Arrays', function () {
  it('should use array of arrays', function () {
    var array = [], row, column;
    for (row = 0; row < 3; row += 1) {
      array[row] = [];
      for (column = 0; column < 4; column += 1) {
          array[row][column] = row + '-' + column;
      }
    }
    expect(array).toEqual([
      ['0-0', '0-1', '0-2', '0-3'],
      ['1-0', '1-1', '1-2', '1-3'],
      ['2-0', '2-1', '2-2', '2-3']
    ]);
  });
});
```

FUNCTIONS

- PART 2 -

NO BLOCK SCOPE

```
describe('No block scope', function () {
  it('should demonstrate that JavaScript has no block scope', function () {
    var name = 'Hattori',
    f = function () {
      expect(name).toBe(undefined);
      var name = 'Myamoto';
      expect(name).toBe('Myamoto');
    };
    expect(name).toBe('Hattori');
    f();
    expect(name).toBe('Hattori');
  });
});
```

CLOSURE

```
describe('Closure', function () {
  it('should demonstrate that inner functions have access to parameters and \
  local variables of the functions they are defined within', function () {
    var name = 'Hattori',
    f = function () {
      expect(name).toBe('Hattori');
    };
    f();
  });
});
```

CLOSURE - NOT A COPY

```
describe('Lexical Scope', function () {
  it('should demonstrate that JavaScript has no block scope', function () {
    var name = 'Hattori',
    before = function () {
      expect(name).toBe('Hattori');
    },
    after = function () {
      expect(name).toBe('Myamoto');
    };
    before();
    name = 'Myamoto';
    after();
  });
});
```

CLOSURE - LIFETIME

```
function doAjax() {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "test.txt", true);
  xhr.onreadystatechange = function() {
    if (xhr.readyState === 4) {
      alert(xhr.responseText);
    }
  };
  xhr.send(null);
}
```

CLOSURE - LIFETIME

```
describe('Closure', function () {
  var samuraiBuilder = function (name) {
    return {
      getName: function () {
        return name;
      }
    };
  };
  it('should demonstrate that instances will close over different names',
    function () {
      var myamoto = samuraiBuilder('Myamoto'),
      hattori = samuraiBuilder('Hattori');
      expect(myamoto.getName()).toBe('Myamoto');
      expect(hattori.getName()).toBe('Hattori');
    }
  );
});
```

CLOSURE GOTCHA

```
var page1 = document.getElementById('page1');
document.getElementById('setupHandlers1').onclick = function (event) {
  var buttons = page1.getElementsByTagName('button'), i;
  for (i = 0; i < buttons.length; i += 1) {
    buttons[i].onclick = function(event) {
      alert('clicked ' + i);
    };
  }
};
```

CLOSURE GOTCHA - SOLUTION

```
var page2 = document.getElementById('page2');
document.getElementById('setupHandlers2').onclick = function (event) {
  var buttons = page2.getElementsByTagName('button'),
  createHandler = function (message) {
    console.log('createHandler', message);
    return function (event) {
      console.log('onclick', message);
      alert(message);
    };
  }, i;
  for (i = 0; i < buttons.length; i += 1) {
    console.log('loop', i);
    buttons[i].onclick = createHandler('clicked ' + i);
  }
}
```

CLOSURE GOTCHA - TWO MORE SOLUTIONS

```
var page2 = document.getElementById('page2');
document.getElementById('setupHandlers2').onclick = function (event) {
  var buttons = page2.getElementsByTagName('button'), i;
  for (i = 0; i < buttons.length; i += 1) {
    buttons[i].onclick = (function (message) {
      return function (event) {
        alert(message);
      };
    })('clicked ' + i);
  }
}
```

```
var page2 = document.getElementById('page2');
document.getElementById('setupHandlers2').onclick = function (event) {
  var buttons = page2.getElementsByTagName('button'), i;
  for (i = 0; i < buttons.length; i += 1) {
    buttons[i].onclick = (function () {
      var message = 'clicked ' + i;
      return function (event) {
        alert(message);
      };
    })();
  }
}
```

(FUNCTION (){})() - AS MODULE PATTERN

```
var SAMURAIPRINCIPLE = {
  /* this acts as a module/namespace */
};

(function () {
  var privateVariable = 'asdf',
    privateFunction = function (arg1, arg2) {
      privateVariable = arg1;
      ...
    };
  SAMURAIPRINCIPLE.publicFunction = function () {
    privateFunction('hello', 'world');
    privateVariable = '!';
    ...
  };
  SAMURAIPRINCIPLE.PublicConstructor = function () {
    ...
  };
})();

var result = SAMURAIPRINCIPLE.publicFunction();
var instance = new SAMURAIPRINCIPLE.PublicConstructor();

/* privateVariable & privateFunction not visible here */
```

LET'S HAVE A PEEK AT JQUERY SOURCE

```
(function (window, undefined) {
  var jQuery = ...
  ...
  //uses both window & undefined in many places
  ...
  window.jQuery = window.$ = jQuery;
})(window);
```

JQUERY MINIFIED

```
(function (w, u) {
  var j = ...
  w.jQuery = w.$ = j;
})(window);
```

FIBONACCI NUMBERS

```
describe('Fibonacci', function () {
  var fibonacci = function (n) {
    if (n < 2) {
      return n;
    } else {
      return fibonacci(n - 1) + fibonacci(n - 2);
    }
  };
  it('should return n-th Fibonacci number', function () {
    var i, sequence = [0, 1, 1, 2, 3, 5, 8, 13, 21];
    for (i = 0; i < sequence.length; i++) {
      expect(fibonacci(i)).toBe(sequence[i]);
    }
  });
});
```

FIBONACCI NUMBERS IMPROVED

```
describe('Fibonacci with memoization', function () {
  var fibonacci = (function () {
    var lookup = {
      "0": 0,
      "1": 1
    };
    return function fib(n) {
      if (lookup[n] !== undefined) {
        return lookup[n];
      } else {
        return lookup[n] = fib(n - 1) + fib(n - 2);
      }
    };
  })();
  it('should return n-th Fibonacci number', function () {
    var i, sequence = [0, 1, 1, 2, 3, 5, 8, 13, 21];
    for (i = 0; i < sequence.length; i++) {
      expect(fibonacci(i)).toBe(sequence[i]);
    }
  });
});
```

MEMOIZATION

```
describe('Memoization', function () {
  var memoize = function (lookup, fn) {
    return function (n) {
      return lookup[n] === undefined ? lookup[n] = fn(n) : lookup[n];
    };
  };
  it('should return n-th Fibonacci number', function () {
    var fibonacci = memoize({ "0": 0, "1": 1 }, function (n) {
      return fibonacci(n - 1) + fibonacci(n - 2);
    });
    expect(fibonacci(8)).toBe(21);
  });
  it('should return n!', function () {
    var factorial = memoize({ "0": 1 }, function (n) {
      return n * factorial(n - 1);
    });
    expect(factorial(6)).toBe(720);
  });
});
```

BIND - THE PROBLEM

```
describe('Bind', function () {
  var controller = {
    click: function () {
      this.clicked = true;
      //this.doSomething();
    },
    doSomething: function () {
    }
  }, button = document.createElement('button');
  it('should update clicked on button instead of controller', function () {
    button.onclick = controller.click;
    button.onclick();
    expect(controller.clicked).toBe(undefined);
    expect(button.clicked).toBe(true);
    delete button.clicked;
  });
});
```

BIND - IMPLEMENTATION

```
describe('Bind', function () {
  Function.prototype.bind = function () {
    var fn = this,
    slice = Array.prototype.slice,
    args = slice.call(arguments),
    object = args.shift();
    return function () {
      return fn.apply(object, args.concat(slice.call(arguments)));
    };
  };
  it('should bind a function to a particular object', function () {
    var setName = function (value) {
      this.name = value;
    };
    var samurai1 = {},
    samurai2 = {},
    setSamurai1Name = setName.bind(samurai1),
    setSamurai2Name = setName.bind(samurai2, 'Hattori');
    setSamurai1Name('Myamoto');
    expect(samurai1.name).toBe('Myamoto');
    setSamurai2Name();
    expect(samurai2.name).toBe('Hattori');
  });
});
```

BIND - IMPLEMENTATION WITHIN MODULE

```
(function () {
  var slice = Array.prototype.slice;
  Function.prototype.bind = function () {
    var fn = this,
    args = slice.call(arguments),
    object = args.shift();
    return function () {
      return fn.apply(object, args.concat(slice.call(arguments)));
    };
  };
})();
```

BIND - SOLUTION

```
describe('Bind', function () {
  var controller = {
    click: function () {
      this.clicked = true;
      //this.doSomething();
    },
    doSomething: function () {
    }
  }, button = document.createElement('button');
  it('should bind this to controller when invoking click() function', function () {
    button.onclick = controller.click.bind(controller);
    button.onclick();
    expect(controller.clicked).toBe(true);
    expect(button.clicked).toBe(undefined);
  });
});
```

OOP

CONFUSION

- How do I do OOP without classes?
- OOP is not about classes - it's about message passing
- Singleton "pattern"
- Observer

WE'LL TALK ABOUT WAYS OF

- creating objects
- acheiving 'inheritance'

CREATING OBJECTS

- constructor function
- factory function (or method)

CONSTRUCTOR FUNCTION

```
describe('Constructor function', function () {
  var Samurai = function (name) {
    this.sayHello = function () {
      return "Hello " + name;
    };
    this.setName = function (value) {
      name = value;
    };
  };
  it('should be able to create new objects using constructor function', function () {
    var myamoto = new Samurai('Myamoto');
    expect(myamoto.sayHello()).toBe('Hello Myamoto');
  });
});
```

CONSTRUCTOR FUNCTION WITH PROTOTYPE

```
describe('Constructor function', function () {
  var Samurai = function (name) {
    this.name = name;
  };
  Samurai.prototype.sayHello = function () {
    return "Hello " + this.name;
  };
  Samurai.prototype.setName = function (value) {
    this.name = value;
  };
  it('should be able to create new objects using constructor function', function () {
    var myamoto = new Samurai('Myamoto');
    expect(myamoto.sayHello()).toBe('Hello Myamoto');
  });
});
```

FACTORY FUNCTION

```
describe('Factory function', function () {
  var createSamurai = function (name) {
    var result = {};
    result.sayHello = function () {
      return "Hello " + name;
    };
    result.setName = function (value) {
      name = value;
    };
    return result;
  };
  it('should be able to create new objects using factory function', function () {
    var myamoto = createSamurai('Myamoto');
    expect(myamoto.sayHello()).toBe('Hello Myamoto');
  });
});
```

FACTORY FUNCTION

```
describe('Factory function', function () {
  var createSamurai = function(name) {
    return {
      sayHello: function() {
        return "Hello " + name;
      },
      setName: function(value) {
        name = value;
      }
    };
  };
  it('should be able to create new objects using factory function', function () {
    var myamoto = createSamurai('Myamoto');
    expect(myamoto.sayHello()).toBe('Hello Myamoto');
  });
});
```

LET'S PUSH THIS A BIT FURTHER - MIXIN

```
describe('Factory function', function () {
  var samuraiMixin = function(host, name) {
    host.sayHello = function () {
      return "Hello " + name;
    };
    host.setName = function (value) {
      name = value;
    };
  };
  it('should use mixin to add features to an existing object', function () {
    var myamoto = {};
    samuraiMixin(myamoto, 'Myamoto');
    expect(myamoto.sayHello()).toBe('Hello Myamoto');
  });
});
```

INHERITANCE

- Pseudo-classical
- Prototypal
- Functional

PSEUDOCLASSICAL INHERITANCE - BASE CLASS

```
var Proxy = function (type, id, onChangeCallback) {
  this.type = type;
  this.id = id;
  this.onChangeCallback = onChangeCallback;
  this.synchronize = function () {
    var self = this;
    jQuery.ajax({
      url: '/' + self.type + '/' + self.id,
      success: function (data) {
        self.onChangeCallback(data);
      }
    });
  };
};
```

EXTENDED CLASS

```
var Account = function (id) {
  this.id = id;
  this.onChangeCallback = function (data) {
    this.currentBalance = data.currentBalance;
  };
  this.getCurrentBalance = function () {
    return this.currentBalance;
  };
};
Account.prototype = new Proxy('account');
Account.prototype.constructor = Account;
```

EXTENDED CLASS

```
var Bet = function (id) {
  this.id = id;
  this.onChangeCallback = function (data) {
    this.status = data.status;
  };
  this.getStatus = function () {
    return this.status;
  };
};
Bet.prototype = new Proxy('bet');
Bet.prototype.constructor = Bet;
```

LET'S HIDE THIS PROTOTYPE BUSINESS

```
var extends = function (Child, Parent) {
  Child.prototype = new Parent();
  Child.prototype.constructor = Child;
};
```

MOVE SHARED STUFF IN PROTOTYPE

```
var Proxy = function () {
};
Proxy.prototype.synchronize = function () {
  var self = this;
  jQuery.ajax({
    url: '/' + self.type + '/' + self.id,
    success: function (data) {
      self.onChangeCallback(data);
    }
  });
};
```

MOVE SHARED STUFF IN PROTOTYPE

```
var Account = function (id) {
  this.id = id;
};
extends(Account, Proxy);
Account.prototype.type = 'account';
Account.prototype.onChangeCallback = function (data) {
  this.currentBalance = data.currentBalance;
};
Account.prototype.getCurrentBalance = function () {
  return this.currentBalance;
};
```

MOVE SHARED STUFF IN PROTOTYPE

```
var Bet = function (id) {
  this.id = id;
};
extends(Bet, Proxy);
Bet.prototype.type = 'bet';
Bet.prototype.onChangeCallback = function (data) {
  this.status = data.status;
};
Bet.prototype.getStatus = function () {
  return this.status;
};
```

LET'S TRY AND INHERIT DIRECTLY FROM PROTOTYPE

```
var extends = function (Child, Parent) {
  Child.prototype = Parent.prototype;
  Child.prototype.constructor = Child;
};
var Parent = function () {
};
Parent.prototype.name = 'Parent';
var FirstChild = function () {
};
extends(FirstChild, Parent);
FirstChild.prototype.name = 'FirstChild';
var SecondChild = function () {
};
extends(SecondChild, Parent);
SecondChild.prototype.name = 'Second';
var parent = new Parent(),
firstChild = new FirstChild(),
secondChild = new SecondChild();
```

INHERIT FROM PROTOTYPE - SOLUTION

```
var extends = function (Child, Parent) {
  var Temp = function () {
  };
  Temp.prototype = Parent.prototype;
  Child.prototype = new Temp();
  Child.prototype.constructor = Child;
};
var Parent = function () {
};
Parent.prototype.name = 'Parent';
var FirstChild = function () {
};
extends(FirstChild, Parent);
FirstChild.prototype.name = 'FirstChild';
var SecondChild = function () {
};
extends(SecondChild, Parent);
SecondChild.prototype.name = 'Second';
var parent = new Parent(),
firstChild = new FirstChild(),
secondChild = new SecondChild();
```

ENCAPSULATE THIS PROTOTYPE BUSINESS

```
Function.prototype.member = function (name, value) {
  this.prototype[name] = value;
  return this;
};
Function.member('extends', function (Parent) {
  var Temp = function () {
  };
  Temp.prototype = Parent.prototype;
  this.prototype = new Temp();
  this.prototype.constructor = this;
  return this;
});
```

BASE CLASS

```
var Proxy = function () {
}.member('synchronize', function () {
  var self = this;
  jQuery.ajax({
    url: '/' + self.type + '/' + self.id,
    success: function (data) {
      self.onChangeCallback(data);
    }
  });
});
```

EXTENDED CLASS

```
var Account = function (id) {
  this.id = id;
}.extends(Proxy)
.member('type', 'account')
.member('onChangeCallback', function (data) {
  this.currentBalance = data.currentBalance;
})
.member('getCurrentBalance', function () {
  return this.currentBalance;
});
```

EXTENDED CLASS

```
var Bet = function (id) {
  this.id = id;
}.extends(Proxy)
.member('type', 'bet')
.member('onChangeCallback', function (data) {
  this.status = data.status;
})
.member('getStatus', function () {
  return this.status;
});
```

INVOKING METHOD FROM BASE CLASS - UBER

```
Function.prototype.member = function (name, value) {
  this.prototype[name] = value;
  return this;
};
Function.member('extends', function (Parent) {
  var Temp = function () {
  };
  Temp.prototype = Parent.prototype;
  this.prototype = new Temp();
  this.prototype.constructor = this;
  this.uber = Parent.prototype;
  return this;
});
Parent = function () {
}.member('name', 'Parent')
.member('getName', function () {
  var uber = this.constructor.uber;
  return (uber ? uber.name + '.' : '') + this.name;
});
Child = function () {
}.extends(Parent)
.member('name', 'Child');
```

ASYNCHRONOUS PROGRAMMING PATTERNS

WE'LL TALK ABOUT:

- Asynchronous method (callback)
- Observable objects
- Promise (future, deferrable)

SYNCHRONOUS (BLOCKING) I/O (JAVA)

```
HttpClient httpclient = new DefaultHttpClient();
HttpGet httpget = new HttpGet("http://localhost/");
HttpResponse response = httpclient.execute(httpget);
HttpEntity entity = response.getEntity();
if (entity != null) {
  InputStream instream = entity.getContent();
  int l;
  byte[] tmp = new byte[2048];
  while ((l = instream.read(tmp)) != -1) {
  }
}
```

CALLBACK - XMLHTTPREQUEST

```
var req = new XMLHttpRequest();
req.open('GET', '/currentBalance.php', true);
req.onreadystatechange = function () {
  if (req.readyState === 4 && (req.status === 200 || req.status === 304)) {
    document.getElementById('currentBalance').innerHTML = req.responseText;
  }
};
req.send(null);
```

WRAP IT UP INTO NICE ABSTRACTION - JQUERY.AJAX

```
jQuery.ajax({
  url: '/currentBalance.php',
  success: function (data) {
    jQuery('#currentBalance').html(data);
  }
});
```

JQUERY.AJAX IS A NICE ABSTRACTION:

- deals with browser differences
- encapsulates logic around statuses (readyState & status)
- cross-domain
- encoding parameters
- JSON deserialization
- ...

IT DOES AWSOME JOB, BUT:

- it's rather low-level
- are there no higher level abstractions?
- do all our tests have to be asynchronous?
- do all our tests have to depend on DOM?
- divide-and-conquer, but how?

OBSERVABLE OBJECT

CURRENT BALANCE WIDGET

- setting $('.currentBalance') to 123.00 once is fine
- what we usually mean is: 'follow' the current balance
- think Excel and A1 = B1 + C1
- VHDL, Verilog (anyone?)
- data flows and the propagation of change

CURRENT BALANCE WIDGET (JQUERY PLUGIN)

```
jQuery.fn.extend({
  currentBalanceWidget: function (account) {
    return this.each(
      function () {
        var widget = jQuery(this);
        account.onBalanceChanged(function (currentBalance) {
          widget.find('.balance').text(currentBalance);
        });
      }
    );
  }
});
...
(function () {
  var account = ...;//This is our observable object!!!
  jQuery('#currentBalance').currentBalanceWidget(account);
}());
```

WE WANT TO USE IT AS A MIXIN

```
/*global beforeEach, describe, expect, it, SAMURAIPRINCIPLE */
describe('eventDispatcher', function () {
    'use strict';
    it('should use eventDispatcher as a mixin', function () {
        var base = {}, result;

        result = SAMURAIPRINCIPLE.eventDispatcher(base);

        expect(result).toBe(base);
    });
});
```

IMPLEMENTATION

```
var SAMURAIPRINCIPLE = {};
SAMURAIPRINCIPLE.eventDispatcher = function (base) {
  return base;
};
```

ADDING EVENT LISTENER

```
/*global beforeEach, describe, expect, it, SAMURAIPRINCIPLE */
describe('eventDispatcher', function () {
    'use strict';
    it('should use addEventListener method to add event listener', function () {
        var underTest = SAMURAIPRINCIPLE.eventDispatcher({}),
            listener = function () {};

        underTest.addEventListener(listener);

        expect(underTest.listener()).toEqual(listener);
    });
});
```

IMPLEMENTATION

```
var SAMURAIPRINCIPLE = {};
SAMURAIPRINCIPLE.eventDispatcher = function (base) {
  var eventListener;
  base.addEventListener = function (listener) {
    eventListener = listener;
  };
  base.listener = function () {
    return eventListener;
  };
  return base;
};
```

DISPATCHING AN EVENT

```
/*global describe, expect, it, jasmine, SAMURAIPRINCIPLE */
describe('eventDispatcher', function () {
    'use strict';
    it('should use dispatchEvent to invoke registered listener', function () {
        var underTest = SAMURAIPRINCIPLE.eventDispatcher({}),
            result,
            listener = function () {
                result = 'listenerInvoked';
            };
        underTest.addEventListener(listener);

        underTest.dispatchEvent('argument');

        expect(result).toBe('listenerInvoked');
    });
    //Same test, but using a Jasmine spy
    it('should use dispatchEvent to invoke registered listener', function () {
        var underTest = SAMURAIPRINCIPLE.eventDispatcher({}),
            listener = jasmine.createSpy();
        underTest.addEventListener(listener);
        underTest.dispatchEvent('argument');
        expect(listener).toHaveBeenCalledWith('argument');
    });
});
```

IMPLEMENTATION

```
var SAMURAIPRINCIPLE = {};
SAMURAIPRINCIPLE.eventDispatcher = function (base) {
  var eventListener;
  base.addEventListener = function (listener) {
    eventListener = listener;
  };
  base.listener = function () {
    return eventListener;
  };
  base.dispatchEvent = function () {
    eventListener.apply({}, arguments);
  };
  return base;
};
```

SUPPORT MULTIPLE SUBSCRIBERS

```
/*global beforeEach, describe, expect, it, jasmine, SAMURAIPRINCIPLE */
describe('eventDispatcher', function () {
    'use strict';
    it('should be able to add multiple listeners', function () {
        var underTest = SAMURAIPRINCIPLE.eventDispatcher({}),
            firstListener = jasmine.createSpy(),
            secondListener = jasmine.createSpy();
        underTest.addEventListener(firstListener);
        underTest.addEventListener(secondListener);

        underTest.dispatchEvent('argument');

        expect(firstListener).toHaveBeenCalledWith('argument');
        expect(secondListener).toHaveBeenCalledWith('argument');
    });
});
```

IMPLEMENTATION

```
var SAMURAIPRINCIPLE = {};
SAMURAIPRINCIPLE.eventDispatcher = function (base) {
  var eventListeners = [];
  base.addEventListener = function (listener) {
    eventListeners.push(listener);
  };
  base.listeners = function () {
    return eventListeners;
  };
  base.dispatchEvent = function () {
    var i;
    for (i = 0; i < eventListeners.length; i += 1) {
      eventListeners[i].apply({}, arguments);
    }
  };
  return base;
};
```

SUPPORT DIFFERENT EVENT TYPES

```
/*global beforeEach, describe, expect, it, jasmine, SAMURAIPRINCIPLE */
describe('eventDispatcher', function () {
    'use strict';
    it('should be able to add listener for an event type', function () {
        var underTest = SAMURAIPRINCIPLE.eventDispatcher({}),
            listenerOnTypeA = jasmine.createSpy(),
            listenerOnTypeB = jasmine.createSpy();
        underTest.addEventListener('TypeA', listenerOnTypeA);
        underTest.addEventListener('TypeB', listenerOnTypeB);

        underTest.dispatchEvent('TypeA', 'argument');

        expect(listenerOnTypeA).toHaveBeenCalledWith('argument');
        expect(listenerOnTypeB).not.toHaveBeenCalled();
    });
});
```

IMPLEMENTATION

```
var SAMURAIPRINCIPLE = {};
SAMURAIPRINCIPLE.eventDispatcher = function (base) {
  var eventListenersByType = {};
  base.addEventListener = function (type, listener, priority) {
    if (!listener) { listener = type; type = 'DefaultType'; }
    eventListenersByType[type] = eventListenersByType[type] || [];
    eventListenersByType[type].push(listener);
  };
  base.listeners = function (type) {
    return eventListenersByType[type || 'DefaultType'] || [];
  };
  base.dispatchEvent = function () {
    var eventArguments, eventType, listeners, i;
    if (arguments.length === 1) {
      eventArguments = arguments;
      eventType = 'DefaultType';
    } else {
      eventArguments = Array.prototype.slice.call(arguments, 1);
      eventType = arguments[0];
    }
    listeners = base.listeners(eventType);
    for (i = 0; i < listeners.length; i++) {
      listeners[i].apply({}, eventArguments);
    }
  }; return base;
};
```

MIS-BEHAVING LISTENERS

```
/*global beforeEach, describe, expect, it, jasmine, SAMURAIPRINCIPLE */
describe('eventDispatcher', function () {
    'use strict';
    it('should invoke all listeners even if one of them throws an error', function () {
        var underTest = SAMURAIPRINCIPLE.eventDispatcher({}),
            badListener = jasmine.createSpy().andThrow('Error!'),
            goodListener = jasmine.createSpy();
        underTest.addEventListener('EventType', badListener);
        underTest.addEventListener('EventType', goodListener);

        underTest.dispatchEvent('EventType', 'argument');

        expect(badListener).toHaveBeenCalledWith('argument');
        expect(goodListener).toHaveBeenCalledWith('argument');
    });
});
```

JUST ADD TRY-CATCH BLOCK?

```
...
base.dispatchEvent = function () {
  ...
  listeners = base.listeners(eventType);
  for (i = 0; i < listeners.length; i++) {
    try {
      listeners[i].apply({}, eventArguments);
    } catch (e) {
    }
  }
};
...
```

TRY WITH TIMERS?

```
...
base.dispatchEvent = function () {
  ...
  listeners = base.listeners(eventType);
  for (i = 0; i < listeners.length; i++) {
    setTimeout(
      (function (listener) {
        return function () {
          listener.apply({}, eventArguments);
        };
      }(listeners[i])), 0
    );
  }
};
...
```

ANOTHER ATTEMPT WITH TIMERS...

```
...
base.dispatchEvent = function () {
  ...
  listeners = base.listeners(eventType);
  for (i = 0; i < listeners.length; i++) {
    try {
      listeners[i].apply({}, eventArguments);
    } catch (e) {
      setTimeout(
        function () {
          throw e;
        }, 0
      );
    }
  }
};
...
```

DEAN EDWARDS'S IDEA

```
var currentHandler;
if (document.addEventListener) {
  document.addEventListener("fakeEvents", function() {
    // execute the callback
    currentHandler();
  }, false);
  var dispatchFakeEvent = function() {
    var fakeEvent = document.createEvent("UIEvents");
    fakeEvent.initEvent("fakeEvents", false, false);
    document.dispatchEvent(fakeEvent);
  };
} else { // MSIE
  // I'll show this code later
}
var onLoadHandlers = [];
function addOnLoad(handler) {
  onLoadHandlers.push(handler);
};
onload = function() {
  for (var i = 0; i < onLoadHandlers.length; i++) {
    currentHandler = onLoadHandlers[i];
    dispatchFakeEvent();
  }
};
```

DEAN EDWARDS'S IDEA (CONTD.)

```
var currentHandler;
if (document.addEventListener) {
  // We've seen this code already
} else if (document.attachEvent) { // MSIE
  document.documentElement.fakeEvents = 0; // an expando property
  document.documentElement.attachEvent("onpropertychange", function(event) {
    if (event.propertyName == "fakeEvents") {
      // execute the callback
      currentHandler();
    }
  });
  dispatchFakeEvent = function(handler) {
    // fire the propertychange event
    document.documentElement.fakeEvents++;
  };
}
```

LISTENER PRIORITY

```
/*global describe, expect, it, SAMURAIPRINCIPLE */
describe('eventDispatcher', function () {
    'use strict';
    it('should be able to specify the order in which listeners are invoked,\
        by setting priority', function () {
        var underTest = SAMURAIPRINCIPLE.eventDispatcher({}),
            result = ':',
            lowPriorityListener = function () { result += 'first:'; },
            highPriorityListener = function () { result += 'second:'; };
        underTest.addEventListener('EventType', lowPriorityListener, 1);
        underTest.addEventListener('EventType', highPriorityListener, 2);

        underTest.dispatchEvent('EventType', 'argument');

        expect(result).toBe(':second:first:');
    });
});
```

IMPLEMENTATION

```
var SAMURAIPRINCIPLE = {};
SAMURAIPRINCIPLE.eventDispatcher = function (base) {
  var eventListenersByType = {};
  base.addEventListener = function (type, listener, priority) {
    if (!listener) {
      listener = type;
      type = 'DefaultType';
    }
    if (!priority) {
      priority = 0;
    }
    eventListenersByType[type] = eventListenersByType[type] || [];
    eventListenersByType[type][priority] = eventListenersByType[type][priority] || [];
    eventListenersByType[type][priority].push(listener);
  };
  base.listeners = function (type) {
    var listenersByType = eventListenersByType[type || 'DefaultType'] || [],
    result = [], i;
    for (i = listenersByType.length - 1; i >= 0; i -= 1) {
      Array.prototype.push.apply(result, listenersByType[i]);
    }
    return result;
  };
  ...
};
```

STOPPING EVENT PROPAGATION

```
/*global beforeEach, describe, expect, it, jasmine, SAMURAIPRINCIPLE */
describe('eventDispatcher', function () {
    'use strict';
    it('should be able to cancel event propagation by returning false from event listener', functi
        var underTest = SAMURAIPRINCIPLE.eventDispatcher({}),
            firstListener = jasmine.createSpy().andReturn(false),
            secondListener = jasmine.createSpy();
        underTest.addEventListener('EventType', firstListener);
        underTest.addEventListener('EventType', secondListener);
        underTest.dispatchEvent('EventType', 'argument');

        expect(firstListener).toHaveBeenCalledWith('argument');

        expect(secondListener).not.toHaveBeenCalled();
    });
});
```

IMPLEMENTATION

```
...
base.dispatchEvent = function () {
  ...
  listeners = base.listeners(eventType);
  for (i = 0; i < listeners.length; i++) {
    try {
      if (listeners[i].apply({}, eventArguments) === false) {
        break;
      }
    } catch (e) {
    }
  }
};
...
```

OBSERVABLE PROPERTIES

```
SAMURAIPRINCIPLE.eventDispatcher = function (base) {
  //...
  base.createObservableProperty = function (propertyName) {
    var propertyValue;
    base['get' + propertyName] = function () {
      return propertyValue;
    };
    base['set' + propertyName] = function (value) {
      if (propertyValue !== value) {
        propertyValue = value;
        base.dispatchEvent(propertyName + 'Changed', value);
      }
    };
    base['on' + propertyName + 'Changed'] = function (listener) {
      base.addEventListener(propertyName + 'Changed', listener);
    };//partial(base.addEventListener, propertyName + 'Changed');
    return base;
    //Also - take note that none of these methods is using 'this'
  };
  return base;
};
```

BACK TO OUR ACCOUNT OBJECT

```
SAMURAIPRINCIPLE.Account = function () {
  var self = this;
  eventDispatcher(this).createObservableProperty('CurrentBalance');
  jQuery.ajax({
    url: 'currentBalance.php',
    success: self.setCurrentBalance /*Can do this because it's not using 'this'*/
  });
};
```

MORE LIKELY TO BE SOMETHING LIKE THIS

```
SAMURAIPRINCIPLE.Account = function () {
  var self = this, interval, fetchBalance = function () {
    jQuery.ajax({
      url: 'currentBalance.php',
      success: self.setCurrentBalance(data)
    });
  };
  eventDispatcher(this).createObservableProperty('CurrentBalance');
  this.addEventListener('onCurrentBalanceListenerAdded', function () {
    if (!interval) {
      fetchBalance();
      interval = setInterval(fetchBalance, 10000);
    }
  });
};
```

JQUERY CUSTOM EVENTS

```
describe('Custom events in jQuery', function () {
  it('should use bind and trigger', function () {
    var observable = jQuery({}),
    listener = jasmine.createSpy();
    observable.bind('BalanceChanged', listener);
    observable.trigger('BalanceChanged', 1234.56);
    expect(listener).toHaveBeenCalled();
    expect(listener.mostRecentCall.args[1]).toBe(1234.56);
  });
});
```

NODEJS EVENTS.EVENTEMITTER

- addListener(event, listener), on(event, listener)
- once(event, listener)
- removeListener(event, listener)
- removeAllListeners(event)
- setMaxListeners(n)
- listeners(event)
- emit(event, [arg1], [arg2], [...])
- Event: 'newListener' function (event, listener) { }

NODEJS EXAMPLE

```
(function () {
  var net = require('net'), sys = require('sys'),
  stream = net.createConnection(80, 'localhost');
  stream.setEncoding('utf8');
  stream.on('connect', function () {
    sys.puts('- on connect');
    stream.write(['GET / HTTP/1.1',
      'Host: localhost',
      'Connection: close'
    ].join('\n') + '\n\n');
  });
  stream.on('data', function (data) {
    sys.puts('- on data');
    sys.puts(data);
  });
  stream.on('close', function (hadError) {
    sys.puts('- on close hadError=' + hadError);
  });
  stream.on('error', function (exception) {
    sys.puts('- on error');
    sys.puts(exception);
  });
  stream.on('end', function () { sys.puts('- on end'); });
}());
```

SUMMARY

- Loosely coupled components
- Makes testing easier
- Watch for memory leaks!
- Check for '(functional) reactive programming'

DEFERRED OBJECT

PRODUCT DETAILS CACHING

- ProductRepository returns products
- product can fetch its details
- product details don't change, so we want to cache them
- product we get from repository may already have its details

SOLUTION WITH OBSERVER

```
jQuery.fn.extend({
  productDetailsWidget: function (productRepository) {
    return this.each(
      function () {
        var widget = jQuery(this),
        productId = widget.attr('productId'),
        product = productRepository.getProduct(productId),
        showProductDetails = function (details) {
          if (!details)
            return;
          widget.find('.name').text(details.name);
          //...
        };
        showProductDetails(product.getDetails());
        product.addEventListener('DetailsLoaded',
          function (event) {
            showProductDetails(event.details);
          }
        );
      }
    );
  }
});
```

DEFERRED VALUES

- also called future/promise

SOLUTION WITH DEFERRED - WIDGET

```
jQuery.fn.extend({
  productDetailsWidget: function (productRepository) {
    return this.each(
      function () {
        var widget = jQuery(this);
        productRepository.getProduct(widget.attr('productId'))
          .when(function (productDetails) {
            widget.find('.name').text(productDetails.name);
          });
      }
    );
  }
});
```

SOLUTION WITH DEFERRED - REPOSITORY

```
var ProductRepository = function () {
  var productDetails = {};
  this.getProduct = function (productId) {
    var result = productDetails[productId];
    if (!result) {
      result = new Deferred();
      jQuery.ajax({
        url: '/product/' + productId,
        success: function (productDetails) {
          result.resolve(productDetails);
        }
      });
      productDetails[productId] = result;
    }
    return result;
  };
};
```

DEFERRED - DONE

```
describe('Deferred', function () {
  it('should be able to register a callback using done method', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred(), result;
    result = deferred.done(function () {});
    expect(result).toBe(deferred);
  });
});
```

DEFERRED - RESOLVE

```
describe('Deferred', function () {
  it('should be able to resolve it', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred(), result;
    result = deferred.resolve('argument');
    expect(result).toBe(deferred);
  });
});
```

DEFERRED - RESOLVE

```
describe('Deferred', function () {
  it('should invoke done callback when resolved', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred(),
    callback = jasmine.createSpy();
    deferred.done(callback);
    deferred.resolve('argument');
    expect(callback).toHaveBeenCalledWith('argument');
  });
});
```

DEFERRED - RESOLVE

```
describe('Deferred', function () {
  it('should invoke done callback imediately if already resolved', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred().resolve('argument'),
    callback = jasmine.createSpy();
    deferred.done(callback);
    expect(callback).toHaveBeenCalledWith('argument');
  });
});
```

DEFERRED - RESOLVE

```
describe('Deferred', function () {
  it('should be able to registere multiple callbacks', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred(),
    firstCallback = jasmine.createSpy(),
    secondCallback = jasmine.createSpy();
    deferred
      .done(firstCallback)
      .done(secondCallback)
      .resolve('argument');
    expect(firstCallback).toHaveBeenCalledWith('argument');
    expect(secondCallback).toHaveBeenCalledWith('argument');
  });
});
```

DEFERRED - RESOLVE

```
describe('Deferred', function () {
  it('should pass all the arguments when invoking registered callbacks', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred().resolve('arg1, 'arg2'),
    callback = jasmine.createSpy();
    deferred.done(callback);
    expect(callback).toHaveBeenCalledWith('arg1', 'arg2');
  });
});
```

DEFERRED - FAILED

```
describe('Deferred', function () {
  it('should be able to register failed callback', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred(), result;
    result = deferred.failed(function () {});
    expect(result).toBe(deferred);
  });
});
```

DEFERRED - REJECT

```
describe('Deferred', function () {
  it('should be able to reject it', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred(), result;
    result = deferred.reject('argument');
    expect(result).toBe(deferred);
  });
});
```

DEFERRED - REJECT

```
describe('Deferred', function () {
  it('should invoke failed callback when rejected', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred(),
    callback = jasmine.createSpy();
    deferred.failed(callback);
    deferred.reject('argument');
    expect(callback).toHaveBeenCalledWith('argument');
  });
});
```

DEFERRED - REJECT

```
describe('Deferred', function () {
  it('should invoke failed callback imediately if already rejected', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred(),
    callback = jasmine.createSpy();
    deferred.reject('argument');
    deferred.failed(callback);
    expect(callback).toHaveBeenCalledWith('argument');
  });
});
```

DEFERRED - REJECT

```
describe('Deferred', function () {
  it('should invoke all registered failed callbacks', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred().reject('argument'),
    firstCallback = jasmine.createSpy(),
    secondCallback = jasmine.createSpy();
    deferred
      .failed(firstCallback);
      .failed(secondCallback);
    expect(firstCallback).toHaveBeenCalledWith('argument');
    expect(secondCallback).toHaveBeenCalledWith('argument');
  });
});
```

DEFERRED - REJECT

```
describe('Deferred', function () {
  it('should pass all the arguments when invoking registered callbacks', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred().reject('arg1, 'arg2'),
    callback = jasmine.createSpy();
    deferred.failed(callback);
    expect(callback).toHaveBeenCalledWith('arg1', 'arg2');
  });
});
```

DEFERRED - REJECT

```
describe('Deferred', function () {
  it('should not invoke done callbacks when rejected', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred().reject('argument'),
    doneCallback = jasmine.createSpy(),
    failedCallback = jasmine.createSpy();
    deferred
      .done(doneCallback)
      .failed(failedCallback);
    expect(doneCallback).not.toHaveBeenCalled();
    expect(failedCallback).toHaveBeenCalledWith('argument');
  });
});
```

DEFERRED - THEN

```
describe('Deferred', function () {
  it('should be able to setup done and failed callbacks with then method', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred().resolve('argument'),
    doneCallback = jasmine.createSpy(),
    failedCallback = jasmine.createSpy();
    deferred.then(doneCallback, failedCallback);
    expect(doneCallback).toHaveBeenCalledWith('argument');
    expect(failedCallback).not.toHaveBeenCalled();
  });
});
```

DEFERRED - THEN

```
describe('Deferred', function () {
  it('should be able to setup done and failed callbacks with then method', function () {
    var deferred = new SAMURAIPRINCIPLE.Deferred().reject('arguments'),
    doneCallback = jasmine.createSpy(),
    failedCallback = jasmine.createSpy();
    deferred.then(doneCallback, failedCallback);
    expect(doneCallback).not.toHaveBeenCalled();
    expect(failedCallback).toHaveBeenCalledWith('argument');
  });
});
```

NEXTTOURNAMENTWIDGET - JUGGLES 2 DEFERREDS

- uses account and tournamentRepository
- account.getCurrentBalance() is a Deferred
- so is tournamentRepository.getNextTournament()
- we have to wait until both deferreds are resolved

SOMETHING LIKE THIS:

```
jQuery.fn.extend({
  nextTournamentWidget: function (account, tournamentRepository) {
    return this.each(
      function () {
        var widget = jQuery(this), balance, nextTournament,
        toResolve = 2, tryRender = function () {
          toResolve -= 1;
          if (!toResolve) {
            if (balance >= nextTournament.fee) {
              //
            } else {
              //
            }
          }
        };
        account.getBalance().done(function (b) {
          balance = b;
          tryRender();
        });
        tournamentRepository.getNextTournament().done(function (t) {
          nextTournament = t;
          tryRender();
        });
      }
    );
  }
});
```

WOULDN'T THIS BE NICE:

```
jQuery.fn.extend({
  nextTournamentWidget: function (account, tournamentRepository) {
    return this.each(
      function () {
        var widget = jQuery(this);
        SAMURAIPRINCIPLE.Deferred.when(
          account.getBalance(),
          tournamentRepository.getNextTournament())
          .done(function (balance, nextTournament) {
            if (balance >= nextTournament.fee) {
              //
            } else {
              //
            }
          });
      }
    );
  }
});
```

DEFERRED - WHEN

```
describe('Deferred.when', function () {
  it('should return Deferred that is resolved when all the deferreds are resolved', function () {
    var doneCallback = jasmine.createSpy(),
    failedCallback = jasmine.createSpy();
    SAMURAIPRINCIPLE.Deferred.when(
      (new SAMURAIPRINCIPLE.Deferred()).resolve(),
      (new SAMURAIPRINCIPLE.Deferred()).resolve(),
      (new SAMURAIPRINCIPLE.Deferred()).resolve()
    ).then(doneCallback, failedCallback);
    expect(doneCallback).toHaveBeenCalled();
    expect(failedCallback).not.toHaveBeenCalled();
  });
});
```

DEFERRED - WHEN

```
describe('Deferred.when', function () {
  it('should return Deferred that is rejected if one of the Deferreds is rejected', function () {
    var doneCallback = jasmine.createSpy(),
    failedCallback = jasmine.createSpy();
    SAMURAIPRINCIPLE.Deferred.when(
      (new SAMURAIPRINCIPLE.Deferred()).resolve(),
      (new SAMURAIPRINCIPLE.Deferred()).reject(),
      (new SAMURAIPRINCIPLE.Deferred()).resolve()
    ).then(doneCallback, failedCallback);
    expect(doneCallback).not.toHaveBeenCalled();
    expect(failedCallback).toHaveBeenCalled();
  });
});
```