Tags

[Pitfalls]
[TODO]

"asyncFun returns a collection. foreach i in collection doAsync with racing conditions." do not use async.

Node.js

How to create a npm package?	$ npm init
How to eval a single node expression at command line?	$ node -e 'console.log(require("./index"))'
How to load a module?	// require by module name var greet = require("greet"); // is the the same as requiring var greet = require("/path_to_greet_module/index.js");
How to set NODE_PATH for packages?	# The path might be different depending on where your node is installed. # You can put this in ~/.bashrc export NODE_PATH=~/.nvm/v0.10.26/lib/node_modules
How to link/unlink npm package?	$ cd targetModule $ npm link # 前提是 NODE_PATH 设置好了 $ npm unlink -g greet
Symbolic link vs. hard link?	A hardlink isn't a pointer to a file, it's a directory entry (a file) pointing to the same inode. Even if you change the name of the other file, a hardlink still points to the file. If you replace the other file with a new version (by copying it), a hardlink will not point to the new file. You can only have hardlinks within the same filesystem. With hardlinks you don't have concept of the original files and links, all are equal (think of it as a reference to an object). It's a very low level concept. On the other hand, a symlink is actually pointing to another path (a file name); it resolves the name of the file each time you access it through the symlink. If you move the file, the symlink will not follow. If you replace the file with another one, keeping the name, the symlink will point to the new file. Symlinks can span filesystems. With symlinks you have very clear distinction between the actual file and symlink, which stores no info beside the path about the file it points to.
Global vs. Local installation?	Global: {prefix}/lib/node_modules, If you're installing something that you want to use in your program, using require('whatever'), then install it locally, at the root of your project. Local: ./node_modules, If you're installing something that you want to use in your shell, on the command line or something, install it globally, so that its binaries end up in your PATH environment variable. http://blog.nodejs.org/2011/03/23/npm-1-0-global-vs-local-installation
What is the return value of require("moduleName") ?	Basically, whatever value you assign to module.exports is the value returned by require. https://nodejs.org/docs/latest/api/modules.html#modules_the_module_object
How to build npm package from existing files?	$ npm pack
How to install/uninstall npm tar file?	$ npm install [@<scope>/]<name> [--save|--save-dev|--save-optional] # --save-dev: Package will appear in your devDependencies. # install and save to package.json $ npm install minimist --save

	$ npm uninstall packageName
How to make package as executable?	in package.json "bin" : { "greet" : "./bin/greet.js" } bin/greet.js `#!` shebang directive #!/usr/bin/env node console.log("Hello World"); add executable to all users $ chmod a+x bin/greet.js
How to get command arguments?	process.argv ['node', '/opt/local/bin/mini-harp', '--port', '5000']
Package to parse command arguments?	https://github.com/substack/minimist **./bin/greet.js** #!/usr/bin/env node var g = require("../index.js"); var argv = require('minimist')(process.argv.slice(2)); // 从2开始的部分 console.log(g(argv._, argv.drunk)); **./index.js** module.exports = function greet(name, drunk) { if(drunk) { return "hello " + name + ", you look sexy today"; } else { return "hello, " + name; } }
How to access NODE_PATH in code?	https://github.com/joyent/node/blob/4d9c81b7e2522c5d5d9d35058cbb0bce1228d360/lib/module.js#L524-L527 var nodePath = process.env['NODE_PATH']; if (nodePath) { paths = nodePath.split(path.delimiter).concat(paths); }
How to launch tests with Mocha?	$ npm install mocha --save-dev Then in package.json, the `test` task runs the `mocha` command: { "name": "greet", ... "scripts": { "test": "mocha" }, ... } will run tests in `./test` dir
BDD vs. TDD?	Basically BDD is a feature driven approach to TDD. There is no difference between BDD and TDD. BDD is TDD done right.
How to assert tests with Chai?	$ npm install mocha chai --save-dev create ./test/greet_spec.js var expect = require("chai").expect;

	describe('greet', function(){ it("is a dummy success case", function(){ expect(1).to.eql(1); }); }); Run? $ npm test -R test/greet_spec.js References http://chaijs.com/api/bdd/
How to share global helper function?	e.g. ./test/support/helper.js global.expect = require("chai").expect; run by requiring it with mocha mocha --require test/support/helper
What is the difference between assert, expect and should in Chai?	assert.equal(3, '3', '== coerces values to strings'); var foo = 'bar'; expect(foo).to.equal('bar'); foo.should.equal('bar'); 最终行为上是一样的 不同在于: 1. The assert and expect interfaces do not modify Object.prototype, whereas should does. So they are a better choice in an environment where you cannot or do not want to change Object.prototype. 2. 可读性不一样 expect, should 更好读一点
What if you feel cumbersome with mocha cmd options?	Originally, run $ mocha --require test/support/helper --reporter spec 这里 spec 大概是所有以 spec 结尾的 tests specify options in ./test/mocha.opts --require test/support/helper --reporter spec Then, run $ mocha
Best practice : test with CoffeeScript and mocha	http://code.tutsplus.com/tutorials/better-coffeescript-testing-with-mocha--net-24696
Understanding `this` keyword	we use the this keyword as a shortcut, a referent; it refers to an object; that is, the subject in context, or the subject of the executing code. In the global scope, when the code is executing in the browser, all global variables and functions are defined on the window object. Therefore, when we use this in a global function, it refers to (and has the value of) the global window object. 1. this 默认是 global/window

	2. 通过一个对象调用一个函数, `this` 就是该对象 3. 除非由 apply/call 替换掉 this 会随着 context 的变化而变化! ``` var person = { firstName :"Penelope", lastName :"Barrymore", showFullName:function () { // The "context" console.log (this.firstName + " " + this.lastName); } } // Context is person person.showFullName (); // Penelope Barrymore var anotherPerson = { firstName :"Rohit", lastName :"Khan" }; // Context is now anotherPerson because anotherPerson invoked // the person.showFullName () method by virtue of using the apply () method // fun.apply(thisArg, [argsArray]) person.showFullName.apply (anotherPerson); // Rohit Khan ## Ref http://javascriptissexy.com/understand-javascripts-this-with-clarity-and-master-it/ http://www.quirksmode.org/js/this.html
bind	we can use bind (See: Function.prototype.bind()) to build a return a function no matter what its owner is ``` var bar = {name: "bar"}; // returnBar is a new function whose `this` is always 'bar' global.returnBar = returnThis.bind(bar) // The owner is global, but `this` is still bar. global.returnBar(); // => bar // The owner is foo, but `this` is still bar. foo.returnBar = returnBar(); foo.returnBar() // => bar
Implement _.once(func) which ensures that a function is called only once.	**// index.js** // 用闭包记录已经做过 ``` var _ = { once: function(fn) { var invoked = false; var rst; return function() { if(!invoked) { rst = fn(); invoked = true; } return rst; } } }; module.exports = _; ``` **// ./spec/once_spec.js** ``` var once = require("..").once; var assert = require("chai").assert;

```
describe("once",function() {
  it("should execute a function only once",function() {
    var i = 0;
    function add1() {
      return i += 1;
    }

    var addOnce = once(add1);

    addOnce();
    addOnce();
    addOnce();

    assert.equal(i,1);
  });

  it("should return the result of the invoked function", function() {
    var i = 0;
    function add1() {
      i += 1;
      return i;
    }

    var addOnce = once(add1);

    assert.equal(addOnce(),1);
    assert.equal(addOnce(),1);
    assert.equal(addOnce(),1);

  });

  it("should return undefined if function returns undefined",function() {
    function blah() {
      return;
    }

    var blahOnce = once(blah);

    assert.equal(blahOnce(),undefined);
  });

  it("should not invoke computation unless the returned function is
invoked",function() {
    var i = 0;
    function add1() {
      i += 1;
      return i;
    }

    once(add1);
    var addOnce = once(add1);

    assert.equal(i,0);
    addOnce();
    assert.equal(i,1);
  });
});
```

Implement _.memorize(func), which caches results

1. cache simply by func and one param

```
memoFoo = _.memoize(foo);
memoFoo(1); // calls and remembers
foo(1)
memoFoo(2); // calls and remembers
foo(2)
```

2. cache by func and a lambda to get the key

// index.js

```
// 用闭包记录已经做过
var _ = {
  memoize: function(fn, getKey) {
    var cache = {};

    return function(param) {
      if (cache[fn] == undefined) {
        cache[fn] = {};
      }

      var key = param;
      if (getKey) {
```

```
function identity(o) {
  return o;
}

memoizedIdentity =
memoize(identity,function(o) {
  return o[0]; // return the actual
key to identify
});

var user1 = [1,"Howard"];
var user2 = [1, "Howard Yeh"];

// The first call uses 1 as cache key
assert.deepEqual(memoizedIdentity(use
r1),user1);

// The second call returns the
previous value
assert.deepEqual(memoizedIdentity(use
r2),user1);
```

```
        key = getKey(param);
      }

      if (cache[fn][key] == undefined) {
        cache[fn][key] = fn(param);
      }
      return cache[fn][key];
    }
  }
};

module.exports = _;
```

// ./spec/memorize_spec.js

```
var memoize = require("..").memoize;
var assert = require("chai").assert;

describe("memoize",function() {
  it("should cache results",function() {
    var i = 0;
    function identity(n) {
      i = i + 1;
      return n;
    }

    memoizedIdentity = memoize(identity);

    assert.equal(memoizedIdentity(1),1);
    assert.equal(memoizedIdentity(1),1);
    assert.equal(i,1);

    assert.equal(memoizedIdentity(2),2);
    assert.equal(memoizedIdentity(2),2);
    assert.equal(i,2);

    assert.equal(memoizedIdentity(1),1);
    assert.equal(memoizedIdentity(2),2);
    assert.equal(i,2);
  });

  it("should use cache_key argument to calculate cache key",function() {
    function identity(o) {
      return o;
    }

    memoizedIdentity = memoize(identity,function(o) {
      return o[0];
    });

    var user1 = [1,"Howard"];
    var user2 = [1, "Howard Yeh"];
    assert.deepEqual(memoizedIdentity(user1),user1);
    assert.deepEqual(memoizedIdentity(user2),user1);

  });
});
```

Implement _.bind which returns a function forcing `this` in fn to be a particular object.

// index.js

```
var _ = {
  bind: function(fn, obj) {
    return function() {
      return fn.apply(obj);
    }
  }
};

module.exports = _;
```

// ./spec/bind_spec.js

```
var assert = require("chai").assert;

var _ = require("..");
```

	describe("bind",function() { it("should force this to be a context object",function() { function returnThis() { return this; } var foo = {name: "foo"}; var bar = {name: "bar"}; var returnFoo = _.bind(returnThis,foo); var returnBar = _.bind(returnThis,bar); assert.deepEqual(returnFoo(),foo); assert.deepEqual(returnBar(),bar); }); });
Coffee script class example	alert = (msg) -> console.log(msg) class Animal constructor: (@name) -> move: (meters) -> alert @name + " moved #{meters}m." class Snake extends Animal move: -> alert "Slithering..." super 5 class Horse extends Animal move: -> alert "Galloping..." super 45 sam = new Snake "Sammy the Python" tom = new Horse "Tommy the Palomino" sam.move() tom.move()
Implement Class Inheritance (with constructor and methods only) [Implement Class mechanism: Unit Tests]	var Class = function(options) { return function() { var ctorArgs = Array.prototype.slice.call (arguments, this.length); if (options.initialize != undefined) { options.initialize.apply(this, ctorArgs); } for (var prop in options) { if (options.hasOwnProperty(prop) && prop != "initialize") { this[prop] = options[prop]; } } }; }; module.exports = Class;
Implement Class Inheritance (with super) [Implement Class Inheritance: Unit Tests] 1. returnThis is a direct property of the object 2. returnThis is a property of the object's prototype 3. returnThis is a property of the	var Class = function(options, Parent) { Parent = Parent || Object; var Child = function () { var ctorArgs = Array.prototype.slice.call(arguments, this.length); if (options.initialize) { options.initialize.apply(this, ctorArgs); } }; Child.prototype = Object.create(Parent.prototype); Child.prototype.constructor = Child;

prototype's prototype

The key insight is that 1,2,3 all returns the object itself.

```
    Child.__super__ = Parent;
//    Child.prototype.super = function () {
//        var args = Array.prototype.slice.call(arguments, 1);
//
//        if (Child.__super__[arguments[0]]) {
//            return Child.__super__[arguments[0]].apply(this, args);
//        }
//    };
    // super 有可能会被反复 recursive call, 例如 c#foo() 中调用 this.super, 而
this 一直是 c, 所以需要在 super 的内部保证每次它
    // 被调用的时候返回不同的正确的结果
    Child.prototype.super = function() {
        var CurClass = Child; // a global var in all super()'s potential
recursions

        return function() {
            var tmp = CurClass;
            CurClass = CurClass.__super__;
            var args = Array.prototype.slice.call(arguments, 1);
            var rst = CurClass.prototype[arguments[0]].apply(this, args); //
potential recursion
            CurClass = tmp;
            return rst;
        };
    }();


    // Assign properties after super class's prototype for overriding.
    for (var prop in options) {
        if (!options.hasOwnProperty(prop)) {
            continue;
        }

        if (typeof options[prop] != "function") {
            Child[prop] = options[prop];
            continue;
        }

        if (prop == "initialize") {
            continue;
        }

        Child.prototype[prop] = options[prop];
        Child[prop] = options[prop];
    }

    return Child;
};

module.exports = Class;
```

[Implement Class mechanism: Unit Tests]

```
var expect = require("chai").expect;
var Class = require("../");

describe("Implement Class Constructor",function() {
  var constructor = function(a,b) {
    this.a = a;
    this.b = b;
  };

  var Foo = Class({initialize: constructor});

  it("should return a class constructor function",function() {
    expect(Foo).to.be.a("function");
  });

  it("should be able define a class",function() {
    obj = new Foo(1,2);
    expect(obj.constructor).to.eq(Foo);
    expect(obj.a).to.eql(1);
    expect(obj.b).to.eql(2);
```

```
    obj2 = new Foo(3,4);
    expect(obj2.a).to.eql(3);
    expect(obj2.b).to.eql(4);
  });

  it("should be able define a class without constructor",function() {
    klass = Class({});
    obj = new klass()
    expect(obj.constructor).to.eq(klass);
  });
});

describe("Implement Instance Methods",function() {
  var Foo = Class({
    initialize: function(a,b) {
      this.a = a;
      this.b = b;
    },

    getA: function() {
      return this.a;
    },

    getB: function() {
      return this.b;
    }
  });

  var foo = new Foo(1,2);

  it("should be able to define methods",function() {
    expect(foo.getA).to.be.a("function");
    expect(foo.getB).to.be.a("function");

    expect(foo.getA()).to.eq(1);
    expect(foo.getB()).to.eq(2);
  });

  it("should not define `initialize` as a method",function() {
    expect(foo.initialize).to.be.undefined;
  });
});

describe("Implement Class __super__",function() {
  var A = Class({
    a: function() {
      return 1;
    }
  });

  var B = Class({
    b: function() {
      return 2;
    }
  },A);

  it("should set the __super__ class property to the parent class",function()
{
    expect(B.__super__).to.eq(A);
  });

  it("should set Object as the default __super__ class",function() {
    expect(A.__super__).to.eq(Object);
  });
});

describe("Implement Methods Inheritance", function() {
  var A = Class({
    a: function() {
      return 1;
    }
```

```
  });

  var B = Class({
    b: function() {
      return 2;
    }
  },A);

  var b = new B();

  describe("b",function() {
    it("should be an instance of B",function() {
      expect(b.constructor).to.eq(B);
    });

    it("should be able to call method `a` through inheritance",function() {
      expect(b.a).to.be.a("function");
      expect(b.a()).to.be.eq(1);
    });

    it("should not have method `a` defined directly on the object",function()
{
      expect(b.hasOwnProperty("a")).to.be.false;
    });

    it("should not have method `a` defined directly on the prototype of
B",function() {
      expect(B.constructor.hasOwnProperty("a")).to.be.false;
    });
  });
});

describe("Implement Super call",function() {
  var A = Class({
    foo: function(a,b) {
      return [this.n,a,b];
    },

    bar: function() {
      return 1;
    },

    self: function() {
      return this;
    }
  });

  var B = Class({
    foo: function(a,b) {
      return this.super("foo",a*10,b*100);
    },

    bar: function() {
      return 20 + this.super("bar");
    },

    self: function() {
      return this.super("self");
    }
  },A);

  var C = Class({
    foo: function(a,b) {
      return this.super("foo",a*10,b*100);
    }
  },B);

  var b = new B();
  b.n = 1;

  it("should define the `super` method", function() {
    expect(b.super).to.be.a("function");
```

	`` }); it("should be able to call super method without arguments",function() { expect(b.bar()).to.equal(21); }); it("should call super method with the correct `this` context", function() { expect(b.self()).to.equal(b); }); it("should be able to call super method with multiple arguments", function() { expect(b.foo(2,3)).to.deep.equal([1,20,300]); }); }); describe("Implement Super's Super",function() { var A, B, C, c; beforeEach(function() { A = Class({ foo: function(a,b) { return [a,b]; } }); B = Class({ foo: function(a,b) { return this.super("foo",a*10,b*100); } },A); C = Class({ foo: function(a,b) { return this.super("foo",a*10,b*100); } },B); c = new C(); }); it("should be able to call super's super",function() { expect(c.foo(1,2)).to.deep.equal([100,20000]); }); }); ```
What is middleware?	Each request goes through layers of middleware, to be filtered (with filter) or responded (with provider). Nodejs Connect, Nodejs Express, Ruby Rack, etc ``` connect() .use(/* middleware */) .use(/* middleware */) .listen(3000); ```
Mini Harp Static Server 实现一个中间件，实现 static serve 解释： HarpJS 是一个静态文件服务器+编译工具，适合拿来做静态页面，比如技术文档和个人博客。它有这些功能： 转化 Markdown，Less，Coffee 文件为 HTML	**./test/mini-harp-spec.js** ``` var expect = require('chai').expect; var request = require('request'); var currentTime = require('../current-time'); var fs = require('fs'); describe('Implement A Current Time Middleware', function() { var createMiniHarp = require("..") , app = createMiniHarp() , port = 4000; ```

, CSS, JavaScript
制定 layout
在 data.json 声明页面参数

```
    app.use(currentTime).listen(port);

    it('GET /current-time should return time with 200', function(done) {
        request.get('http://localhost:' + port + '/current-time',
function(err, res, body) {
            expect(res.statusCode).to.eql(200);
            expect(new Date() - new Date(body)).to.be.below(50);

            done();
        });
    });

    it('GET /unknown/path should return 404', function(done) {
        request.get('http://localhost:' + port + '/unknown/path',
function(err, res, body) {
            expect(res.statusCode).to.eql(404);

            done();
        });
    });
});

describe('Serving Static Content', function() {
    var miniHarp = require('..');
    var root = __dirname + '/../';
    var app = miniHarp(root);
    var port = 4000;
    app.listen(port);

    it('GET /package.json should match the file content', function(done) {
        var filename = 'package.json';
        fs.readFile(root + filename, 'utf8',function(err, data) {
            if (err) throw err;

            request.get('http://localhost:' + port + '/' + filename,
function(err, res, body) {
                expect(body).to.eql(body);

                done();
            });
        });
    });

    it('GET /i-dont-exist.js should return 404', function(done) {
        request.get('http://localhost:' + port + '/i-dont-exist.js',
function(err, res, body) {
            expect(res.statusCode).to.eql(404);

            done();
        });
    });
});
```

./index.js

```
var serveStatic = require('serve-static');

var createMiniHarp = function(staticRootPath) {
    var connect = require('connect');

    var app = connect();

    staticRootPath = staticRootPath || __dirname;
    app.use(serveStatic(staticRootPath));

    return app;
};

module.exports = createMiniHarp;
```

./current-time.js

```
var currentTime = function(req, res, next) {
    if (req.url == '/current-time') {
        res.writeHead(200, {'Content-Type': 'text/plain'});
        res.end((new Date()).toISOString());
    }

    next();
};

module.exports = currentTime;
```

./bin/mini-harp.js

```
#!/usr/bin/env node

var mini_harp = require("../index.js");
var argv = require('minimist')(process.argv.slice(2));
var app = mini_harp(argv.dir);
console.log("Starting mini-harp on http://localhost:" + argv.port + " and
static serving " + argv.dir);
app.listen(argv.port);
```

./package.json

```
{
  "name": "mini-harp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "mocha test/mini-harp-spec.js"
  },
  "author": "",
  "license": "ISC",
  "bin": {
    "mini-harp": "bin/mini-harp.js"
  },
  "dependencies": {
    "connect": "^3.0.1",
    "minimist": "^1.1.1",
    "serve-static": "^1.0.3"
  },
  "devDependencies": {
    "request": "^2.58.0",
    "mocha": "^2.2.5",
    "chai": "^3.0.0"
  }
}
```

Chunked transfer encoding

```
data is sent in a series of "chunks". It uses the Transfer-Encoding HTTP
header in place of the Content-Length header

var http = require("http");

var server = http.createServer(function(req,res) {
  var i = 0;
  function tick() {
    i++;

    res.write(String(i)+"\n");
    setTimeout(tick,500);
  }

  tick();
});

server.listen(4000);

You can see the streaming output using curl:

$ curl -i --no-buffer localhost:4000
HTTP/1.1 200 OK
```

```
Date: Mon, 31 Mar 2014 15:15:39 GMT
Connection: keep-alive
Transfer-Encoding: chunked

1
2
3
4
5
6
7
8
9
^C


It is better to turn off chunked encoding unless you are dynamically
generating content, and you don't know the how much content there is.
```

Mini Harp Preprocessors

create middlewares that extend the static mini-harp application so it can:

render .jade templates as HTML.
render .coffeescript as JavaScript.
render .less as CSS.

Each of the preprocessor is in its own independent middleware.

Unit Tests:

git clone https://github.com/hayeah/fork2-mini-harp-verify.git verify

./lib/processor/less.js

```
var less = require('less');
var path = require('path');
var fs = require('fs');

function makeLess(root) {
    return function (req, res, next) {
        if (path.extname(req.url) != '.css') {
            return next();
        }

        var filePath = root + path.dirname(req.url) + path.basename(req.url,
'.css') + '.less';
        fs.readFile(filePath,
            { encoding: "utf8" },
            function (err, data) {
                if (err) {
                    return next();
                }

                less.render(data, function (err, css) {
                    if (err) {
                        return next();
                    }

                    res.setHeader('Content-Length', css.length); //Turning Off
Chunked Transfer
                    res.setHeader('Content-Type', 'text/css; charset=UTF-8');
                    res.end(css);
                });
            }
        );
    }
}

module.exports = makeLess;
```

./lib/processor/jade.js

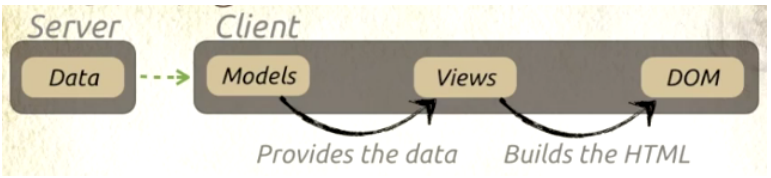
```
var jade = require('jade');
var path = require('path');
var fs = require('fs');

function makeJade(root) {
    return function(req, res, next) {
        if (path.extname(req.url) != '.html') {
            return next();
        }

        var filePath = root + path.dirname(req.url) + path.basename(req.url,
'.html') + '.jade';
        fs.readFile(filePath,
```

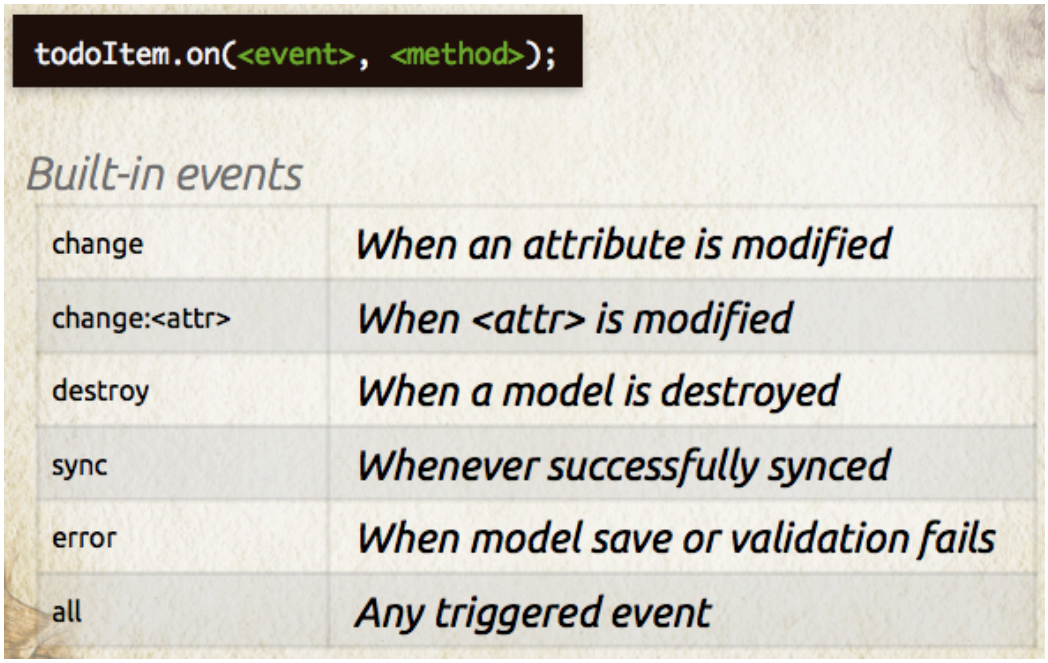
	`` { encoding: "utf8" }, function (err, data) { if (err) { return next(); } var html = jade.render(data); res.setHeader('Content-Length', html.length); res.setHeader('Content-Type', 'text/html; charset=UTF-8'); res.end(html); }); } } module.exports = makeJade; ./index.js var serveStatic = require('serve-static'); var makeJade = require('./lib/processor/jade'); var makeLess = require('./lib/processor/less'); var path = require('path'); var createMiniHarp = function(root) { var connect = require('connect'); var app = connect(); root = root || __dirname; app.use(function (req, res, next) { if (req.url == "/") { req.url = "/index.html"; } next(); }) .use(function (req, res, next) { var ext = path.extname(req.url); if (ext == '.jade' || ext == '.less') { res.statusCode = 404; return res.end(); } next(); }) .use(serveStatic(root)) .use(makeJade(root)) .use(makeLess(root)); return app; }; module.exports = createMiniHarp; ```
Explain connect middleware	http://code.tutsplus.com/tutorials/meet-the-connect-framework--net-31220
Explain ``` extend = function(child, parent) { for (var key in parent) { if (hasProp.call(parent, key)) child[key] = parent[key]; } function ctor() { this.constructor = child; } ctor.prototype = parent.prototype; child.prototype = new ctor(); child.__super__ = parent.prototype; return child; ```	``` child.prototype = Object.create(parent.prototype); child.prototype.constructor = child; ```

}	

Backbone.js

http://backbonejs.org/

Backbone.js key ideas on updating data for the DOM?	## Motivation Without Backbone.js  We need an object to maintain the data. (Getting Truth Out of the DOM)  ## Appointment App: General Steps to Render a view ``` var Appointment = Backbone.Model.extend({}); // Model class var appointment = new Appointment(); appointment.set('title', 'My knee hurts'); // 要拿 attribute 就是 get, 设是 set, 注意, 要用obj.get('attr'), 不能用obj.attr appointment.get('title'); var AppointmentView = Backbone.View.extend({ render: function() { // $(this.el).html(' ' + this.model.get('title') + ' '); // 下面是更好的办法 this.$el.html(' ' + this.model.get('title') + ' '); } }); var appointmentView = new AppointmentView({model: appointment}); appointmentView.render(); // 似乎先render, 就会有el $('#app').html(appointmentView.el); ```
How to set default value for a model?	``` var Appointment = Backbone.Model.extend({ defaults: { title: "Checkup", date: new Date() } }); ``` 但是上面的代码只 eval 了一次, 导致每个instance的date都是相同的, 改进如下: ``` var Appointment = Backbone.Model.extend({ defaults: function() { return { title: 'Checkup', date: new Date() } } }); ```
Fetch with URL?	``` var Appointment = Backbone.Model.extend({ urlRoot: "/appointments" }); var appointment = new Appointment({ id: 1}); appointment.fetch(); ```

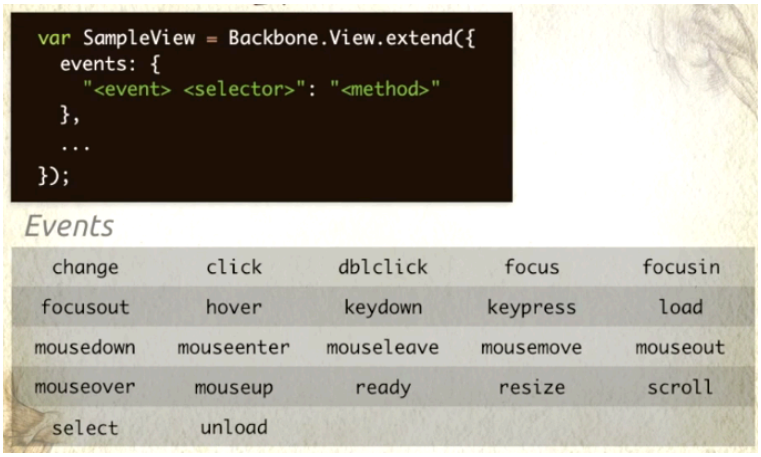
Syncing changes?	var appointment = new Appointment({id: 1}); appointment.set({ cancelled: true }); appointment.save();
Object listening on changes?	appointment 监听所有 attribute changes var appointment = new Appointment({id: 1}); appointment.on("change", function() { alert(); }); 如何只监听cancelled这一个attribute？ appointment.on('change:cancelled', function(){ alert("Hey Dr. Goodparts, your appointment has changed!"); }); 
object to JSON	appointment.toJSON();
view.el value?	By default, it is ' '. Then we can set it: var TodoView = Backbone.View.extend({ tagName: 'article', id: 'todo-view', className: 'todo' }); var todoView = new TodoView(); console.log(todoView.el) // => <article id="todo-view" class="todo"></article> If you want to use jQuery, do not use $('#todo-view').html(); use $(todoView.el).html(); or shortcut todoView.$el.html();
$ sign in Javascript?	By default, jQuery uses "$" as a shortcut for "jQuery".
template?	Default: underscore template lib 原来的代码是这样的: var AppointmentView = Backbone.View.extend({ render: function(){ this.$el.html('' + this.model.get('title') + ''); } }); 用 template 之后是这样的:

```
var AppointmentView = Backbone.View.extend({
  template: _.template('<span><%= title %></span>'),
  render: function(){
    var attributes = this.model.toJSON();
    this.$el.html(this.template(attributes));
  }
});
```

Other engines:

mustache.js, haml-js, eco

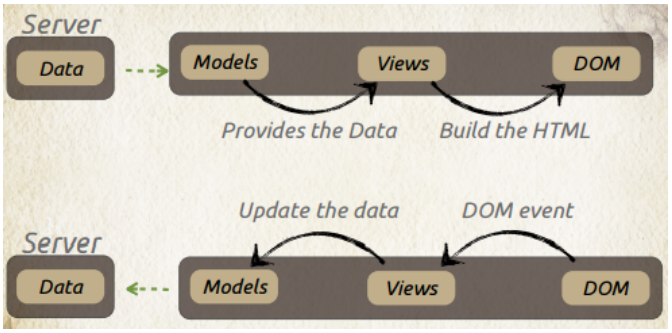
view event



双击 span 跳出 alert

```
var AppointmentView = Backbone.View.extend({
  template: _.template('<span><%= title %></span>'),

  events: { "dblclick span": "alertTitle" },
  alertTitle: function(){
    alert(this.model.get('title'));
  },

  render: function(){
    this.$el.html(this.template(this.model.toJSON()));
  }
});
```

How do the view events update the model?



click the <a href>x to set the appointment cancelled:

```
var AppointmentView = Backbone.View.extend({
  template: _.template('<span><%= title %></span> <a href="#">x</a>'),
  events:  { "click a": "cancel" },
  cancel: function(){
    this.model.set({cancelled: true});
  },
  render: function(){
    this.$el.html(this.template(this.model.toJSON()));
  }
});
```

上面代码的问题是，在 View 里面有大量代码操纵 Model，把 setter 放到 model 里：

	var AppointmentView = Backbone.View.extend({ template: _.template('<%= title %> x'), events: { "click a": "cancel" }, cancel: function(){ this.model.cancel(); }, render: function(){ this.$el.html(this.template(this.model.toJSON())); } }); var Appointment = Backbone.Model.extend({ cancel: function(){ this.set({cancelled: true}); this.save(); // 完事儿后记得向远端保存 } }); 可是，光向远端保存是不够的，我们有时候还需要更新 view var AppointmentView = Backbone.View.extend({ template: _.template('<span class="<% if(cancelled) print("cancelled") %>">' + '<%= title %>' + 'x'), events: { "click a": "cancel" }, initialize: function(){ this.model.on('change', this.render, this); }, // model有任何变化，就更新 view cancel: function(){ this.model.cancel(); }, render: function(){ this.$el.html(this.template(this.model.toJSON())); } }); 而 destroy 的情况则比较特殊 var AppointmentView = Backbone.View.extend({ template: _.template('<span class="<% if(cancelled) print("cancelled") %>">' + '<%= title %>' + 'x'), initialize: function(){ this.model.on('change', this.render, this); this.model.on('destroy', this.remove, this); }, events: { "click a": "cancel" }, cancel: function(){ this.model.cancel(); }, render: function(){ this.$el.html(this.template(this.model.toJSON())); }, remove: function(){ this.$el.remove(); } });
How to manipulate a Collection for a group of models?	## Define the class: var AppointmentList = Backbone.Collection.extend({ model: Appointment }); ## Init values:

	var appointments = new AppointmentList(); var json = [{title: 'Back pain'}, {title: 'Dry mouth'}, {title: 'Headache'}]; appointments.reset(json); ## Fetch from remote server: var AppointmentList = Backbone.Collection.extend({ model: Appointment, url: '/appointments' }); var appointments = new AppointmentList(); appointments.fetch(); ## Listen to events: appointments.on("reset", function() { alert(this.length); }); built-in events: add, remove, reset, change, change:<attr>, destroy, sync, error, all 有时候这个 function() 会有输入值: var appointments = new AppointmentList(); appointments.on('add', function(a) { console.log(a.get('title')); }); events triggered on a model in a collection will also be triggered on the collection. ## 那么如何避免fetch emits event呢？ appointments.fetch({silent: true}); ## iteration function on collections? http://backbonejs.org/#Collection-Underscore-Methods e.g. books.each(function(book) { book.publish(); });
Interaction between collection and view?	## create model view var appointments = new AppointmentList(); var AppointmentListView = Backbone.View.extend({}); var appointmentsView = new AppointmentListView ({ collection: appointments }); ## Render the collectoin into DOM. var AppointmentView = Backbone.View.extend({ template: _.template('<span class="<%= if(cancelled) print("cancelled") %>">' + '<%= title %>' + 'x'), render: function(){ this.$el.html(this.template(this.model.toJSON())); return this; } }); var AppointmentListView = Backbone.View.extend({ render: function(){ this.collection.forEach(this.addOne, this); }, addOne: function(appointment){ var appointmentView = new AppointmentView({ model: appointment }); this.$el.append(appointmentView.render().el);

```
  }
});

var appointmentsView = new AppointmentListView({collection: appointmentList});
appointmentsView.render();
$("#app").append(appointmentsView.el);

## adding to collection or resetting will update the view

var AppointmentListView = Backbone.View.extend({
  initialize: function(){
    this.collection.on('add', this.addOne, this);
    this.collection.on('reset', this.render, this);
  },
  render: function(){
    this.collection.forEach(this.addOne, this);
  },
  addOne: function(model){
    var appointmentView = new AppointmentView({model: model});
    appointmentView.render();
    this.$el.append(appointmentView.el);
  }
});


## removing a model in the collection (by customizing events)

var AppointmentList = Backbone.Collection.extend({
  model: Appointment,
  initialize: function(){
    this.on('remove', this.hideModel);
  },
  hideModel: function(model) {
    model.trigger('hide');
  }
});

var AppointmentView = Backbone.View.extend({
  initialize: function(){
        this.model.on('hide', this.remove, this); // view 里记得加this保证func
处理的是同样的instance
  },
  remove: function(){
    this.$el.remove();
  }
});
```

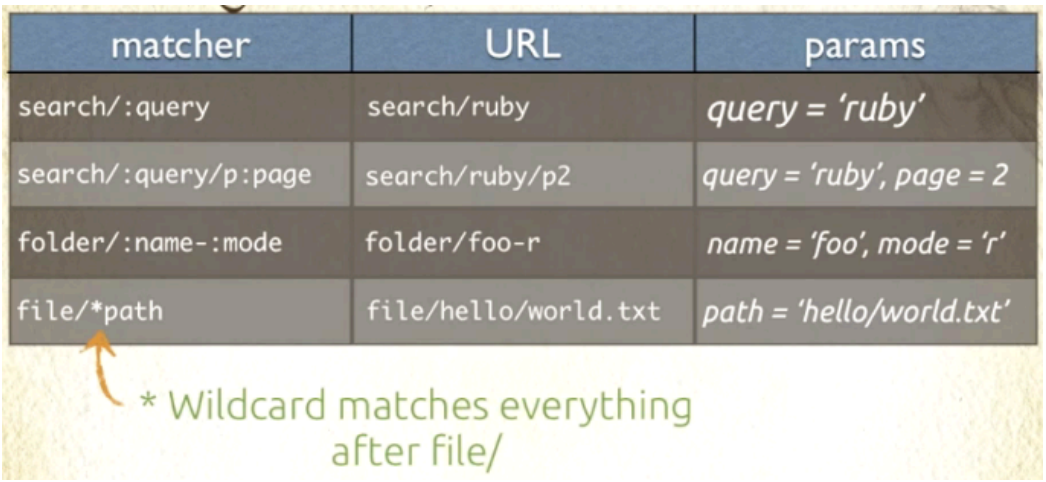
Routers & history:

how to have semantic urls with actions in a single page application?

hashbangs (#) vs HTML5 pushState (/)

http://www.ruanyifeng.com/blog/2011/03/url_hash.html

Router: to match URL with action




```
## Create a Router

var AppRouter = Backbone.Router.extend({
  routes: { "appointments/:id" : 'show' },
  show: function(id){
```

```
    console.log("heyo we're in show with id %d", id);
  }
});

Backbone.history.start({ pushState: true }); // 这样就是/appointments/1 而不是
#appointments/1 了

## Navigate and Trigger

var router = new AppRouter();
router.navigate("appointments/1", { trigger: true });

## Route and show the content

var AppRouter = Backbone.Router.extend({

  routes: {
    "appointments/:id": "show",
    "": "index"
  },

  initialize: function(options){
    this.appointmentList = options.appointmentList;
  },

  index: function(){
    var appointmentsView = new AppointmentListView({collection:
this.appointmentList});
        appointmentsView.render();
    $('#app').html(appointmentsView.el);
    this.appointmentList.fetch();
  },

  show: function(id){
    var appointment = new Appointment({id: id});
    var appointmentView = new AppointmentView({model: appointment});
    appointmentView.render();
    $('#app').html(appointmentView.el);
    appointment.fetch();
  }
});

## Wrap up and get the App organization

var App = new (Backbone.Router.extend({

  routes: { "appointments/:id": "show", "": "index" },

  initialize: function(){
    this.appointmentList = new AppointmentList();
  },

  start: function() {
    Backbone.history.start({ pushState: true });
  },

  index: function(){
    var appointmentsView = new AppointmentListView({collection:
this.appointmentList});
    appointmentsView.render();
    $('#app').html(appointmentsView.el);
    this.appointmentList.fetch();
  },

  show: function(id){
    var appointment = new Appointment({id: id});
    var appointmentView = new AppointmentView({model: appointment});
    appointmentView.render();
    $('#app').html(appointmentView.el);
    appointment.fetch();
  }
}));
```

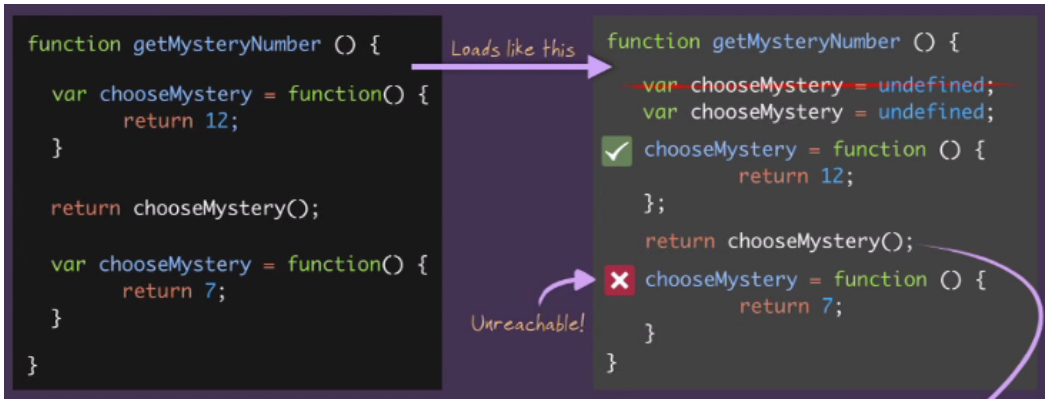
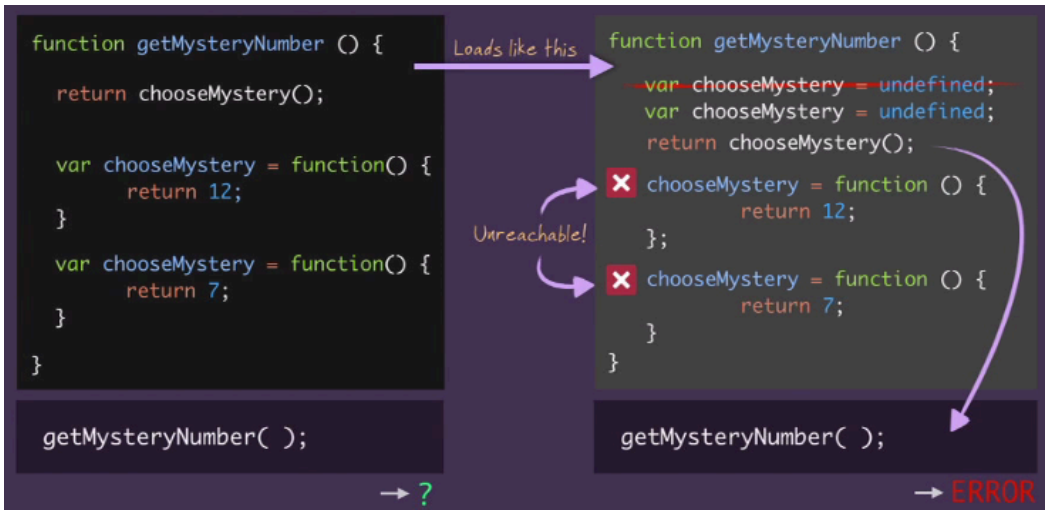
	$(function(){ App.start() }); https://www.codeschool.com/account/courses/anatomy-of-backbone-js

Javascript Basics

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Javascript 有 class 吗？	Ecmascript 6 开始才会有class
Javascript prompt (aka the console)	>
String comparisons	"blah" == "blah"
naming of variable	camel case
How to embed js script into html?	<script type="text/javascript" src="pathTo/train.js"></script> script 耗时间？有几种解决办法 1. 放到 body 接近结尾的地方 2. <script type="text/javascript" src="pathTo/train.js" async></script>
Built-in functions	alert() confirm() 返回用户选择的boolean值 prompt() 返回用户输入的值
typeof operator	typeof() method is useful in checking a variable's contents => "typeName"
function	function funcName() { }
array	var passengers = ["a", 1, "c"]; // 里面的东西类型可以不同 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
function expression?	// Eval only run into that exact line of code var diff = function diffOfSquares(a, b) { return a*a - b*b; }; diff(9, 5); // anonymous function var diff = function (a, b) { return a*a - b*b; }; console.log(diff); // => print the whole function content
passing function expressions as parameters	var numbers = [12, 4, 3, 9, 8, 6, 10, 1]; var results = numbers.map(function(e) { return e * 2; });
queue api	[0, 1, 2].shift()
What is closure and a simple example?	## Motivation

	Build different but similar functions. ## What 返回函数的函数，内部的这个函数会 capture (or close) 外部函数在那一刻的状态，内部的这个函数就叫做闭包。 ## Sample ``` function wrapValue(n) { var localVariable = n; return function() { return localVariable++; }; } var wrap1 = wrapValue(1); var wrap2 = wrapValue(2); console.log(wrap1()); // → 1 console.log(wrap2()); // → 2 console.log(wrap1()); // → 2 console.log(wrap2()); // → 3 ``` http://eloquentjavascript.net/03_functions.html#h_hOd+yVxaku
Closure's pitfalls [Pitfalls]	``` function alertNumberInArray(number, array) { var alertFunc; for (var i = 0; i < array.length; i++) { if (array[i] == number) { alertFunc = function() { alert(i); }; } } return alertFunc; } var array = [0, 1, 2, 3, 4, 5, 6]; alertNumberInArray(3, array)(); // => 7 (expected to be 3!) 因为这个闭包里面使用的 i 是闭包被outer func返回那一刻最后的状态 ``` 有两种解决方法： 1. 遇到期望的结果立即返回 ``` if (array[i] == number) { return function() { alert(i); }; } ``` 2. 把 outer func 里的状态放到 inner func 里面去，不 close 那些会导致问题的状态
[Anonymous closure] and its immediate invocation?	``` (function() { // write code in a fresh environment. })(); ``` js 只有函数才能打开一个独立的作用域，定义一个匿名函数并立即执行相当于开辟了一个块级的作用域。用匿名函数主要就是为了不想让函数内部的局部变量污染全局空间。 非匿名函数 funcName 会引入一个全局变量 funcName，js没有 access modifier，而 anonymous closure 能起到这作用 ``` var ARMORY = (function() { // namespace object, aka module // private fields and funcs var weaponList = []; var armorList = []; var removeWeapon = function() {}; ```

	`var replaceWeapon = function() {};` `var removeArmor = function() {};` `var replaceArmor = function() {};` `return {` `// public functions` `makeWeaponRequest: function() {},` `makeArmorRequest: function() {}` `};` `})();`
hoistering	Hoisting is JavaScript's default behavior of moving declarations to the top. `function getMysteryNumber() {` `function getNumber() {` `return 99;` `}` `return getNumber();` `function getNumber() {` `return 0;` `}` `}` `getMysteryNumber(); // => 0 (expected to be 99)` However, function expressions are never hoisted! They are treated as assignments.   http://www.w3schools.com/js/js_hoisting.asp
Javascript object?	JSON `myBox = {};` `myBox.height = 6;` `myBox.material = "cardboard";` `// 访问 property 的两种方式`

	console.log(myBox.height); // => 6 console.log(myBox["height"]); // => 6 myBox["# of stops"] = 2; console.log(myBox["# of stops"]); // => 2 myBox.contents = ["a", "b"]; // delete keyword!! delete myBox.contents; // => will always return true
Enumeration with for-in loop	myBox = { name: "Tian's box", size: 100, addOneBook: function(book) { this.books.push(book); }, books: [] }; myBox.addOneBook("Bible"); for (var key in myBox) { console.log(myBox[key]); }
prototype, constructor, and inheritance	## Motivation Inheritance ## Built-in examples in prototype chains object prototype -> {} -> array prototype -> [] -> string prototype -> "" -> number prototype -> 3 -> function prototype -> function() {} ## Add inheritable properties to prototypes String.prototype.countAll = function(letter) { var count = 0; for (var i = 0; i < this.length; i++) { if (this.charAt(i).toUpperCase() == letter.toUpperCase()) { count++; } } return count; }; "Tian Pan".countAll("a"); // => 2 ## Constructor and prototype function Shoe(size, color, gender, style) { this.size = size; this.color = color; this.gender = gender; this.style = style; this.putOn = function() { alert("put on " + this.style); }; this.takeOff = function() { alert("take off " + this.style); }; } var beachShoe = new Shoe(10, "blue", "women", "flipflop"); beachShoe.putOn(); But there is no need to instantiate putOn and takeOff every time, and the Shoe constructor can share funcs via prototype: [Best Practice] function Shoe(size, color, gender, style) { this.size = size; this.color = color; this.gender = gender;

	this.style = style; } Shoe.prototype.putOn = function() { alert("put on " + this.style); }; Shoe.prototype.takeOff = function() { alert("take off " + this.style); }; var beachShoe = new Shoe(10, "blue", "women", "flipflop"); beachShoe.putOn(); 还有一种给 prototype 赋值的做法: Shoe.prototype = { putOn: function() { alert("put on " + this.style); }, takeOff: function() { alert("take off " + this.style); } }; 注意 using a property approach to access indices will also add in all methods that have been added to the prototype: for (var key in beachShoe) { console.log(key); } // => size // color // gender // style // putOn // takeOff ## Override beachShoe.toString(); // => "[object Object]" beachShoe.constructor.prototype.toString = function() { return "this is a " + this.color + " shoe with size of " + this.size; }; beachShoe.toString(); // => "this is a blue shoe with size of 10" 上面的 beachShoe.constructor.prototype 等同于 Shoe.prototype, beachShoe.__prototype__
Build objects using Object.create()	var shoe = { size: 6, gender: "women", construction: "slipper" }; var magicShoe = Object.create(shoe); Object.prototype.isPrototype(shoe); // => true Object.prototype.isPrototype(magicShoe); // => true
Assignments with logical operators	var armory = { addSword: function(sword) { this.swords = this.swords || []; this.swords.push(sword); } }; undefined, 0, "" 都相当于 false, 如果都是 false 的话返回最后的一个值, 不管最后是不是相当于 false 的
variable scope pitfalls [Pitfalls]	for (var i = 0; i < 10; i++) { console.log(i); } console.log(i); => 10 (expected to be undefined) 注意这个比较特殊, 然后是 local variable 一定要加 var
Adding individual DOM elements ain't always speedy. Each new addition to DOM causes document reflow, which can really hinge the UI.	感觉这里的 var declaration 是奇怪的优化。。。Javascript 性能就差到需要这些小动作吗？ var list = document.getElementById("kotwList"), kotw = ["Jenna", "Neric", "Darkin"], fragment = document.createDocumentFragment(), element;

Use var fragment = document.createDocumentFragment(); Just like a builder.	for (var i = 0, x = kotw.length; i < x; i++) { element = document.createElement("li"); element.appendChild(document.createTextNode(kotw[i]); fragment.appendChild(element); } list.appendChild(fragment);
How to test the speed of your code?	console.time console.time("timeLabel"); // ... expressions console.timeEnd("timeLabel"); Date (more accurate) var start = +new Date(); // 等同于 var start = new Number(new Date); var end = +new Date(); var elapsedTime = end - start; function SpeedTest(testImplement, testParams, repetitions) { this.testImplement = testImplement; this.testParams = testParams; this.repetitions = repetitions || 10000; this.average = 0; } SpeedTest.prototype = { startTest: function() { var start, end, sum = 0; for (var i = 0, x = this.repetitions; i < x; i++) { start = +new Date(); this.testImplement(this.testParams); end = +new Date(); sum += end - start; } this.average = sum / this.repetitions; return console.log("avg time: " + this.average); } }
Comparison [Pitfalls]	'4' == 4; // => true '4' === 4; // => false true == 1; // => true true === 1; // => false false == 0; // => true false === 0; // => false "\n \n \t" == 0; // => true "\n \n \t" === 0; // => false === equal value and equal type Use instanceof operator to identify objects
Exception handling	try { throw new ReferenceError(); throw new TypeError(); } catch (e) { if (e instanceof ReferenceError) { } if (e instanceof TypeError) { } // ... } finally { }
avoid `with` keyword	本意是 helper coders avoid typing deep access repeatedly.

[Pitfalls]	var a, x, y; var r = 10; with (Math) { a = PI * r * r; x = r * cos(PI); y = r * sin(PI / 2); } 上面 Math 的方法直接用 但是会导致 confusion, 所以不要用, 直接用变量就好了
use little `eval` as possible [Pitfalls]	## Treating string as runnable code is expensive. (function evalPrint(owner, toy) { eval("console.log(owner + ''s' + 'iPhone')"); })("Tian", "iPhone"); ## Using eval to parse JSON is unsafe and unefficient. Use JSON.parse()
Be careful with numbers [Pitfalls]	Inacuate double needs to be fixed var num = 0.1 + 0.2; console.log(parseFloat(num.toFixed(2))); parseInt(string, radix); typeof NaN; // => "number" console.log(NaN === NaN); // => false isNaN("42"); // => false ## Do double check! function isNumber(data) { return (typeof data === "number" && !isNaN(data)); } isNumber(640); // => true isNumber("640"); // => false isNumber(NaN); // => false
Namespace object	An application that has multiple files which contain variables with the same name will experience a loss of data in the event of an overwrite, which erases old data in favor of new (and perhaps wrong) data. This occurs specifically when these variables are declared globally in multiple files. This error may not be found until runtime. 原来是这样 var stalactites = 4235; var stalagmites = 3924; var bats = 345; var treasureChests = 3; var openChest = function(){ treasureChests--; alert("DA DADADA DAAAAAAA!"); }; 要保护这些变量, 用object 包起来 var CAVESOFCLARITY = { stalactites: 4235, stalagmites: 3924, bats: 345, SECRET: { treasureChests: 3, openChest: function() { this.treasureChests--; alert("DA DADADA DAAAAAAA!"); }

	} }; 至于 access modifier, goto [Anonymous closure]
Avoid using global var in a module (2 reasons). Instead, use imported global variables	1. entire length of scope chain is checked. inefficient. 2. confusing 正确的用法: var wartime = true; var ARMORY = (function(war) { // namespace object, aka module // private fields and funcs var weaponList = []; var armorList = []; var removeWeapon = function() {}; var replaceWeapon = function() {}; var removeArmor = function() {}; var replaceArmor = function() {}; return { // public functions makeWeaponRequest: function() { if (war) {}; }, makeArmorRequest: function() {} }; })(wartime);
Augmentation	## Motivation It provides extra properties for existing modules. In a separate file, we keep a function which will add values or funcs to the existing module. ## Sample var AUGMENTEDARMORY = (function(oldNS) { oldNS.newProperty = "Blah"; return oldNS; })(ARMORY); Note: previous private data will not be accessible to the new properties.
Inheritance	// define the Person Class function Person(name) { this.name = name; } console.log(Person == Person.prototype.constructor); // true Person.prototype.copy = function() { return new this.constructor(this.name); }; // define the Student class function Student(name) { Person.call(this, name); } // inherit Person Student.prototype = Object.create(Person.prototype); // copy 一个相同的 prototype 相当于 new Person Student.prototype.constructor = Student; // 但是 constructor 得 reset var student1 = new Student("trinth"); console.log(student1.copy() instanceof Student); // => true
Multiple inheritance and mixin	function ClassA () {

	} ClassA.prototype.a = function() { console.log("ClassA.a()"); }; function ClassB () { } ClassB.prototype.b = function() { console.log("ClassB.b()"); }; function SubClass() { ClassA.call(this); ClassB.call(this); } SubClass.prototype = Object.create(ClassA.prototype); // inherit mixin(SubClass.prototype, ClassB.prototype); // mixin var sub = new SubClass(); sub.a(); sub.b(); The mixin function would copy the functions from the superclass prototype to the subclass prototype, the mixin function needs to be supplied by the user. An example of a mixin like function would be jQuery.extend(). underscore http://underscorejs.org/#mixin 库
func.constructor vs func.prototype.constructor	obj.constructor: A method called at the moment an object is instantiated. It usually has the same name as the class containing it. var obj = new Class() obj.constructor == Class obj.prototype.constructor == undefined func.prototype.constructor: Returns a reference to the Object function that created the instance's prototype. Note that the value of this property is a reference to the function itself, not a string containing the function's name. The value is only read-only for primitive values such as 1, true and "test". func.prototype.constructor == func func.constructor 不知道是什么

jQuery

Motivation	js lib to find elements, change HTML, listen on user actions, animate content, talk to network. Write less and do more. ## Find <!DOCTYPE html> <html> <head> <title>jQuery Adventures</title> </head> <body> Where do you want to go? --> jQuery("h1"); or $("h1"); Plan your next adventure. </body> </html>

<table>
<tr><td></td><td>

```
## Modify

$("h1").text("Where to?");

but make sure it gets exec after the DOM is loaded

$(document).ready(function() {
    // code
    $("h1").text("Where to?");
});
```

</td></tr>
<tr><td>How to use jQuery?</td><td>

```
<script src="jquery.min.js"></script>
```

</td></tr>
<tr><td>Find</td><td>

```
## Selector

Same as CSS selector
```

CSS2 Selectors	
*	All elements
div	
div *	All elements within
div span	within
div, span	and
div > span	with parent
div + span	preceded by
.class	Elements of class "class"
div.class	of class "class"
#itemid	Element with id "itemid"
div#itemid	with id "itemid"
a[attr]	<a> with attribute "attr"
a[attr='x']	<a> when "attr" is "x"
a[class~='x']	<a> when class is a list containing 'x'
a[lang|='en']	<a> when lang begins "en"

```
       #destinations li:first
       #destinations li:last
       #destinations li:even
       #destinations li:odd

## Traversing (faster)

$("#destinations li")

等同于

$("#destinations").find("li");

后者更快

$("li").first();
$("li").first().next();
$("li").first().next().prev();
$("li").first().parent();

children()
```

</td></tr>
<tr><td>Manipulate DOM</td><td>

```
## Create DOM
```

</td></tr>
</table>

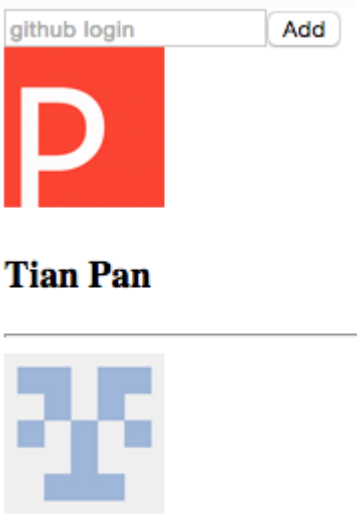
	`` var price = $(' From $399.99 '); ## Put before $(document).ready(function() { var price = $(' From $399.99 '); $('.vacation').before(price); }); ## Put after $(document).ready(function() { var price = $(' From $399.99 '); $('.vacation').after(price); }); ## Put to the begining of the inner HTML of .vacation $(document).ready(function() { var price = $(' From $399.99 '); $('.vacation').prepend(price); }); ## Put to the end of the inner HTML of .vacation $(document).ready(function() { var price = $(' From $399.99 '); $('.vacation').append(price); // <==> price.appendTo($('.vacation')); }); ## Remove $(document).ready(function() { var price = $(' From $399.99 '); $('.vacation').append(price); $('button').remove(); }); ```
Acting on Interaction	``` ## Listen on events Click, append, and remove $(document).ready(function() { $('button').on('click', function() { var price = $(' From $399.99 '); $('.vacation').append(price); $('button').remove(); }); }); ```

React.js

What is React.js	``` React is the V in MVC simple, performant, server-side friendly, self-contained components Playground with Plunker ## Component this.state. 会变 this.props. 不会变 ## Format JSX and the virtual DOM ```

	## Render React.createClass({render: function() {}}) React.render({<>, mountNode }) ## Innovation JSX: markup in js Virtual DOM: render diff only Isomorphic rendering: render at client and server unidirectional flows ## Helpful tools gulp task runner jest tests browserify: bundler using npm packages on the browser side
Implement a component for clicking to increase 1	**./index.html** <!DOCTYPE html> <html> <head> <link rel="stylesheet" href="style.css"> </head> <body> <script src="https://fb.me/react-0.13.3.js"></script> <script src="script.js"></script> </body> </html> **./script.jsx** var Button = React.createClass({ getInitialState: function() { return { counter: 0 }; }, handleClick: function() { this.setState({ counter: this.state.counter + 1 }); }, render: function() { return (<button onClick={this.handleClick}>{this.state.counter}</button>) } }); React.render(<Button />, document.getElementById("root"));
Implement reusable components for clicking to increase 1, 5, 10, 100	**./index.html** <!DOCTYPE html> <html> <head> <link rel="stylesheet" href="style.css"> </head> <body> <script src="https://fb.me/react-0.13.3.js"></script> <script src="script.js"></script> </body>

	</html> **./script.jsx** var Button = React.createClass({ localHandleClick: function () { this.props.localHandleClick(this.props.increment); }, render: function () { return (<button onClick={this.localHandleClick}>+{this.props.increment}</button>) } }); var Result = React.createClass({ render: function () { return ({this.props.localCounter}) } }); var Main = React.createClass({ getInitialState: function () { return {counter: 0}; }, handleClick: function (increment) { this.setState({counter: this.state.counter + increment}); }, render: function () { return (<Button localHandleClick={this.handleClick} increment={1}/> <Button localHandleClick={this.handleClick} increment={5}/> <Button localHandleClick={this.handleClick} increment={10}/> <Button localHandleClick={this.handleClick} increment={100}/> <Result localCounter={this.state.counter}/>) } }); React.render(<Main />, document.getElementById("root"));
Working with Data Build a Github Card Component	**./index.html** <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title></title> </head> <body> <script src="//code.jquery.com/jquery-1.11.3.min.js"></script> <script src="https://fb.me/react-0.13.3.js"></script> <script src="https://fb.me/JSXTransformer-0.13.3.js"></script> <script type="text/jsx" src="script.jsx"></script> </body> </html> **./script.jsx** var Card = React.createClass({

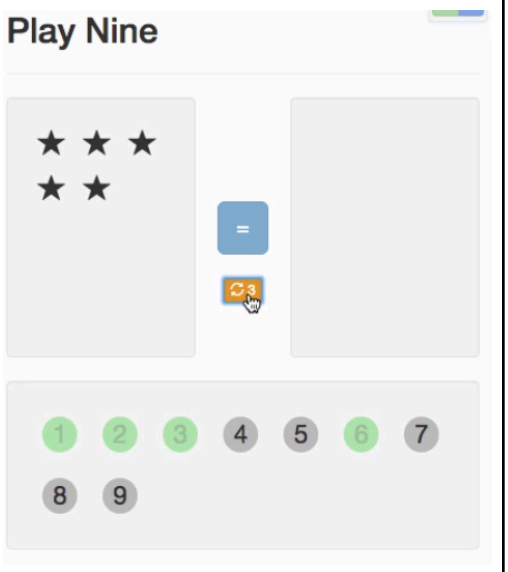



```
  getInitialState: function () {
    return {};
  },
  componentDidMount: function () {
    var component = this;
    $.get("https://api.github.com/users/" + this.props.login, function (data)
{
      component.setState(data);
    });
  },
  render: function () {
    return (
      <div>
        <img src={this.state.avatar_url} width="80"/>

        <h3>{this.state.name}</h3>
        <hr/>
      </div>
    )
  }
});

var Form = React.createClass({
  handleSubmit: function (e) {
    e.preventDefault();
    var loginInput = React.findDOMNodecc(this.refs.login);
    this.props.addCard(loginInput.value);
    loginInput.value = '';
  },
  render: function () {
    return (
      <form onSubmit={this.handleSubmit}>
        <input placeholder="github login" ref="login"/>
        <button>Add</button>
      </form>
    );
  }
});

var Main = React.createClass({
  getInitialState: function () {
    return {logins: []};
  },
  addCard: function (loginToAdd) {
    this.setState({
      logins: this.state.logins.concat(loginToAdd)
    })
  },
  render: function () {
    var cards = this.state.logins.map(function (login) {
      return (<Card login={login}/>);
    });
    return (
      <div>
        <h>Github</h>
        <Form addCard={this.addCard}/>
        {cards}
      </div>
    )
  }
});

React.render(<Main />, document.getElementById("root"));
```

JSX file import

```
    <script src="https://fb.me/react-0.13.3.js"></script>
    <script src="https://fb.me/JSXTransformer-0.13.3.js"></script>
    <script type="text/jsx" src="script.jsx"></script>

./script.jsx
```

Building Game Interface

Redux.js

Try with redux form	