
Курс Javascript

— jQuery и lodash —

Занятие 7. Темы для обсуждения

1. Зачем нужны библиотеки jQuery и lodash.
2. Подключения jQuery.
3. Использование jQuery / lodash.
4. Написание jQuery plugin.
5. jQuery Ajax.

Зачем нужны библиотеки jQuery и lodash?

jQuery – это небольшая, быстрая, легко расширяемая JS библиотека, которая делает такую работу как навигацию по DOM, обратку событий, выбору и аякс запросы более удобными, включая в себя кросс браузерность.

В работе очень часто используется переменная `$`, которая является тем же аналогом объекта **jQuery**.

Сайт документации: <http://api.jquery.com/>

lodash - эта JS библиотека помогает в работе с массивами, объектами, строками, числами и т.д. беря на себя основную часть работы (*подразумевается что библиотека содержит очень много готовых решений для расчетов или перебора данных.*)

Сайт документации: <https://lodash.com/>

Подключения jQuery (у lodash такой же вариант)

Для того, чтобы использовать **jQuery**, нужно скачать саму библиотеку или же использовать (**cdn** - ресурс, предоставляющий онлайн версию библиотеки).

Пример:

`<script type='text/javascript' src='/js/jquery.2.1.1.js'></script>` - локально из папки `js` в проекте (можно подключать `jquery.2.1.1.min.js` файл для минифицированной версии библиотеки, рекомендуется для продакшена).

`<script type='text/javascript' src='https://code.jquery.com/jquery-2.2.4.js'></script>` - подключения через `cdn` jQuery, аналоги таких `cdn` есть у `google`, тоже часто подключаются на `дев/продакшен` сайтах.

еще не плохой ресурс - <https://cdnjs.com>

Заметка:

После подключения, весь последующий JS код должен быть написан **ПОСЛЕ** подключенной библиотеки (включая jQuery плагины!).

Использования jQuery / lodash

jQuery (альтернативный объект `$`):

jQuery можно вызывать как функцию обертку: `jQuery('#content h1.title')` что альтернативно нативному `querySelector/All`.

Так же, после такой обертки, DOM элемент получает ряд методов **jQuery**.

С **jQuery** можно вызывать методы напрямую:

```
jQuery.each();
```

```
jQuery.ajax();
```

lodash (обычно используют переменную - `_`)

lodash использует вызов методов напрямую через объект `_`

`_.filter();` -> фильтр по массиву объектов

<https://lodash.com/docs/4.17.4#filter>

`_.map();`

Набор всех методов можно увидеть тут:

<https://lodash.com/docs/4.17.4>

написание jQuery plugin

Написать **jQuery plugin** очень просто (*подразумевается оболочка плагина, то, насколько будет сложен плагин, зависит от разработчика*).

1. Плагин должен быть вынесен в отдельный js файл, и иметь характерное название (*необязательно, но приветствуется, к примеру `jquery.my_plugin.js`*).
2. Плагин должен быть подключен **ОБЯЗАТЕЛЬНО** после самой **jQuery** библиотеки.

Пример оболочки плагина (функция):

```
;(function($) {  
    $.fn.myPlugin = function(options) {  
        // Код вашего плагина...  
    }  
})(jQuery);
```

Пример вызова: `jQuery("#some-selector").myPlugin(optionsObj);`

jQuery Ajax

jQuery ajax облегчает работу с **XMLHttpRequest** (ссылка почитать <https://learn.javascript.ru/ajax-xmlhttprequest>) запросами. Ссылка на доку <http://api.jquery.com/jquery.ajax/>

Пример использования:

```
$.ajax({
  url: 'http://jsonplaceholder.typicode.com/posts',
  method: 'GET',
  beforeSend: function() {
    //alert('123');
  }
}).then(function(data, xhr) {
  console.log(data);
  $.each(data, function(index) {

    if (index == 10) return false;

    var elem = $('<div></div>', {
      html: "<h2>" + this.title + "</h2><p>" + this.body + "</p>",
      class: "item"
    });
    $('#content').append(elem);
  });
}, function(xhr) {});
```

Литература

<http://api.jquery.com/>

<https://lodash.com/docs/4.17.4>

<https://learn.javascript.ru/ajax-xmlhttprequest>

практика запросов ајах:

<http://jsonplaceholder.typicode.com>