

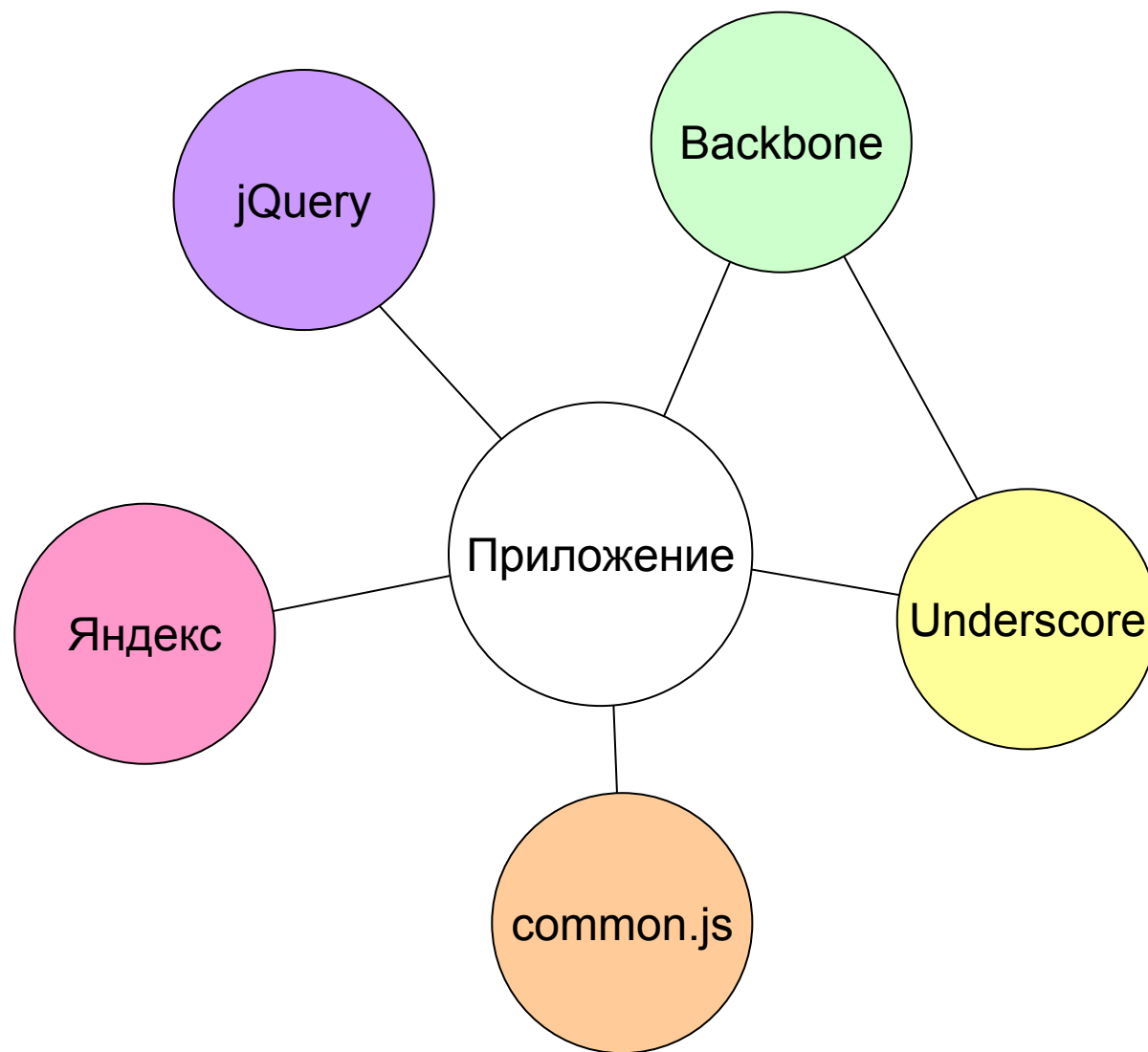
# JavaScript фреймворки. Куда катится мир.

Владимир Кузнецов

**UWDC2012**

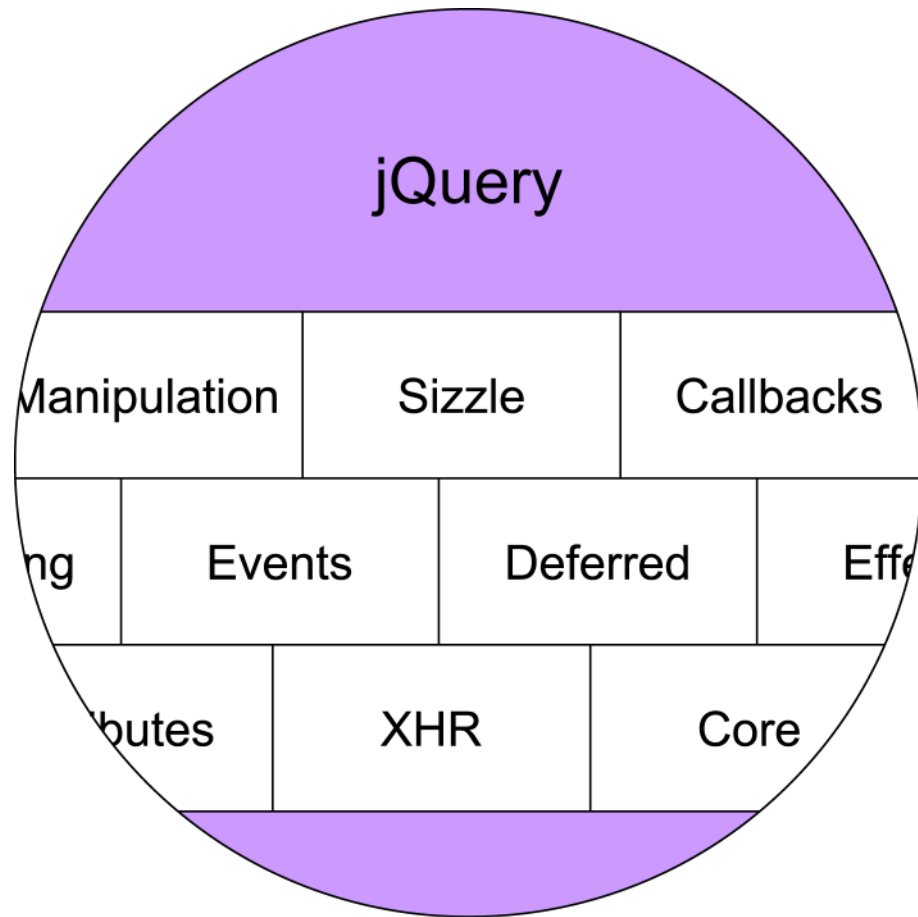


Framework — основа или  
каркас приложения



Файл с полезными  
функциями

# Модульность



...и еще масса мелких модулей

# Дробим на модули чтобы:

- улучшить архитектуру фреймворка;
- упростить командную разработку и тестирование;
- из «большого» фреймворка можно собрать «маленький» только с нужными функциями.

# Тестирование

# Тестирование

- Автоматическое дешевле и проще ручного.
- Дает возможность покрыть больше различных случаев.
- Не заменимо при рефакторинге.

# QUnit

QUnit example ■ noglobals ■ notrycatch

Hide passed tests

Mozilla/5.0 (Windows NT 5.1; rv:10.0) Gecko/20100101 Firefox/10.0

Tests completed in 38 milliseconds.  
5 tests of 6 passed, 1 failed.

1. a basic test example (0, 2, 2) Rerun
2. Module A: first test within module (0, 1, 1) Rerun
3. Module A: second test within module (0, 1, 1) Rerun
4. Module B: some other test (1, 1, 2) Rerun

1. failing test

Expected: false

Result: true

Diff: false true

Source: ()@http://code.jquery.com/qunit/git/qunit.js:106

2. passing test

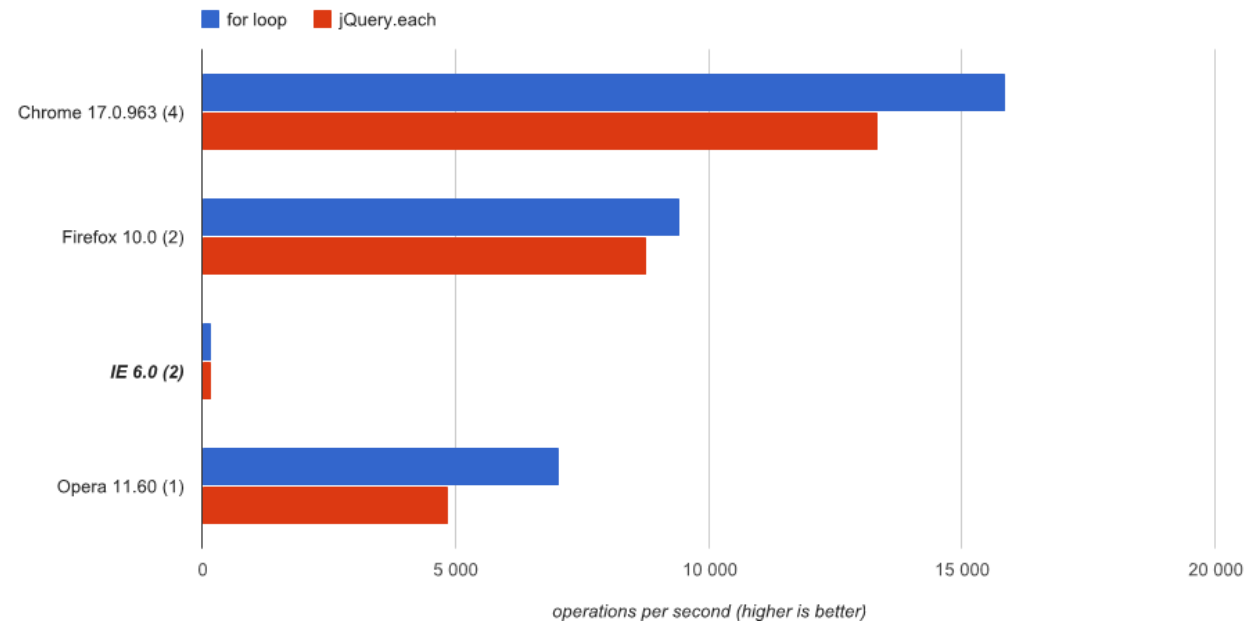
Expected: true

# Jasmine

```
describe("Test case", function() {  
  it('should be true', function() {  
    expect(App.method()).toEqual(true);  
  });  
});
```

# jsPerf

<b>jQuery.each</b>	<pre>\$.each(a, function() {   e = \$(this); });</pre>	192 ±7.61% fastest
<b>for loop</b>	<pre>var len = a.length; for (var i = 0; i &lt; len; i++) {   e = \$(a[i]); };</pre>	213 ±8.50% fastest



Selenium

# Точки расширения функциональности

# События

```
$(root).on("tableWasFilled.module1",  
    function () { alert("Filled"); });
```

```
${a: 1}.on("tableWasFilled.module1",  
    function () { alert("Filled"); });
```

# Функции обратного вызова

- Можно передать только одну функцию обратного вызова.
- Перед вызовом нужно проверить, что передали именно функцию.
- Явное связывание модулей.

# Отложенные объекты

- Можно зарегистрировать сколько угодно функций-обработчиков.
- Объект может изменить состояние только один раз.
- Явное связывание модулей.
- Мало кто понимает как это работает!

# Стиль кода

Программы пишут,  
чтобы их читать.

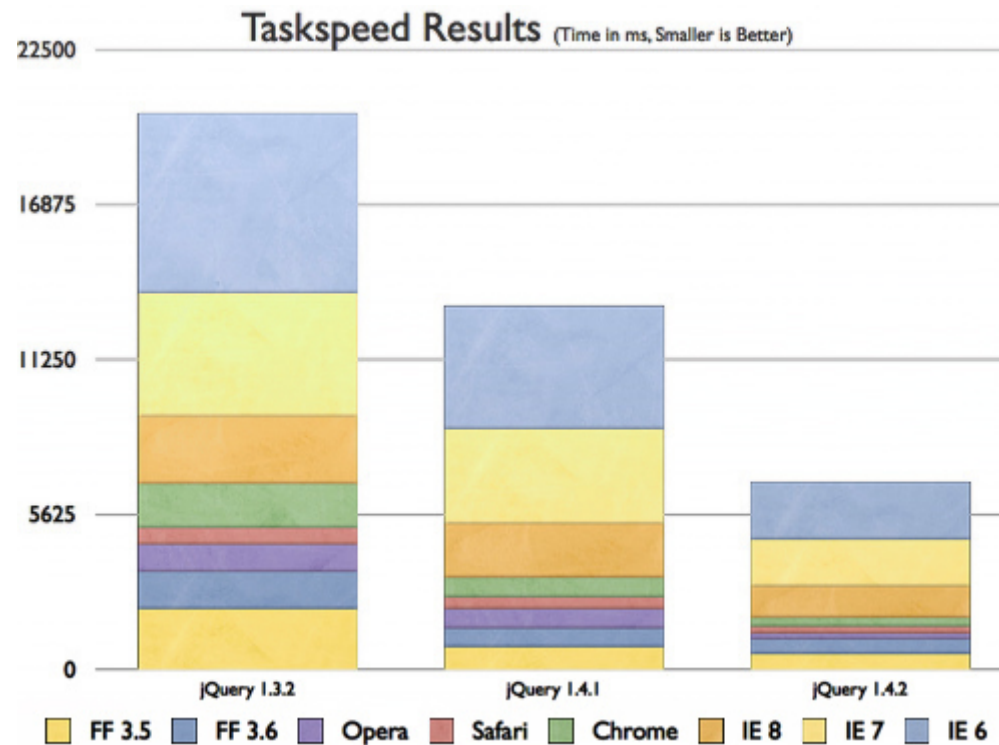
# Стиль кода

- Douglas Crockford: “Programming Style & Your Brain”
- JSLint в качестве валидатора
- Code Guidelines для вашей команды

# Рефакторинг

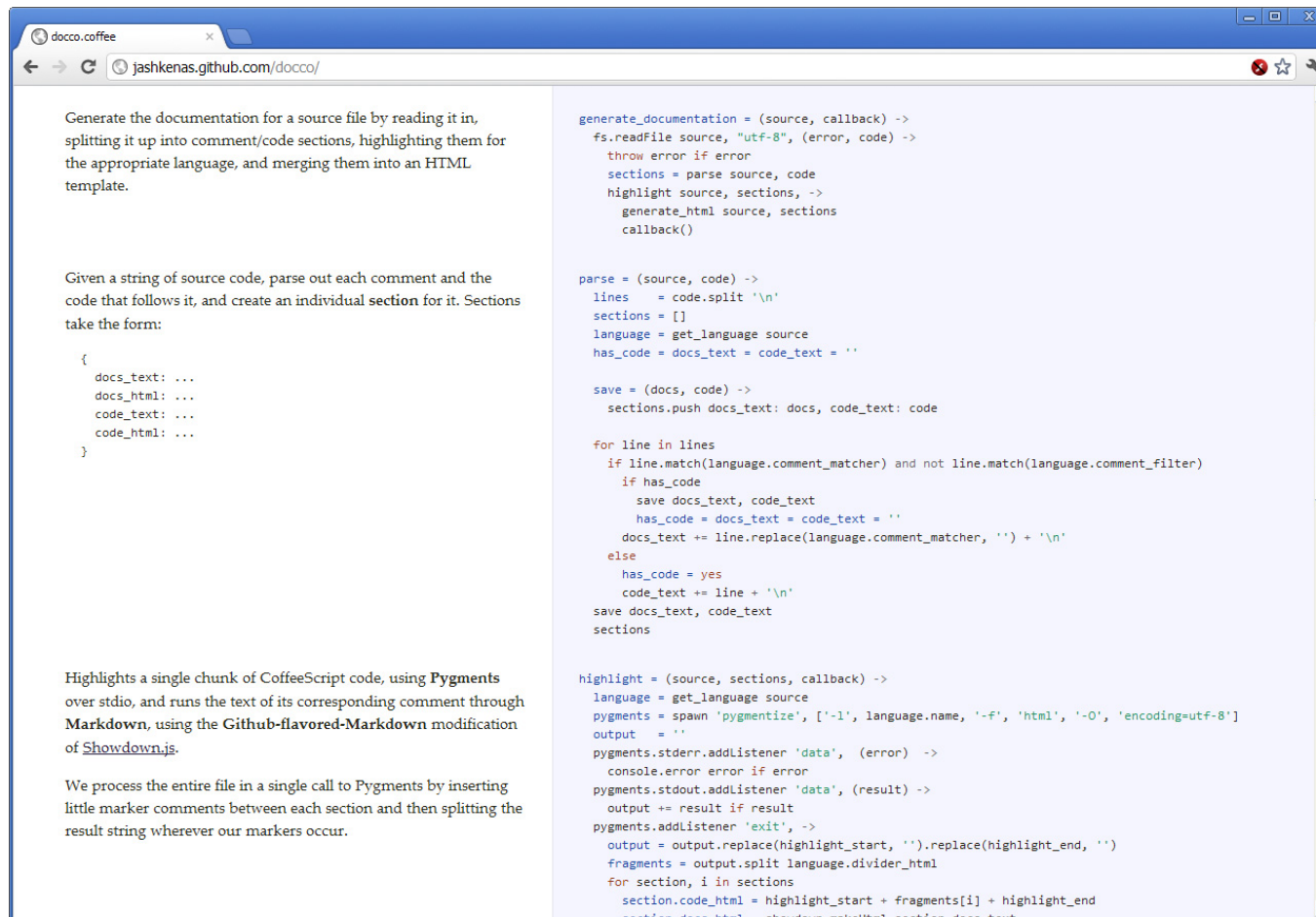
Не боимся трогать то,  
что работает.

# Ускорение работы jQuery



Документация

# docco



Generate the documentation for a source file by reading it in, splitting it up into comment/code sections, highlighting them for the appropriate language, and merging them into an HTML template.

Given a string of source code, parse out each comment and the code that follows it, and create an individual section for it. Sections take the form:

```
{
  docs_text: ...
  docs_html: ...
  code_text: ...
  code_html: ...
}
```

Highlights a single chunk of CoffeeScript code, using **Pygments** over **stdio**, and runs the text of its corresponding comment through **Markdown**, using the **Github-flavored-Markdown** modification of [Shodown.js](#).

We process the entire file in a single call to **Pygments** by inserting little marker comments between each section and then splitting the result string wherever our markers occur.

```
generate_documentation = (source, callback) ->
  fs.readFile source, "utf-8", (error, code) ->
    throw error if error
    sections = parse source, code
    highlight source, sections, ->
      generate_html source, sections
      callback()

parse = (source, code) ->
  lines = code.split '\n'
  sections = []
  language = get_language source
  has_code = docs_text = code_text = ''

save = (docs, code) ->
  sections.push docs_text: docs, code_text: code

for line in lines
  if line.match(language.comment_matcher) and not line.match(language.comment_filter)
    if has_code
      save docs_text, code_text
      has_code = docs_text = code_text = ''
    docs_text += line.replace(language.comment_matcher, '') + '\n'
  else
    has_code = yes
    code_text += line + '\n'
  save docs_text, code_text
  sections

highlight = (source, sections, callback) ->
  language = get_language source
  pygments = spawn 'pygmentize', ['-l', language.name, '-f', 'html', '-O', 'encoding=utf-8']
  output = ''
  pygments.stderr.addListener 'data', (error) ->
    console.error error if error
  pygments.stdout.addListener 'data', (result) ->
    output += result if result
  pygments.addListener 'exit', ->
    output = output.replace(highlight_start, '').replace(highlight_end, '')
    fragments = output.split language.divider_html
    for section, i in sections
      section.code_html = highlight_start + fragments[i] + highlight_end
      section.doc_html = language.divider_html + section.doc_text
```

# GitHub Markdown

📖 README.md

## Скелет приложения

Набор методов, которые позволяют организовать код.

### namespace()

---

```
.namespace( namespace )
```

```
.namespace( namespace, origin )
```

Модули и функции приложения не захламляют глобальное пространство имен, а группируются в специально отведенной для них переменной. Базовое имя задается в константе **NS**. Для совместимости с минимизаторами кода она определяется как параметр замыкания.

```
App.namespace("App.Module1");
```

Можно использовать укороченный синтаксис, чтобы получить аналогичный результат:

```
App.namespace("Module1");
```

Базовое имя будет подставляться автоматически, если он не будет найдено до первого разделителя.

Второй опциональный параметр задает исходный объект, который нужно расширить существующими полями и методами, если они уже есть.

# Чеклист

- Модульность
- Тесты
- Точки расширения
- Хороший стиль кода
- Документация

Решил сделать  
«идеальный» фреймворк.

<http://bit.ly/app-skeleton>

# Создаем пространство имён

```
App.namespace("App.Module", {  
    method: function () {  
        ...  
    }  
});
```

# Хранилище данных модуля

```
App.defaults("App.Module", {  
    text: "Hello, world!",  
    enabled: true,  
});
```

# Доступ к данным

```
var data = App.defaults("App.Module");
```

```
var enabled = App.defaults("App.Module",  
    "enabled");
```

```
var text = App.defaults("App.Module",  
    "error.text", "Default text");
```

# Описание зависимостей

```
App.register({  
  name: "module1",  
  path: [  
    "/js/module1.js", "/css/module1.css"  
  ],  
  requires: ["jQuery", "module2"]  
});
```

# Загрузка модуля

```
App.load({  
  load: App.calculate("module1");  
  complete: function () {  
    $(App.Module.init);  
  }  
});
```

# Загрузка обязательных ресурсов

```
App.bootstrap({  
  load: "jquery-1.7.1.min.js",  
  complete: function () {  
    // в этом месте уже можно декорировать  
    // страницу с применением jQuery.  
  }  
});
```

<http://bit.ly/app-skeleton>

- ☑ Модульность
- ☑ Тесты
- ☑ Точки расширения
- ☑ Хороший стиль кода
- ☑ Документация

# Спасибо!

Владимир Кузнецов

<http://noteskeeper.ru/>

[@mista\\_k](#)

