Bogate Interfejsy Użytkownika System.js

Łukasz Rybka · Gdańsk 2015

Problemy dużych aplikacji webowych

Problemy dużych aplikacji webowych

Duże strony stają się aplikacjami

Problemy dużych aplikacji webowych

Duże strony stają się aplikacjami

Złożoność kodu rośnie wraz z rozrostem aplikacji

Problemy dużych aplikacji webowych

Duże strony stają się aplikacjami

Złożoność kodu rośnie wraz z rozrostem aplikacji

"Assembly problem"

Problemy dużych aplikacji webowych

Duże strony stają się aplikacjami

Złożoność kodu rośnie wraz z rozrostem aplikacji

"Assembly problem"

Potrzeba mniejszych, "dyskretnych" modułów

Problemy dużych aplikacji webowych

Duże strony stają się aplikacjami

Złożoność kodu rośnie wraz z rozrostem aplikacji

"Assembly problem"

Potrzeba mniejszych, "dyskretnych" modułów

Optymalizowany kod dostępny w zaledwie kilku zapytaniach HTTP

Własne rozwiązanie

```
var Zoo = (function() {
  var getBarkStyle = function(isHowler) {
    return isHowler? 'wooooooow!': 'woof, woof!';
  };

  var Dog = function(name, breed) {
    this.bark = function() {
      return name + ': ' + getBarkStyle(breed === 'husky');
    };
  };

  var Wolf = function(name) {
    this.bark = function() {
      return name + ': ' + getBarkStyle(true);
    };
  };

  return {
    Dog: Dog,
    Wolf: Wolf
  };
})();
```

Własne rozwiązanie

```
var myDog = new Zoo.Dog('Sherlock', 'beagle');
console.log(myDog.bark()); // Sherlock: woof, woof!

var myWolf = new Zoo.Wolf('Werewolf');
console.log(myWolf.bark()); // Werewolf: wooooooow!
```

Co jest nie tak z tym rozwiązaniem?

Co jest nie tak z tym rozwiązaniem?

Wrażliwość na zmiany - każdy moduł może być zmodyfikowany (dostęp globalny)

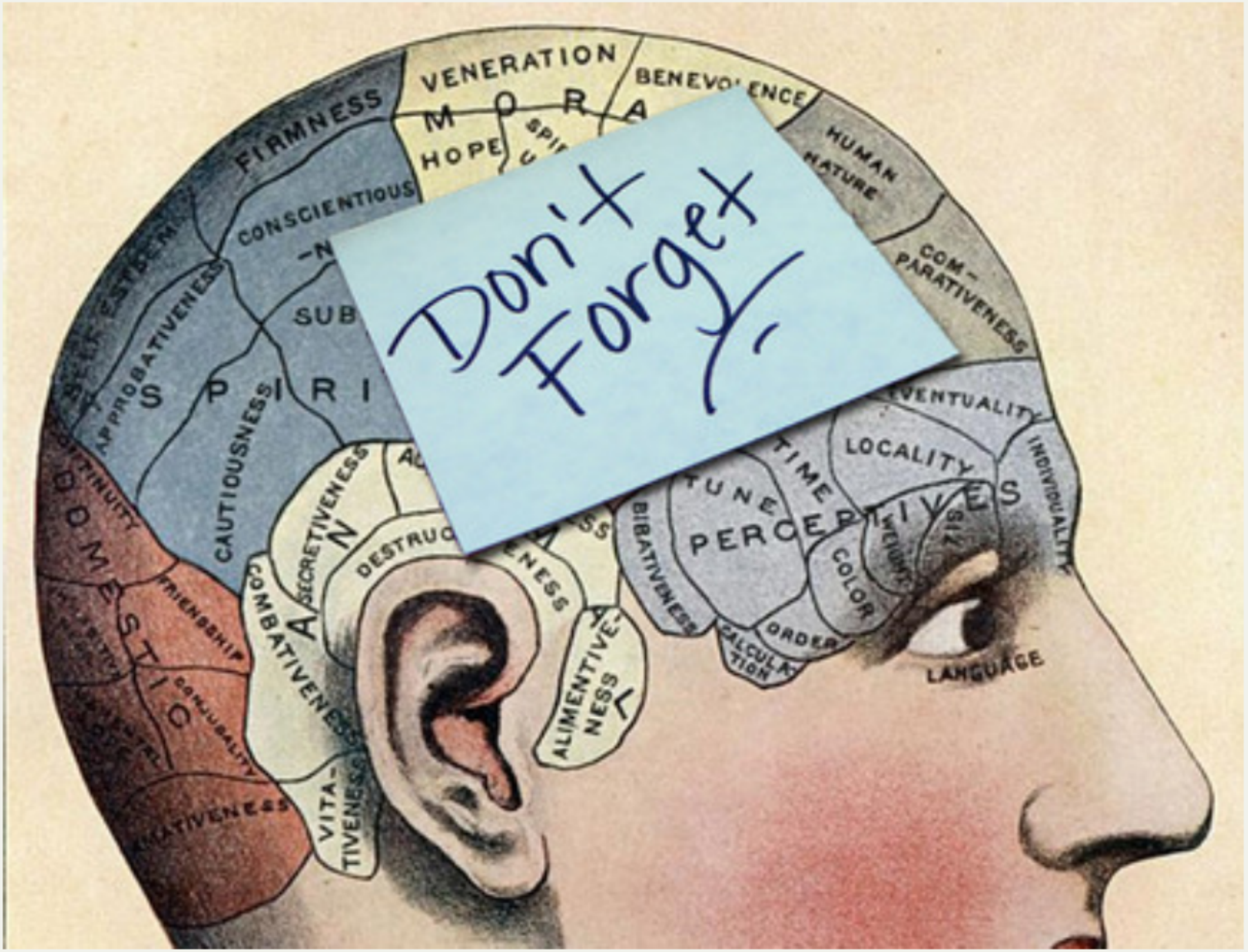
Co jest nie tak z tym rozwiązaniem?

Wrażliwość na zmiany - każdy moduł może być zmodyfikowany (dostęp globalny)

Brak skalowalności - przy dużej ilości modułów w aplikacji, ładujemy je wszystkie (zazwyczaj) zamiast tylko te wymagane

Co jest nie tak z tym rozwiązaniem?

Wrażliwość na zmiany - każdy moduł może być zmodyfikowany (dostęp globalny)
Brak skalowalności - przy dużej ilości modułów w aplikacji, ładujemy je wszystkie (zazwyczaj) zamiast tylko te wymagane
Nieproduktywność - ręczne rozwiązywanie zależności (kolejność skryptów etc.)


VENERATION
BENEVOLENCE
FIRMNESS
HOPE
CONSCIENTIOUS
HUMAN NATURE
COM-PARATIVENESS
SUB
APPROBATIVENESS
CAUTIOUSNESS
SECRETIVENESS
DESTRUC
COMBATIVENESS
ALIMENTIVE-NESS
VITA-TIVENESS
DOMESTIC
FRIENDSHIP
CONJUGALITY
ACQUISITIVENESS
BIBATIVENESS
CALCULA-TION
ORDER
COLOR
LANGUAGE
LOCALITY
PERCEPTIVES
Don't Forget

Gotowe rozwiązania

Gotowe rozwiązania

Dojo: dojo.require("some.module")

Gotowe rozwiązania

Dojo: dojo.require("some.module")

LABjs: $LAB.script("some/module.js")

Gotowe rozwiązania

Dojo: dojo.require("some.module")
LABjs: $LAB.script("some/module.js")
CommonJS: require("some/module")

Gotowe rozwiązania

Dojo: dojo.require("some.module")
LABjs: $LAB.script("some/module.js")
CommonJS: require("some/module")
AMD

Gotowe rozwiązania

Dojo: dojo.require("some.module")
LABjs: $LAB.script("some/module.js")
CommonJS: require("some/module")
AMD

...

Rozwiązanie

Mechanizm podobny do #include/import/require...

Mechanizm podobny do #include/import/require...

...z możliwością pobierania zagnieżdżonych zależności...

Mechanizm podobny do #include/import/require...

...z możliwością pobierania zagnieżdżonych zależności...

...łatwy w użyciu...

Rozwiązanie

Mechanizm podobny do #include/import/require...

...z możliwością pobierania zagnieżdżonych zależności...

...łatwy w użyciu...

...wspierany przez narzędzia do optymalizacji!

CommonJS

```
(function (module) {
    var getBarkStyle = function(isHowler) {
      return isHowler? 'wooooooow!': 'woof, woof!';
    };

    var Dog = function(name, breed) {
      this.bark = function() {
        return name + ': ' + getBarkStyle(breed === 'husky');
      };
    };

    var Wolf = function(name) {
      this.bark = function() {
        return name + ': ' + getBarkStyle(true);
      };
    };

    module.exports = {
      Dog: Dog,
      Wolf: Wolf
    };
})(module);
```

CommonJS

```
var Zoo = require('./zoo');

var myDog = new Zoo.Dog('Sherlock', 'beagle');
console.log(myDog.bark()); // Sherlock: woof, woof!

var myWolf = new Zoo.Wolf('Werewolf');
console.log(myWolf.bark()); // Werewolf: wooooooow!
```

CommonJS - zalety

CommonJS - zalety

Oficjalny format Node.js oraz komponentów NPM...

CommonJS - zalety

Oficjalny format Node.js oraz komponentów NPM...

...a to oznacza, że moduł CommonJS ma dostęp do całego ekosystemu NPM

CommonJS - zalety

Oficjalny format Node.js oraz komponentów NPM...

...a to oznacza, że moduł CommonJS ma dostęp do całego ekosystemu NPM

Prosta i wygodna składnia

CommonJS - zalety

Oficjalny format Node.js oraz komponentów NPM...

...a to oznacza, że moduł CommonJS ma dostęp do całego ekosystemu NPM

Prosta i wygodna składnia

Istnieje możliwość zagwarantowania kolejności ładowania modułów

CommonJS - wady

CommonJS - wady

Aby korzystać z niego potrzebne jest dodatkowe wsparcie (np. Browserify albo Webpack)

CommonJS - wady

Aby korzystać z niego potrzebne jest dodatkowe wsparcie (np. Browserify albo Webpack)

Ładowanie synchroniczne, a co za tym idzie - sekwencyjne

CommonJS - wady

Aby korzystać z niego potrzebne jest dodatkowe wsparcie (np. Browserify albo Webpack)

Ładowanie synchroniczne, a co za tym idzie - sekwencyjne

Zazwyczaj komponenty NPM posiadają wiele zależności - tzw. "dependency hell"

AMD - Asynchronous Module Definition

```
define('zoo', [], function() {
  var getBarkStyle = function (isHowler) {
    return isHowler? 'wooooooow!': 'woof, woof!';
  };

  var Dog = function (name, breed) {
    this.bark = function() {
      return name + ': ' + getBarkStyle(breed === 'husky');
    };
  };

  var Wolf = function (name) {
    this.bark = function() {
      return name + ': ' + getBarkStyle(true);
    };
  };

  return {
    Dog: Dog,
    Wolf: Wolf
  };
});
```

AMD - Asynchronous Module Definition

```
require(['zoo'], function(Zoo) {
  var myDog = new Zoo.Dog('Sherlock', 'beagle');
  console.log(myDog.bark()); // Sherlock: woof, woof!

  var myWolf = new Zoo.Wolf('Werewolf');
  console.log(myWolf.bark()); // Werewolf: wooooooow!
});
```

AMD - zalety

Równoległe ładowanie wielu modułów

AMD - zalety

Równoległe ładowanie wielu modułów

Łatwe oddalenie ładowania modułów nie potrzebnych przy ładowaniu strony

AMD - wady

Ładowanie asynchroniczne źle zaprojektowane potrafi doprowadzić do wyścigów

AMD - wady

Ładowanie asynchroniczne źle zaprojektowane potrafi doprowadzić do wyścigów

Nie można zagwarantować kolejności załadowania modułów

AMD - wady

Ładowanie asynchroniczne źle zaprojektowane potrafi doprowadzić do wyścigów

Nie można zagwarantować kolejności załadowania modułów

Bardziej skomplikowany i mniej czytelny zapis, szczególnie przy dużej ilości zależności (wartość tablicowa)

System.js

System.js

Uniwersalne narzędzie do ładowania modułów

System.js

Uniwersalne narzędzie do ładowania modułów

Wspiera takie moduły jak ES6, CommonJS, AMD, NodeJS oraz globalne zależności

System.js

Uniwersalne narzędzie do ładowania modułów

Wspiera takie moduły jak ES6, CommonJS, AMD, NodeJS oraz globalne zależności

Wsparcie dla Traceur i Babel (transpilers)

System.js

Uniwersalne narzędzie do ładowania modułów

Wspiera takie moduły jak ES6, CommonJS, AMD, NodeJS oraz globalne zależności

Wsparcie dla Traceur i Babel (transpilers)

Rozszerzalne za pomocą pluginów

System.js

```
<script src="javascript/system.js"></script>
<script type="text/javascript">
    System.config({
        baseURL: 'javascript'
    });

    System.import('main.js').then(function (Zoo) {
        var myDog = new Zoo.Dog('Sherlock', 'beagle');

        console.log(myDog.bark()); // Sherlock: woof, woof!

        var myWolf = new Zoo.Wolf('Werewolf');

        console.log(myWolf.bark()); // Werewolf: wooooooow!
    });
</script>
```

System.js - ladowanie wielu modułów

```
<script src="javascript/system.js"></script>
<script type="text/javascript">
    System.config({
        baseURL: 'javascript'
    });

    Promise.all([
        System.import('main.js'),
        System.import('jquery.js')
    ]).then(function(modules) {
        var Zoo = modules[0],
            jQuery = modules[1];

        var myDog = new Zoo.Dog('Sherlock', 'beagle');

        console.log(myDog.bark()); // Sherlock: woof, woof!

        var myWolf = new Zoo.Wolf('Werewolf');

        console.log(myWolf.bark()); // Werewolf: wooooooow!
    });
</script>
```

System.js - ladowanie innych zasobów

```
<script src="javascript/system.js"></script>
<script type="text/javascript">
    System.config({
        baseURL: ''
    });

    System.import('css/custom.css');
    System.import('images/image.png!image');
    System.import('data/config.json');
    System.import('data/data.txt!text');
</script>
```

Potencjalne problemy wykorzystania module loaderów

Potencjalne problemy wykorzystania module loaderów

Kompresja, transpiling, pre-procesing - zmieniają źródło skryptu

Potencjalne problemy wykorzystania module loaderów

Kompresja, transpiling, pre-procesing - zmieniają Źródło skryptu

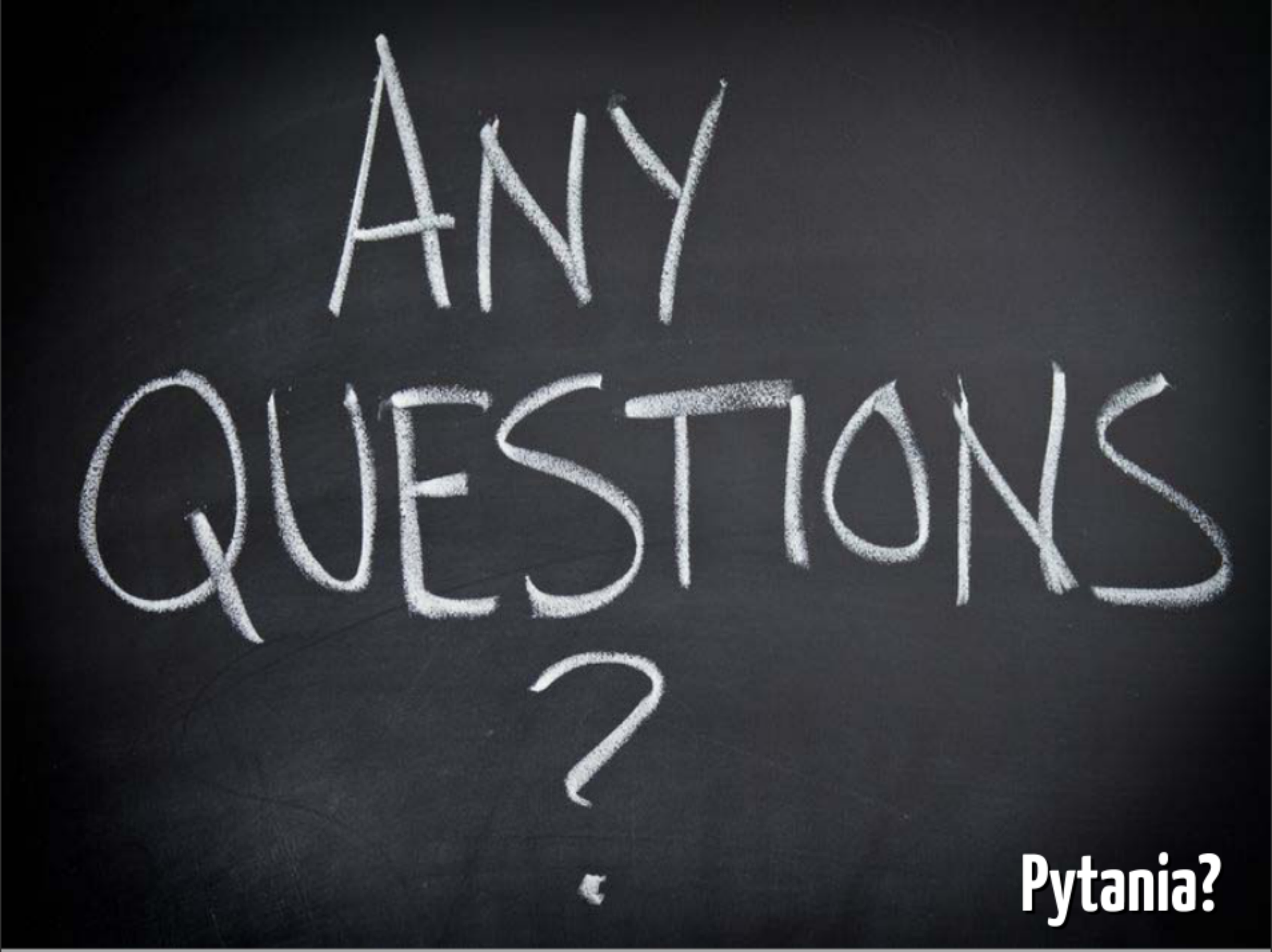
Częsty brak rozpoznawania skryptów nie załadowanych za pomocą tagu <script>

Potencjalne problemy wykorzystania module loaderów

Kompresja, transpiling, pre-procesing - zmieniają źródło skryptu

Częsty brak rozpoznawania skryptów nie załadowanych za pomocą tagu <script>

Problematyczna kolejność ładowania modułów - zależna od narzędzia



Pytania?