

JavaScript: инструменты, сервер

Пугачёв Константин (K.V.Pugachev@inp.nsk.su)

2018.12.11 14h

В предыдущей серии

- С 1990х работает WWW, с 1995 – JavaScript, развивается и сейчас
- Браузер показывает HTML, применяет стили, исполняет скрипты
- В DOM есть элементы страницы, можно достучаться через JS, назначить обработчики событий
- JavaScript выполняется в одном потоке, применяют асинхронные вычисления
- Асинхронные функции записываются через коллбеки, промисы, `async/await`

Веб – это много модных (и вышедших из моды) инструментов

- **Бандлеры** – собирают много файлов в один (напр: browserify, webpack)
- **Трансляторы** языков, компилирующихся в JS (напр: CoffeeScript, TypeScript)
- **Транспайлеры** (напр: Babel)
- **Препроцессоры CSS** (напр: Less, Sass, Stylus)
- Другие инструменты (расскажите о своём любимом)...

Веб – это много модных (и вышедших из моды) инструментов

- Фреймворки

- Для преодоления JS (напр: jQuery)
- Для отображения данных (напр: Bootstrap, jQuery UI)
- Для борьбы с изменяемым состоянием (напр: RxJS – работает с бесконечными потоками событий)
- Для внесения в код MVC и Тнтерпрайзности (напр: Angular)
- Другие фреймворки (расскажите о своём любимом)...

HTTP – протокол передачи данных

- Клиент отправляет заголовки и тело запроса
 - Заголовки (Cache-Control, If-Modified-Since, User-Agent, ...)
 - Методы (в основном – GET и POST, другие)
- Сервер отправляет заголовки и тело ответа
 - Заголовки (Content-Type, WWW-Authenticate, Set-Cookie, ...)
 - Коды состояния
 - 1xx – иду-иду
 - 2xx – всё хорошо
 - 3xx – всё течёт, всё изменяется
 - 4xx – клиент послал что-то не то
 - 5xx – сервер не смог обработать запрос

AJAX, XmlHttpRequest

AJAX (Asynchronous Javascript and XML) – загрузка XML (и прочих данных) в фоне без перезагрузки страницы.

XmlHttpRequest – специальный класс для работы с запросами (см. xmlhttprequest.ru)

- Синхронный (блокирующий) или асинхронный (с колбеками/событиями)
- Поддерживает разные методы
- Отлавливает разные этапы загрузки
- Можно отменить

Асинхронная отправка формы методом POST

```
function asyncPOST(url, params, cb) {  
    var xhr = new XMLHttpRequest;  
    xhr.open('POST', url, true /* async */);  
  
    // для GET параметры идут в URL, для POST – в тело запроса  
    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  
  
    xhr.onreadystatechange = function() {  
        if (xhr.readyState == 4) {  
            cb(xhr.status == 200 ? xhr.responseText : null);  
        }  
    };  
  
    // формируем форму  
    var qs = Object.keys(params)  
        .map(k => k + '=' + encodeURIComponent(params[k]))  
        .join('&');  
  
    xhr.send(qs && '?' + qs);  
  
    return xhr;  
}  
  
// использование:  
// asyncPOST('/', {name: 'anon', age: '20'}, console.log);
```

Краткий код, устранение браузерных несовместимостей.

- Выбор элементов по селекторам, поиск элементов

```
var buttons = $('.button');
```

- Конструирование элементов, манипуляции

```
$('.button')  
  .append($('
```

- Настройка стилей, обработчиков событий

```
$('.button')  
  .css('color', '#def')  
  .click(function(event){  
    $(this).parent().find('.uname').val() = 'anonimous';  
  });
```


Краткий код, устранение браузерных несовместимостей.

- Живые события (навешиваются на ещё не созданные элементы)

```
$( document ).on( "click", ".button", function() {  
    $(this).parent().find('.uname').val() = 'anonymous';  
});
```

- Анимации

```
$('.button').fadeIn();
```

Краткий код, устранение браузерных несовместимостей.

- AJAX

```
$.ajax({  
  method: "POST",  
  url: "/my/url",  
  data: { name: "John", location: "Boston" }  
})  
.done(function( msg ) {  
  alert( "Data Saved: " + msg );  
})  
.fail(function( jqXHR, textStatus ) {  
  alert( "Request failed: " + textStatus );  
});
```

- И многое другое

jQuery – монадический подход

jQuery работает, пользуясь множественными вычислениями – свойством монады List.

```
// погасить все или никакие  
$('.button').fadeIn();
```

```
// все i внутри всех p внутри всех контейнеров  
$('.container').find('p').find('i');
```

```
S '.container' >>= find 'p' >>= find 'i' -- List Monad
```

jQuery расширяема

// Простой плагин:

```
$.fn.greenify = function() {  
    this.css( "color", "green" );  
    return this; // чтобы работали цепочки вызовов  
}
```

// его использование:

```
$( "a" ).greenify().addClass( "greenified" );
```

jQuery расширяема, внутри плагина – стрелка Клейсли

```
$.fn.fields = function() {  
  
    return this.each(function() {  
        return $(this).find('input');  
    });  
  
    return this;  
}  
  
$( ".form" ).fields().val('42');
```

```
fnFields xs = concat . map (find 'input') $ xs  
fnFields xs = (find 'input') =<< xs
```

Array.prototype.map vs \$.fn.map

```
[1,2,3].map(x => x+1)  
// [2, 3, 4]
```

```
$.map([1,2,3], x => x+1)  
// [2, 3, 4]
```

```
[1,2,3].map(x => Array(x).fill(x))    // map  
// [[1], [2, 2], [3, 3, 3]]
```

```
$.map([1,2,3], x => Array(x).fill(x)) // concatMap aka >>=  
// [1, 2, 2, 3, 3, 3]
```

jQuery вообще использует монадический подход и слабую типизацию JS

```
(+1) <$> [1,2,3]      -- [2,3,4]  
(\x -> replicate x x) =<< [1,2,3] -- [1,2,2,3,3,3]
```

jQueryUI – библиотека с полезными виджетами

- Кнопки
- Меню
- Диалоги
- Слайдеры

server-side

- CGI-скрипты – stdin/stdout
- скрипты на PHP, ASP, ASP.NET – скрипты внутри фреймворка/языка на сервере
- сервер на Node.js – свой сервер под дудку фреймворка
- свой рукописный сервер

Node.JS

Node.JS – рантайм для создания приложений, связанных с сетью

- JavaScript на сервере
- Асинхронность возведена в абсолют
- кроссплатформенно (Linux, MacOS, Windows, Android)
- пакетный менеджер NPM
- работает с
 - HTTP(S) (http, https)
 - UDP (dgram)
 - бинарными массивами (Buffer), потоками (stream.Stream)
 - файловой системой (fs)
 - криптографией (crypto)
 - многим другим см. API
- Можно писать дополнения на C++

`npm install <имя модуля>` – и всё работает

Но

- Загрузить модуль может кто угодно
- Он может удалить свой модуль из-за борьбы с захватчиками имён (напр. `leftpad`)
- Или отдать свой модуль хакеру, который будет его поддерживать и воровать криптовалюту (напр. `event-stream`)

Модули в JS, CommonJS

- Раньше их не было, придумали
 - AMD
 - CommonJS
- Node.js использует нотацию CommonJS

```
// twice.js
const theirs = require('their-module');
const mine = require('./my/module');

module.exports = x => x * 2;
module.exports.call = (f, x) => f(x);

// -----
var twice = require('./twice');
console.log(twice(3)); // 6
console.log(twice.call(x => x+4, 2)); // 6
```

- В ECMAScript 2016 появились честные модули

HTTP сервер под Node.js

```
const http = require('http');
const url = require('url');

const hostname = '127.0.0.1', port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  var u = url.parse(req.url, true);

  if(u.pathname == '/') {
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello World\n');
    return;
  }

  // u.query.name - GET-параметр name
  res.setHeader('Content-Type', 'text/html');
  res.end(`<html><body>Hello, <b>${u.query.name}<b></body></html>`);
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

HTTP сервер под Node.js, POST

```
const http = require('http');
const url = require('url');
const formidable = require('formidable');

const hostname = '127.0.0.1', port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  var u = url.parse(req.url, true);

  if(u.pathname == '/') {
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello World\n');
    return;
  }

  var form = new formidable.IncomingForm();

  form.parse(req, function(err, fields, files) {
    // fields.name - POST-параметр name
    res.setHeader('Content-Type', 'text/html');
    res.end(`<html><body>Hello, <b>${fields.name}<b></body></html>`);
  });
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Node.js: полезные модули

Для **HTTP(S)** сервера:

- `mysql` – работа с RDBMS MySQL
- `formidable` – работа с POST-формами
- `sessions` – работа с сессиями (ID сессии передаётся в куках)
- `node-static` – отдача статического содержимого

Для жизни

- `pegjs` – генератор парсеров (грамматика задаётся в виде, приближенном к РБНФ)
- `nodemailer` – для отправки писем
- `ssh2` – подключение через SSH
- `search-index` – для индексирования и полнотекстового поиска
- `express` – для создания веб-приложений

Советы для работы с WEB+Node.js

- Не используйте тривиальные модули (напр. `leftpad`), используйте нетривиальные модули (напр. `mysql`).
- Модули изменятся. Записывайте версии модулей, с которыми еваш сервер ещё работает.
- Модули удаляются. Храните локальные копии установленных модулей.
- Учитывайте все ветви в асинхронном коде (`done/fail`, `then/catch`)
- Оборачивайте асинхронные функции в промисы и используйте `async/await`
- Не следуйте моде, следуйте здравому смыслу. Реализация и поддержка должны быть дешёвыми, заказчик платит за функционал.

Где водятся родственники ECMAScript

- WSH – Windows Script Host

Можно писать на MS JScript и MS VBScript скрипты для управления Windows

- JScript.NET

Язык от MS для .NET. В отличие от ECMAScript

- может быть типизированным (явные задания типов переменных, функций; строгая проверка количеств и типов аргументов для интерфейсов)
- с ООП
- Компилируемый (в байткод .NET)
- Но eval работает

- Espruino

Микроконтроллер, где можно писать на JS.

Задачи

Знаний достаточно для реализации всех задач:

- Задача 5 об асинхронных вычислениях
- Задача 6 о клиент-серверном приложении web-chat

Внимание, лекции, задания и сниппеты обновляются.