Tuna Framework

Кононенко Сергей

2012

План доклада

- Что и зачем?
- Почти MVC
- Быстрее InnerHTML

ЧТО И ЗАЧЕМ?

Что?

- Удобная и производительная client-side библиотека
- Невероятно* быстрый шаблонизатор
- Ничего лишнего

---

* – честно.

Зачем?

- Получать удовольствие от разработки
- Разделить верстку, JS и backend
- Легко добавлять и изолировать функционал*
- Не зависить от навязанной модели данных

---

* – без кучи событий.

ПОЧТИ MVC

Пример

```
<body data-widgets="button-group, form,
                    template-transformer">

  <form class="j-form" data-name="add-task">
     <input type="text" name="task" />
  </form>

  <!-- ... -->
```

Используемые виджеты

```
<body data-widgets="button-group, form,
                    template-transformer">

  <form class="j-form" data-name="add-task">
     <input type="text" name="task" />
  </form>

  <!-- ... -->
```

Объявление виджета

```
<body data-widgets="button-group, form,
                    template-transformer">

  <form class="j-form" data-name="add-task">
     <input type="text" name="task" />
  </form>

  <!-- ... -->
```

Параметры виджетов

```
<div class="j-button-group
            j-template-transformer"
     data-action-done=".j-task-done"
     data-template-id="todo_list_template"
     data-name="todo-list">

  <ul class="t-list"></ul>

</div>
```

Инициализация

```
tuna.control.init(document.body);
```

Контроллер

```
var MainController = function() {
  tuna.control.Controller.call(this);
};

MainController.prototype.initActions = function() {
  /* Инизиализация логики */
};
```

Доступ к виджетам

```
MainController.prototype.initActions = function() {
  var form = this._container.getWidget
                                ('form', 'add-task');
  form.submit();
};
```

Регистрация контроллера

```
<div data-controller-id="page">
    <!-- ... -->
</div>

tuna.control
    .registerController('page', new SomeController());
```

Основной контроллер

```
tuna.control
    .setApplicationController(new MainController());
```

Последовательность

1. Верстка
2. Объявление виджетов
3. Реализация логики приложения

Организация приложения

1. Композитная структура контроллеров
2. Кастомные виджеты (jQuery и все дела)
3. Фабрика запросов к REST API
4. Организация модели данных

Управление памятью

1.Контейнеры и навигация

2. Ленивый шаблонизатор

Сравнение

- Backbone's TODO* — 9.20 Mb
- Tuna Framework TODO** — 1.7 Mb

* - http://documentcloud.github.com/backbone/examples/todos/index.html
** - http://kononencheg.github.com/Tuna-Framework/todo-list.html3

БЫСТРЕЕ INNERHTML

Жизненный цикл

- Извлечение шаблона из специальной разметки
- Привязка к рабочей верстке (компиляция)
- Применение данных (отрисовка)
- Разрушение (верстка сохраняется)

Возможности

- Отрисовка текстовых данных
- Установка аттрибутов
- Списки любой вложенности
- Условное действие (установка CSS-класса)

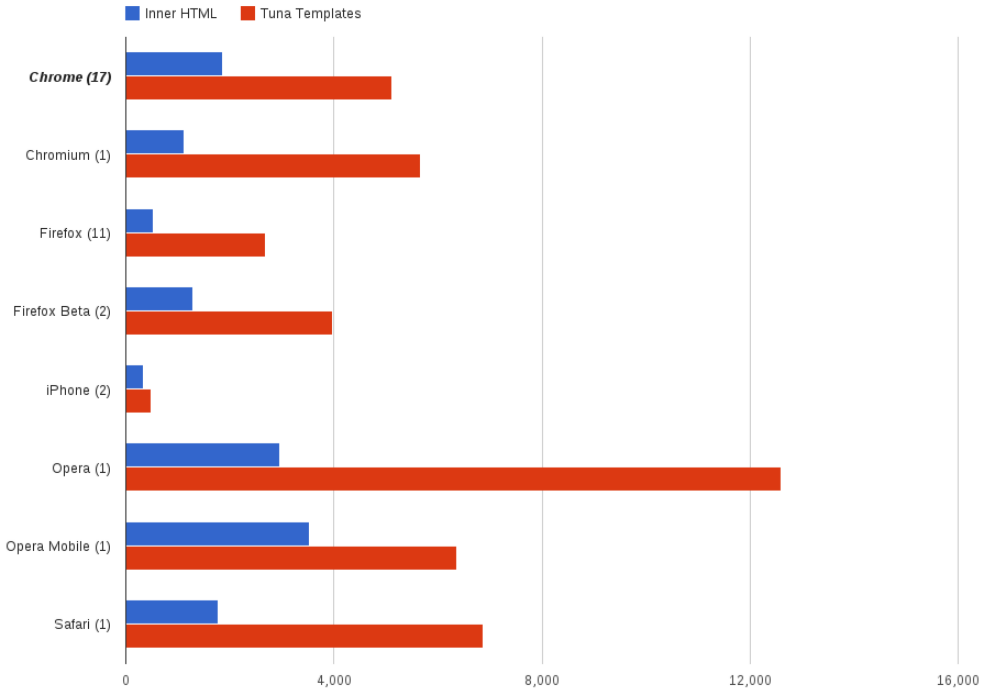
Плюсы

- ✔ Очень быстрая перерисовка
- ✔ Сохранение DOM-дерева
- ✔ Экономия памяти
- ✔ Аккуратное первое появление

Минусы

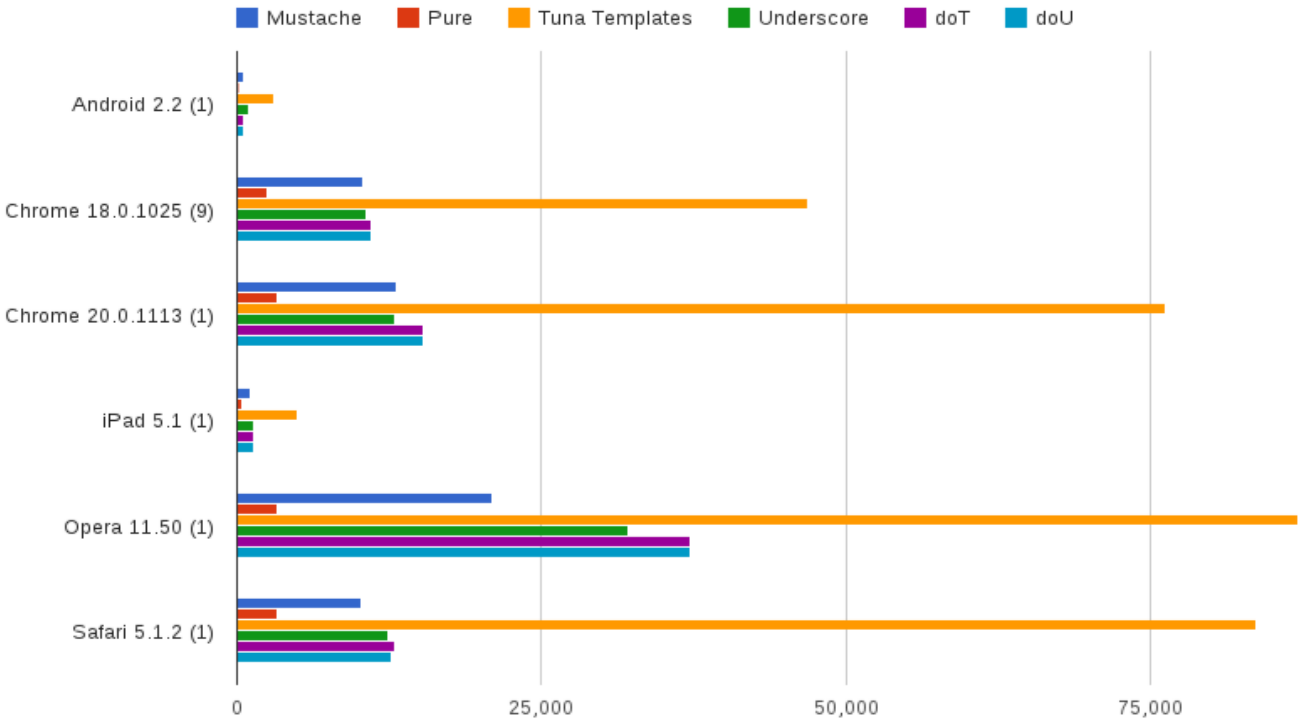
- ✗ Громоздкая разметка
- ✗ Зависит от Sizzle (только IE)

Операций в секунду *



* - http://jsperf.com/tuna-templates-vs-inner-html

Операций в секунду *



* - http://jsperf.com/tuna-templates-vs-mustache-vs-pure-vs-underscore/2

Выборка данных

```
{ "users": [
    { "name": "Сергей Кононенко",
      "isSelected": true,
      "imageUrl": "http://goo.gl/2pRFN" },
    { "name": "Милый котик",
      "isSelected": true,
      "imageUrl": "http://goo.gl/RHuA3" }
]}
```

Выборка данных

```
 users/0/name        -> "Сергей Кононенко"
/users/length        -> 2
 users/1/$key        -> 1
/users/0/../1/name   -> "Милый котик"
 users/1/../$key     -> "users"
```

Пример

```
<div id="user_info">
    <div class="avatar">
        <img class="t-avatar" width="50" height="50" />
    </div>
    <div class="name">
        <span class="t-first-name" ></span>
        <span class="t-last-name"></span>
    </div>
</div>
```

Пример

```
<tuna:template id="user_template">
    <tuna:spot tuna:target="t-last-name"
               tuna:path="lastName" />

    <tuna:spot tuna:target="t-first-name"
               tuna:path="firstName" />

    <tuna:attr tuna:target="t-avatar"
               tuna:path="avatarUrl" tuna:name="src" />
</tuna:template>
```

Пример

```
var templateContainer = document.getElementById('user_info');

var tunaTemplate = tuna.tmpl.compileFromMarkup
                        (templateContainer, 'user_template');

tunaTemplate.processTransform({
    "firstName": "Sergey",
    "lastName": "Kononenko",
    "avatarUrl": "http://goo.gl/2pRFN"
});
```

Готово!

Sergey Kononenko

СПАСИБО!

Спасибо за внимание!