Patryk Jar
Meet.js, Gdańsk 11 marca 2013 r.

MODULARNY JAVASCRIPT

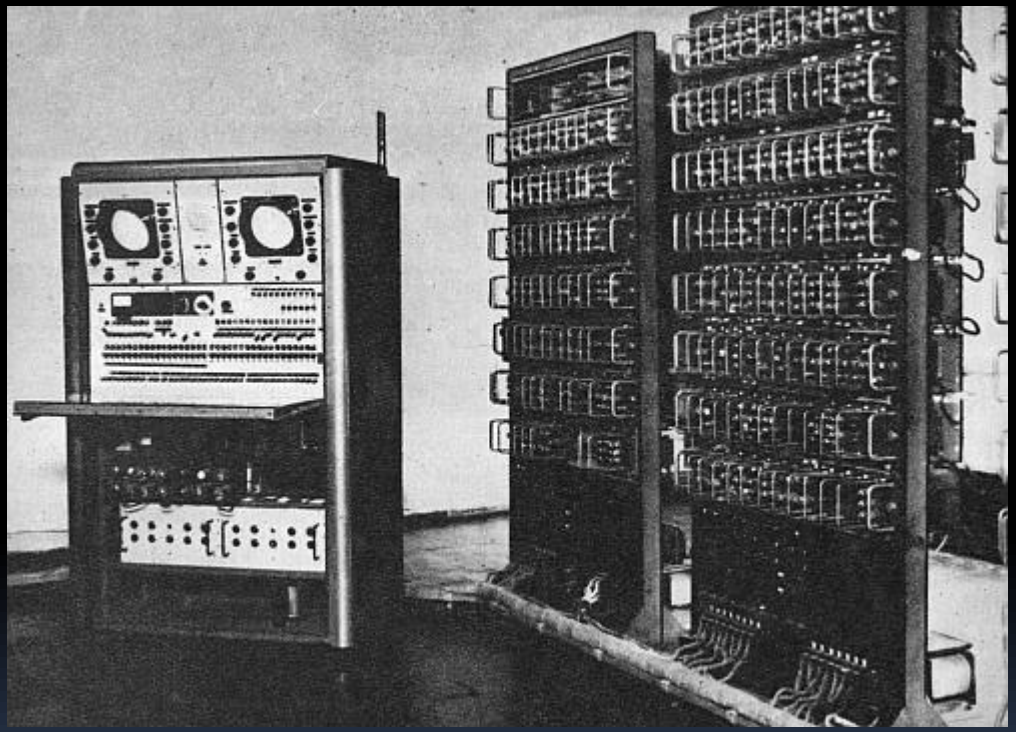
O mnie

- Patryk ‘yarpo’ Jar
- Programista JavaScript (nor-sta.eu)
- yarpo.pl

Agenda

- Chaos
- Obiekty
- Biblioteki
- AMD
- Podsumowanie
- Pytania

Dawno, dawno temu…



źródło grafiki: **gadzetomania.pl**

Początki JavaScript

- Brak „dobrych praktyk” dla modułów
- Traktowany jako **banalny** dodatek do HTML
- Słabe wsparcie ze strony przeglądarek
  - Rozbieżności
  - Wolne silniki
- Ni to „Java”, ni to „Script”
  - D. Crockford: „ The World's Most Misunderstood Programming Language”

Chaos

```
function dodaj(a, b) { return a+b; }
function odejmij (a, b) { return a-b; }
```

```
<script src='obliczenia.js'></script>
```

Obiekty

```
1. function Obliczenia() {
2.     this.dodaj = function(a, b) { return a+b; };
3.     this.odejmij = function(a, b) { return a-b; };
4. };

5. var obl = new Obliczenia();
6. console.log(obl.dodaj(2,2)); // = 4
```

Zobacz: **tnij.org/obiekty-js**

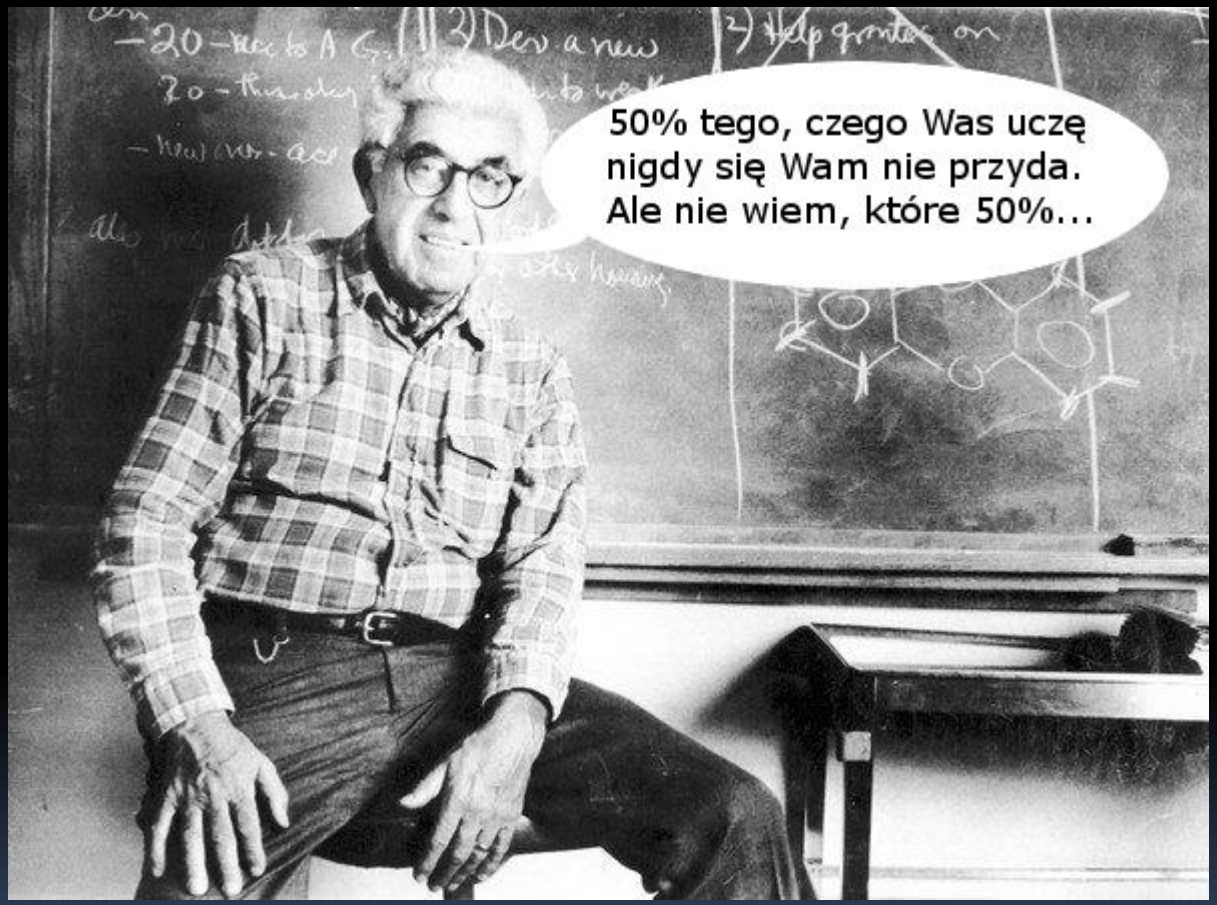
Problemy integracji kodu

- Różne źródła – różne koncepcje programistyczne
- Konflikty nazw zmiennych globalnych
  - Global namespace pollution

Wzorzec modułu

```
1.  var app = {
2.      utils : {
3.          CONST: 42 // to nadal tylko zmienna
4.      },
5.      ajax : {
6.          get : function(id) { ... },
7.          add : function() { ... }
8.      }
9.  };
10. app.ajax.get(app.utils.CONST);
```

Biblioteki


50% tego, czego Was uczę
nigdy się Wam nie przyda.
Ale nie wiem, które 50%...

jQuery: $('future')

```
$('div').hide().parent().css('color', 'red');
jQuery.ajax({ ... }); // $ === jQuery
```

\+ Nie zanieczyszczamy przestrzeni globalnej

\+ Tworzymy wtyczki (moduły)

\- Nadal globalnie załączamy pliki z kodem

\- „$ namespace pollution" (*jQuery.noConflict()*)

\- „Programista jQuery" często nie zna JavaScript

Dotychczasowy kod HTML

```
<html>
<head>
   <title>Robisz to źle</title>
   <script src="jquery/1.7.1/jquery.min.js"></script>
   <script src="pluginA.jquery.js"></script>
   <script src="pluginB.jquery.js"></script>
   <script src="logic.js"></script>
   <script src="dojo.js"></script>
   <link rel="stylesheet" type="text/css" href="all.css">
   <script src="stillNotEnoughJSFiles.js"></script>
   <link rel="stylesheet" type="text/css" href="style.css">
...
```

Trochę lepszy kod

```
<html>
<head>
   <title>Robisz to trochę lepiej;)</title>
   <script src="all.compressed.js"></script>
   <link rel="stylesheet" type="text/css"
   href="all-styles.merged.css">
```

dojo.declare: ¿¡klasy w JS?!

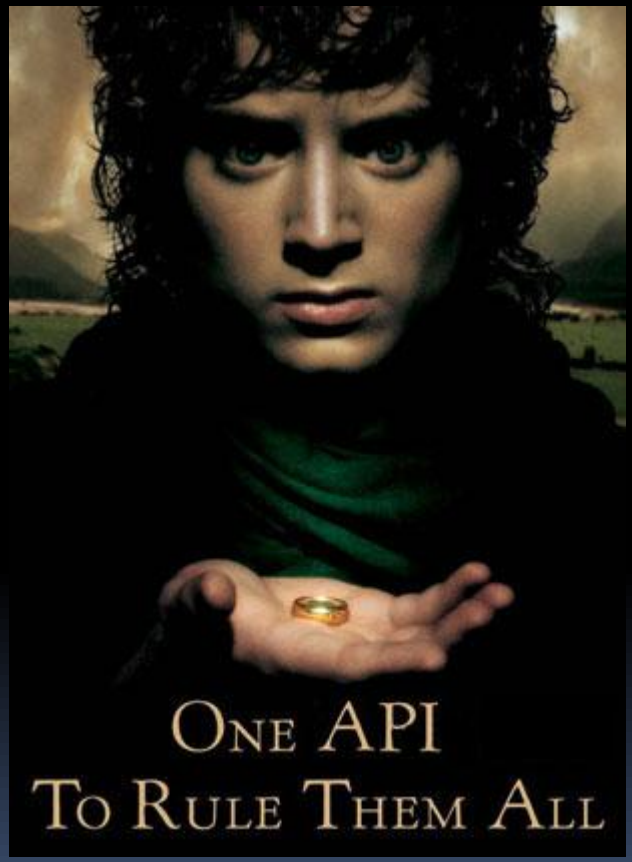
```
1.  // plik `yarpo/MyFirstClass.js'
2.  dojo.provide("yarpo.MyFirstClass");
3.  dojo.require("yarpo.Xyz"); // pobiera yarpo/Xyz.js
4.  dojo.require("yarpo.Abc"); // pobiera yarpo/Abc.js
5.  dojo.declare("yarpo.MyFirstClass", [yarpo.Xyz], {
6.      constructor : function(a, b) {
7.          console.log("działam", a, b);
8.          var abc = new yarpo.Abc();
9.          //this.inherited(arguments);
10.     }
11. });
12. var obj = new yarpo.MyFirstClass(1, 2); // „działam 1 2"
```

dojo.declare

+ Czytelne i podobne do znanych rozwiązań
+ Prawie samowystarczalne (CSS)
+ Przyjemne i kompletne

- Uzależnienie od Dojo Toolkit

Użyjmy niezależnego API!



Nowe sposoby na moduły

- **AMD – Asynchronous Module Definition**
  - Dojo 1.7+
  - jQuery 1.7+
  - Mootools 2.0+
  - Node.js
- CommonJS
  - Node.js

Zobacz: **RequireJS.org**

AMD – przykład: quick start

```
1.  // plik 'moduly/Obliczenia.js'
2.  define('moduly/Obliczenia', function() {
3.    return function() {
4.      this.dodaj = function(a, b) { return a + b; };
5.      this.odejmij = function(a, b) { return a - b; };
6.    };
7.  });

8.  require(['moduly/Obliczenia'], function(Obliczenia) {
9.    var obliczenia = new Obliczenia();
10.   console.log(obliczenia.dodaj(2,2)); // 4
11. });
```

AMD – przykład 2: zależności

```
1. define('moduly/Obliczenia', ['mat/dodaj', 'mat/odejmij'],
2.    function(dodaj, odejmij) {
3.       return function() {
4.             this.dodaj = function(a, b) { return dodaj(a, b); }
5.             this.odejmij = function(a,b){ return odejmij(a, b); }
6.       };
7. });
```

```
1.  // plik 'mat/dodaj.js'
2.  define('mat/dodaj', function() {
3.     return function(a, b) { return a+b; };
4.  });
```

AMD – wtyczki

- Szablony widgetów, pliki txt
  - dojo/text!sciezka/do/pliku
- Pliki językowe
  - dojo/i18n!nls/nazwaModuluI18n
  - Język ustalony per aplikacja
- Style CSS
  - xstyle!./sciekza/do/pliku.css
  - github.com/kriszyp/xstyle

Zobacz: **requirejs.org/docs/plugins.html**

AMD – wady: kolejność

```
1.  define('tct/DetailsPanel', [
2.      "dojo/_base/declare",
3.      "dijit/_TemplatedMixin",
4.     "dojo/i18n!./nls/DetailsPanel"],
5.  function(
6.     declare,
7.      template,
8.      _TemplatedMixin)  {
9.         return declare("tct.DetailsPanel", [_TemplatedMixin],
10.        function() { ... });
```

AMD – wady: nazwy

```
1.  define('tct/DetailsPanel', [
2.      "dojo/_base/declare",
3.      "dijit/_TemplatedMixin",
4.      "dojo/i18n!./nls/DetailsPanel",
5.      "dojo/text!./templates/detailsPanel.html"],
6.  function(
7.      deklarowanie,
8.      templejt,
9.      a,
10.     template) {
11.       return deklarowanie("tct.DetailsPanel", [templejt], fun... });
```

AMD – zalety

- Lokalny (nie brudzimy globalnej przestrzeni)
- Samowystarczalny (sam definiuje, jakie moduły muszą być pobrane)
  - Szablony widgetów, pliki txt (np. `dojo/text!`)
  - Pliki językowe (np. `dojo/i18n!`)
  - Pliki stylów CSS! (github.com/kriszyp/xstyle)
- Uniwersalny (define.amd == *true*)
- Dobrze się kompresuje (np. Shrinksafe)

AMD – zalety: kompresja

```
define("dojo/cookie",["./_base/kernel","./regexp
"],function(_1,_2){_1.cookie=function(_3,_4,_
5){var
c=document.cookie,_6;if(arguments.length=
=1){var _7=c.match(new RegExp("(?:^|;
)"+_2.escapeString(_3)+"=([^;]*)"));_6=_7?de
codeURIComponent(_7[1]):undefined;}else{_
5=_5||{};var _8=_5.expires;if(typeof
_8=="number"){
```

...

Zobacz: **shrinksafe.dojotoolkit.org**

Podsumowanie

- Aktualny kierunek: uniwersalne API modułów
- Moduł vs. Biblioteka
- Przyszłość: ECMAScript Harmony
- Alternatywa: *Asemblaryzacja* JavaScript
  - Google Web Toolkit
  - Cappucino (cappuccino-project.org)
  - CoffeScript (coffeescript.org)
  - TypeScript (typescriptlang.org)
  - Poniekąd jQuery

Pytania

Dziękuję za uwagę

- yarpo.pl
- Jar.Patryk@gmail.com