

Яндекс

Яндекс

JQuery




Ванильный JS - какие есть проблемы

Длинный синтаксис для обращению к элементам. И не только.


Очень сложный и неструктурированный код для ванильной шаблонизации.

Двойственность при обращении к одному элементу/коллекции

Отсутствие каких-то удобных шорткатов для обычных штук типа скрытия/показа



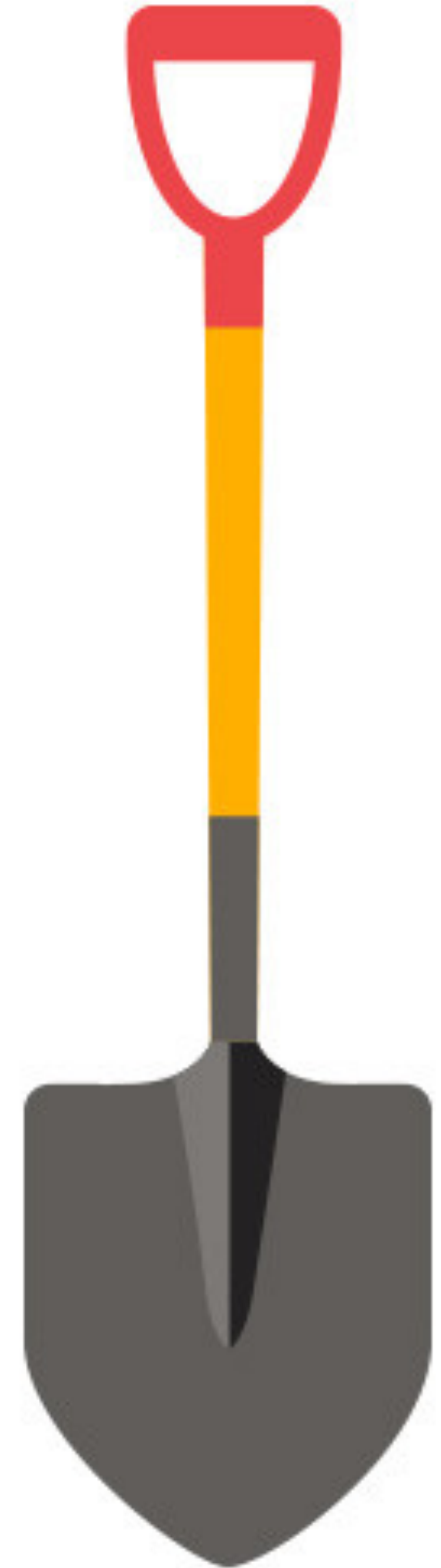
Все эти проблемы можно
решать вручную



Существует много
фреймворков, которые
позволяют вам в том или
ином виде забыть о
ванильном синтаксисе

| Самый простой - jQuery

Что умеет jQuery



Некоторые возможности jQuery

Проще обращаться к элементу `$('.element')`

Единый способ поменять содержимое элемента и элементов `$('.elements').text('ololo')` меняет текст у всех элементов с таким классом, независимо от того, много их или один

Аналогично, удобно подписываться на события `$('.elements').click(function() { console.log('ololo'); });`

Ну и еще всякое...



Обо всем по порядку

Подключение

Сайт jQuery

<https://jquery.com/>

Download

Подключаем удобным способом

????

PROFIT!!!



\$

\$

Доллар - основная функция фреймворка. Он работает по-разному, в зависимости от аргумента.

Если передадим функцию - то она станет обработчиком события onDOMContentLoaded

```
$(function() {  
    console.log('тут безопасно писать код');  
});
```

\$

Доллар - основная функция фреймворка. Он работает по-разному, в зависимости от аргумента.

Если передадим строку-селектор, то будут найдены все объекты-узлы, соответствующие этому селектору, и результатом будет нечто вроде «коллекции», но с блекджеком и всяким

```
$( 'a.link' ); // все теги a с классом link, «завернутые» в  
                jQuery объект
```

\$

Доллар - основная функция фреймворка. Он работает по-разному, в зависимости от аргумента.

Если передадим строку-html код, то будет создана верстка с таким кодом. Также завернутая в jQuery возможности. Вполне удобно так создавать кирпичики для верстки

```
$( '<div/>' ); // создаст div, «завернутый» в jQuery объект
```

\$

Доллар - основная функция фреймворка. Он работает по-разному, в зависимости от аргумента.

Если передадим объект-узел, то он будет «завернут» в возможности jQuery, и в получившейся структуре будут доступны все методы. То же касается коллекции

```
var element = document.querySelector('.element');  
$(element); // объект-узел, «завернутый» в jQuery
```

Подробности про создание верстки



`$('<div/>')`

Самой частой сигнатурой использования является вот такая

```
$(' <a/>', {  
    href: 'https://ya.ru',  
    'class': 'avatar',  
    text: 'click me!'  
});
```

Второй объект трактуется как атрибуты создаваемого объекта-узла, а поле `text` - как содержимое.

Подробности про поиск по селектору



`$('.element')`


В случае, если вы ищете элементы документа по селектору, то вторым аргументом можно указать контекст поиска, например вот так

```
$('.element', $('.container'));
```

В качестве контекста может быть и jQuery объект и обычный объект-узел

**Краткий курс всего, что
мы обсуждали в
переделке на jQuery**





Простейшее взаимодействие
с содержимым


Содержимое тега

`$('.element').text()` - в таком виде работает как считывание `.innerText`, вернет текст внутри тега

`$('.element').text('ololo')` - а вот так работает как запись в поле `.innerText`, установит текст внутрь элемента

`$('.element').html()` - считывание `.innerHTML`, вернет html внутри тега

`$('.element').html('ololo')` - запись в поле `.innerHTML`, установит верстку внутрь элемента



Добавление элементов и
шаблонизация

Добавление элементов

`$('.element').append($('#<div/>'))` - добавляет аргумент в конец детей объекта слева от точки

Аналогичные методы:

`$('.element').prepend($('#<div/>'))` - в начало детей

`$('.element').after($('#<div/>'))` - после «себя»

`$('.element').before($('#<div/>'))` - после «собой»

Шаблонизация

Если всегда использовать `.append`, то можно шаблонизировать примерно вот так:

```
$( '.element' ).append(  
    $( '<h1/>', { text: 'Занятие 15' } ).append(  
        $( '<small/>', { text: 'jQuery' } )  
    )  
);
```



Перемещение по DOM

Перемещение по DOM

`$('.element').prev()` - аналогично `previousElementSibling`

`$('.element').next()` - аналогично `nextElementSibling`

`$('.element').parent()` - аналогично `parent`

`$('.element').children()` - аналогично `children`

`$('.element').closest('.container')` - ближайших из родителей, на который сматчится селектор



События

События

Подписка на события осуществляется похожим образом

```
$( '.element' ).click(function() {  
    // обработчик события  
});
```

this внутри обработчика - объект-узел (HTML Element) на котором возникло событие. Методов jQuery не будет. Чтобы получить всю магию, надо завернуть this в \$

```
$( '.element' ).click(function() {  
    $(this).text('отличный клик!');  
});
```

Делегирование, подписка на будущие события

Если подписаться на события через `$(document).on`, то можно «в кредит» подписать на события для всех элементов с соответствующим селектором, еще даже не добавленных в документ.

```
$(document).on('click', '.element', function() {  
    // обработчик события  
});
```

Теперь на все клики внутри документа мы будем реагировать, если цель клика имеет селектор `.element`



AJAX

\$.ajax

Создание аякс-запроса.

```
$.ajax({  
    url: 'test.html',  
    context: document.body  
}).done(function() {  
    $(this).addClass('done');  
});
```


`$.ajax` - важнейшие параметры

`url` - ссылка, на которую делаем запрос

`dataType` - влияет на то, как будет трактоваться ответ сервера

`method` - GET/POST запрос

`context` - то, что будет записано в `this` в обработчиках

`data` - данные, которые вы хотите передать на сервер

`cache` - кешировать или нет ответы браузера

`timeout` - через сколько миллисекунд считать запрос неуспешным

\$.ajax - обработчики

.done - успех

.fail - неудача

.always - выполнить при любом исходе

Эти обработчики можно так же передать в виде событий `success`, `error`, `complete`, передавая функции в соответствующие поля Ajax-запроса (в тот же объект, в котором передаются настройки)

Глобальные AJAX события

ajaxSend - момент перед отправкой запроса

ajaxSuccess - успешное выполнение запроса

ajaxError - ошибка при запросе

ajaxComplete - сразу после окончания запроса

Можно использовать, чтобы делать какие-то общие действия

```
$(document).ajaxComplete(function(event, xhr) {  
    // например, убираем паранджу  
});
```



Эффекты

Яндекс

Спасибо

Шлейко Александр

Разработчик интерфейсов



dusty@yandex-team.ru



@dustyo_O