Разработка пользовательского веб-интерфейса

XHR, CORS, setTimeout/setInterval, исключения, область видимости, замыкания

Сергей Геннадьевич Синица

КубГУ, 2020

sin@kubsu.ru

XMLHttpRequest

Придуман Microsoft

Выполняет HTTP-запрос из JavaScript

Любой метод HTTP

AJAX?

Cross Domain Policy, CORS

Бывает синхронным и асинхронным

Автоматическое перенаправление

Cookies, Авторизация

Создание объекта XHR

В 6-й версии Internet Explorer:

```
var xmlhttp =
  new ActiveXObject("Microsoft.http");
```

В других браузерах:

```
var xmlhttp = new XMLHttpRequest();
```

Методы XHR

open(method, URL, async, login, password)

setRequestHeader(’name’,’value’)

send(content)

abort()

getAllResponseHeaders()

getResponseHeader(name)

Свойства XHR

onreadystatechange = function(event) (обработчик асинхронного запроса)

readystate (статус асинхронного запроса)

responseText (ответ текстом)

responseXML (ответ в XML)

status (код статуса HTTP, например 200)

statusText (текст статуса HTTP, например Ok)

withCredentials (true/false отправка Cookies и WWW-Authenticate в CORS-запросах)

Свойство readystate

Для асинхронного запроса, число от 0 до 4:

- 0 – объект не инициализирован;
- 1 – идет загрузка;
- 2 – загрузка окончена;
- 3 – идет обмен с сервером;
- 4 – запрос завершен, можно получать результат.

Пример синхронного запроса

```
<script>
var xmlhttp = new XMLHttpRequest();
function go() {
  xmlhttp.open('GET', '/web8/json/a.txt', false);
  xmlhttp.send(null);
  document.getElementById('frag').innerHTML =
    xmlhttp.responseText;
  return false;
}
</script>
<div id="frag"> </div>
<form>
  <input type="submit" value="Пуск"
    onclick="return go();"/>
</form>
```

Пример асинхронного запроса

```
<script>
var xmlhttp = new XMLHttpRequest();
function go() {
  xmlhttp.open('GET', '/web8/json/a.txt', true);
  xmlhttp.onreadystatechange = function() {
   if (xmlhttp.readyState == 4 &&
       xmlhttp.status == 200) {
      document.getElementById("frag").innerHTML =
       xmlhttp.responseText;
   }
  }
  xmlhttp.send(null);
  return false;
}
</script>
<div id="frag"> </div>
<form>
  <input type="submit" value="Пуск"
    onclick="return go();"/>
</form>
```

encodeURI("русский")

%D0%B1%C2%80%D0%B1%C2%83%D0%B1%C2%81%D0%B1%C2%81%D0%B0%D0%9A%D0%B0%D0%98%D0%B0%D0%99

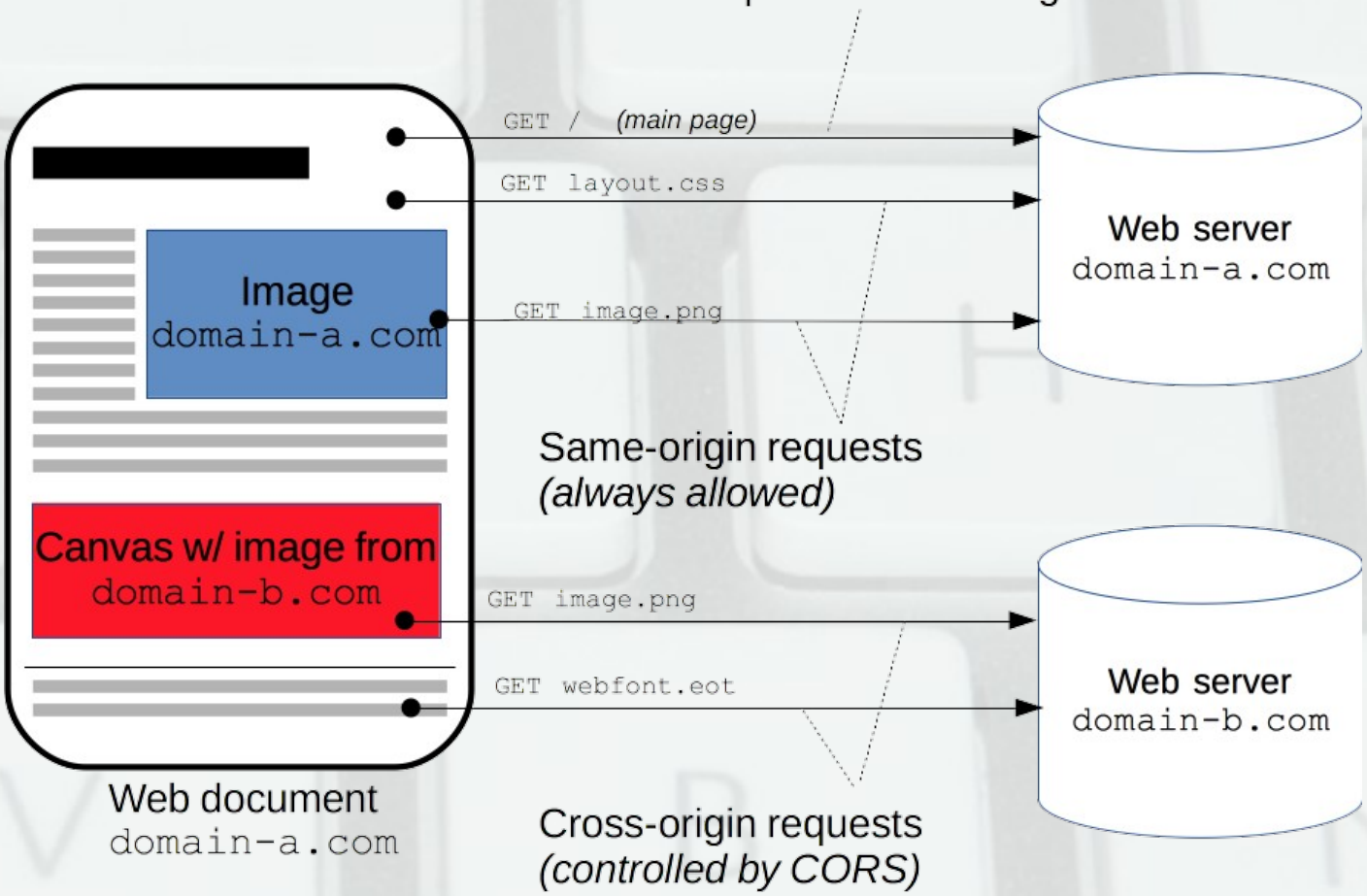
Пример отправки POST

```
xmlhttp.open("POST","ajax.php",true);

xmlhttp.setRequestHeader("Content-type",
  "application/x-www-form-urlencoded");

xmlhttp.send("comment=Hello%20World!&name=Anonymous");
```

Пример jQuery

```
// Загрузка содержимого элемента #frag
// с сервера методом GET.
$('#frag').load('/ajax.php');

// Асинхронный POST-запрос.
$.ajax({
  type: "POST",
  url: "/ajax.php",
  data: "key1=value1&key2=value2",
  beforeSend: function(XHR) {
    // Задаем заголовки запроса.
  },
  success: function(){
   // Обрабатываем ответ.
  }
});
```

CORS


Main request: defines origin.
GET / (main page)
GET layout.css
GET image.png
Web server
domain-a.com
Image
domain-a.com
Same-origin requests
(always allowed)
Canvas w/ image from
domain-b.com
GET image.png
GET webfont.eot
Web server
domain-b.com
Web document
domain-a.com
Cross-origin requests
(controlled by CORS)

CORS

```
var invocation = new XMLHttpRequest();
var url = 'http://bar.other/resources/public-data/';

function callOtherDomain() {
  if(invocation) {
    invocation.open('GET', url, true);
    invocation.onreadystatechange = handler;
    invocation.send();
  }
}
```

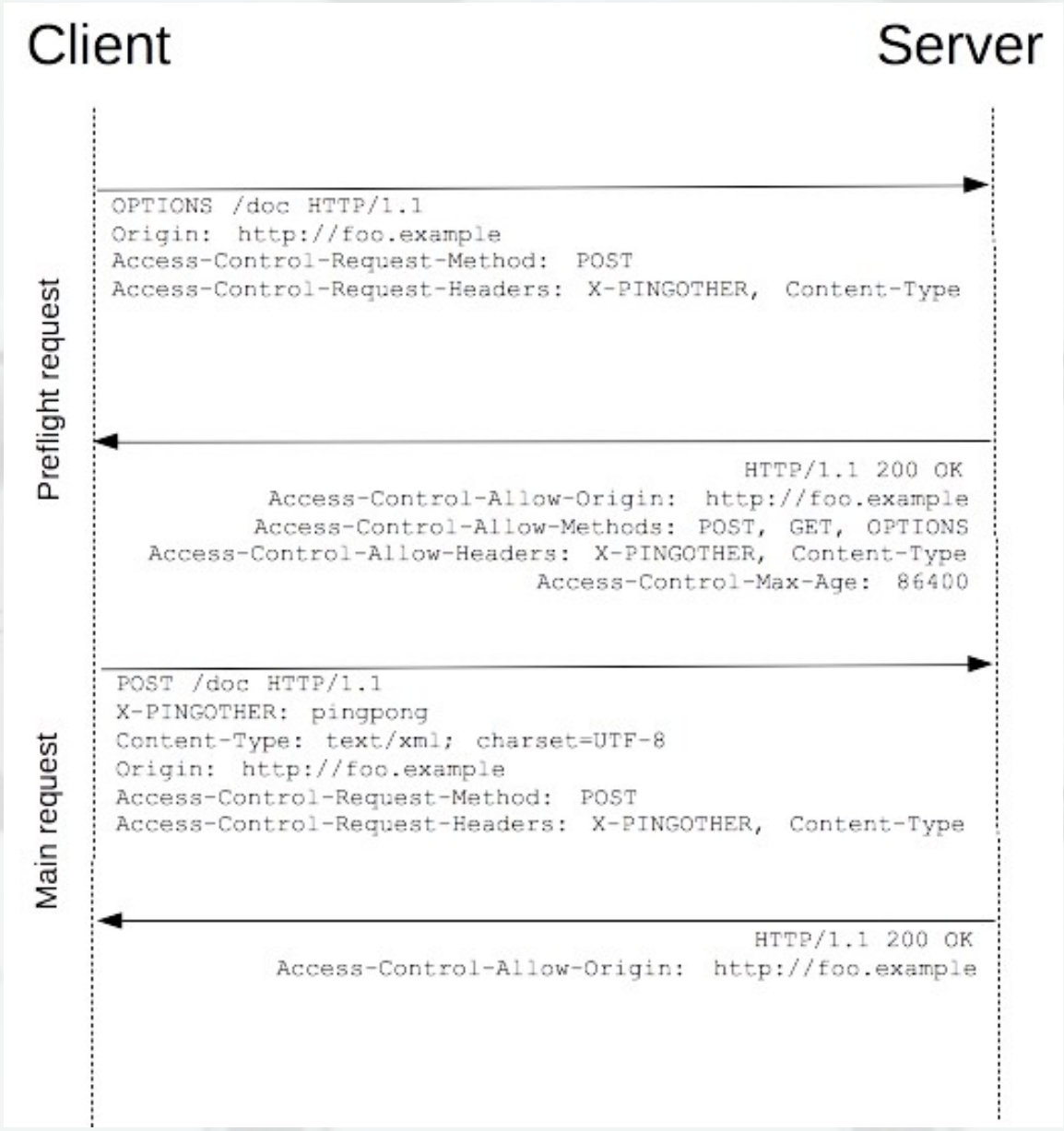
https://developer.mozilla.org/ru/docs/Web/HTTP/CORS

```
GET /resources/public-data/ HTTP/1.1
Host: bar.other
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US; rv:1.9.1b3pre)
    Gecko/20081130 Minefield/3.1b3pre
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Referer: http://foo.example/examples/access-control/simpleXSInvocation.html
Origin: http://foo.example

HTTP/1.1 200 OK
Date: Mon, 01 Dec 2008 00:23:53 GMT
Server: Apache/2.0.61
Access-Control-Allow-Origin: *
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/xml
[XML Data]
```

https://developer.mozilla.org/ru/docs/Web/HTTP/CORS

```
var invocation = new XMLHttpRequest();
var url = 'http://bar.other/resources/post-here/';
var body = '<?xml version="1.0"?><person><name>Arun</name></person>';

function callOtherDomain(){
  if(invocation)
    {
      invocation.open('POST', url, true);
      invocation.setRequestHeader('X-PINGOTHER', 'pingpong');
      invocation.setRequestHeader('Content-Type', 'application/xml');
      invocation.onreadystatechange = handler;
      invocation.send(body);
    }
}

......
```

Все «не стандартные» запросы (стандартные это POST и GET с «безопасными» заголовками), предварительно «просматриваются» запросом методом OPTION:
https://developer.mozilla.org/ru/docs/Web/HTTP/CORS


Client
Server
Preflight request
OPTIONS /doc HTTP/1.1
Origin: http://foo.example
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER, Content-Type
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://foo.example
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER, Content-Type
Access-Control-Max-Age: 86400
Main request
POST /doc HTTP/1.1
X-PINGOTHER: pingpong
Content-Type: text/xml; charset=UTF-8
Origin: http://foo.example
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER, Content-Type
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://foo.example

setTimeout / clearTimeout

```
var timeoutID = scope.setTimeout(function[, delay, arg1,
arg2, ...]);
var timeoutID = scope.setTimeout(function[, delay]);
var timeoutID = scope.setTimeout(code[, delay]);

// timeoutID уникальны
// Scope обычно window, но может быть и worker.

scope.clearTimeout(timeoutID);
```

```
<h1>setTimeout Example</h1>
<button onclick="delayedAlert();">Show an alert box after
two seconds</button>
<button onclick="clearAlert();">Cancel alert before it
happens</button>

<script>
var timeoutID;

function delayedAlert() {
  timeoutID = window.setTimeout(window.alert, 2*1000, 'That
was really slow!');
}

function clearAlert() {
  window.clearTimeout(timeoutID);
}
</script>
```

this это проблема

```
myArray = ['zero', 'one', 'two'];
myArray.myMethod = function (sProperty) {
  alert(arguments.length > 0 ? this[sProperty] : this);
};

myArray.myMethod(); // prints "zero,one,two"
myArray.myMethod(1); // prints "one"

setTimeout(myArray.myMethod, 1.0*1000); // prints "[object
Window]" after 1 second

setTimeout(myArray.myMethod, 1.5*1000, '1'); // prints
"undefined" after 1.5 seconds

setTimeout.call(myArray, myArray.myMethod, 2.0*1000); //
error: "NS_ERROR_XPC_BAD_OP_ON_WN_PROTO: Illegal operation
on WrappedNative prototype object"
```

Решение

```
// prints "zero,one,two" after 2 seconds
setTimeout(function(){myArray.myMethod()}, 2.0*1000);

// prints "one" after 2.5 seconds
setTimeout(function(){myArray.myMethod('1')}, 2.5*1000);
```

setInterval / clearInterval

```
var intervalID = scope.setInterval(func, delay, [arg1, arg2,
...]);
var intervalID = scope.setInterval(code, delay);

// delay >= 10 мс

scope.clearInterval(intervalID)

// IntervalID те же, что и для clearTimeout!
```

Лексическая область видимости

```
function init() {
    // name - локальная переменная, созданная в init
    var name = "Mozilla";
    // displayName() - внутренняя функция, замыкание
    function displayName() {
        // displayName() использует переменную, объявленную в
родительской функции
        alert (name);
    }
    displayName();
}
init();
```

В JavaScript область действия переменной определяется по её расположению в коде (это очевидно лексически), и вложенные функции имеют доступ к переменным, объявленным вовне. Этот механизм и называется Lexical scoping (область действия, ограниченная лексически).

Замыкания (closures)

```
function makeFunc() {
  var name = "Mozilla";
  function displayName() {
    alert(name);
  }
  return displayName;
};

var myFunc = makeFunc();
myFunc();
```

Замыкание — это комбинация функции и лексического окружения, в котором эта функция была объявлена. Это окружение состоит из произвольного количества локальных переменных, которые были в области действия функции во время создания замыкания.

https://developer.mozilla.org/ru/docs/Web/JavaScript/Closures

Замыкания (closures)

```
function makeAdder(x) {
  return function(y) {
    return x + y;
  };
};

var add5 = makeAdder(5);
var add10 = makeAdder(10);

console.log(add5(2));  // 7
console.log(add10(2)); // 12
```

Замыкания (closures)

```
<a href="#" id="size-12">12</a>
<a href="#" id="size-14">14</a>
<a href="#" id="size-16">16</a>
…

function makeSizer(size) {
  return function() {
    document.body.style.fontSize = size + 'px';
  };
};

var size12 = makeSizer(12);
var size14 = makeSizer(14);
var size16 = makeSizer(16);

document.getElementById('size-12').onclick = size12;
document.getElementById('size-14').onclick = size14;
document.getElementById('size-16').onclick = size16;
```

[scope]

Интерпретатор, при доступе к переменной, сначала пытается найти переменную в текущем LexicalEnvironment, а затем, если её нет – ищет во внешнем объекте переменных. В данном случае им является window.

Такой порядок поиска возможен благодаря тому, что ссылка на внешний объект переменных хранится в специальном внутреннем свойстве функции, которое называется [[Scope]]. Это свойство закрыто от прямого доступа, но знание о нём очень важно для понимания того, как работает JavaScript.

При создании функция получает скрытое свойство [[Scope]], которое ссылается на лексическое окружение, в котором она была создана.

В примере выше таким окружением является window, так что создаётся свойство:

```
sayHi.[[Scope]] = window
```

Это свойство никогда не меняется. Оно всюду следует за функцией, привязывая её, таким образом, к месту своего рождения.

https://learn.javascript.ru/closures

[scope]

Если переменная не найдена в функции – она будет искаться снаружи.

- Каждая функция при создании получает ссылку [[Scope]] на объект с переменными, в контексте которого была создана.

- При запуске функции создаётся новый объект с переменными LexicalEnvironment. Он получает ссылку на внешний объект переменных из [[Scope]].

- При поиске переменных он осуществляется сначала в текущем объекте переменных, а потом – по этой ссылке.

Значение переменной из внешней области берётся всегда текущее. Оно может быть уже не то, что было на момент создания функции.

[scope]

```
var phrase = 'Привет';

function sayHi(name) {
  alert(phrase + ', ' + name);
}

sayHi('Вася');  // Привет, Вася (*)

phrase = 'Пока';

sayHi('Вася'); // Пока, Вася (**)
```

[scope] и возврат функции

```
function makeCounter() {
  var currentCount = 1;

  return function() { // (**)
    return currentCount++;
  };
}

var counter = makeCounter(); // (*)

// каждый вызов увеличивает счётчик и возвращает результат
alert( counter() ); // 1
alert( counter() ); // 2
alert( counter() ); // 3

// создать другой счётчик, он будет независим от первого
var counter2 = makeCounter();
alert( counter2() ); // 1
```

Immidiately invoked function expression (IIFE)

```
(function () {
    statements
})();
```

Пример:

```
(function () {
    var aName = "Barry";
})();
// Variable name is not accessible from the outside scope
aName // throws "Uncaught ReferenceError: aName is not
defined"
```

https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/IIFE

Immidiately invoked function expression (IIFE)

Переменная, которой присвоено IIFE, хранит в себе результат выполнения функции, но не саму функцию.

```
var result = (function () {
    var name = "Barry";
    return name;
})();
// Immediately creates the output:
result; // "Barry"
```

Исключения (exceptions)

```
function getMonthName(mo) {
  mo = mo - 1; // Adjust month number for array index (1 = Jan, 12 =
Dec)
  var months = ["Jan","Feb","Mar","Apr","May","Jun","Jul",
                "Aug","Sep","Oct","Nov","Dec"];
  if (months[mo]) {
    return months[mo];
  } else {
    throw "InvalidMonthNo"; //throw keyword is used here
  }
}

try { // statements to try
  monthName = getMonthName(myMonth); // function could throw exception
}
catch (e) {
  monthName = "unknown";
  logMyErrors(e); // pass exception object to error handler -> your own
}
```

Исключения (exceptions)

```
function f() {
  try {
    console.log(0);
    throw "bogus";
  } catch(e) {
    console.log(1);
    return true;    // приостанавливается до завершения блока `finally`
    console.log(2); // не выполняется
  } finally {
    console.log(3);
    return false;   // заменяет предыдущий `return`
    console.log(4); // не выполняется
  }
  // `return false` выполняется сейчас
  console.log(5);  // не выполняется
}
f();               // отображает 0, 1, 3 и возвращает `false`

https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/
Control_flow_and_error_handling
```

Захват и логирование стека

```
window.addEventListener('error', function (e) {
    var stack = e.error.stack;
    var message = e.error.toString();
    if (stack) {
        message += '\n' + stack;
    }
    var xhr = new XMLHttpRequest();
    xhr.open('POST', '/log', true);
    xhr.send(message);
});
```

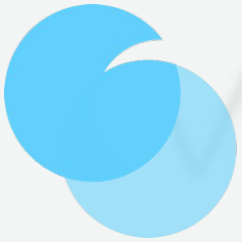
```
TypeError: foo.bar is not a function
error@http://localhost:1337/scripts/error.js:3:12
@http://localhost:1337/scripts/properHandler.dom.js:6:13
EventListener.handleEvent*@http://localhost:1337/scripts/properHandler.dom.js:5:9
@http://localhost:1337/scripts/properHandler.dom.js:1:2
```

Бесплатный бекэнд

Serverless...

SAAS-формы

https://slapform.com/docs/ajax-submission/
https://formcarry.com/documentation/getting-deeper#using-ajax



formcarry.

Slapform

```
<script
src="https://code.jquery.com/jquery-3.2.1.min.js"
integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="
crossorigin="anonymous"></script>


<script type="text/javascript">
  $( document ).ready(function() {
    $.ajax({
      url: 'https://api.slapform.com/your@email.com',
      dataType: 'json',
      method: 'POST',
      data: {
        name: 'Jon Snow',
        message: 'Hello World! This is my first Slapform submission.',
        /* Name triggers */
        slap_subject: 'My Favorite Message',
        slap_replyto: 'custom@replyto.com'
        slap_debug: false,
        slap_webhook: 'https://yourwebsite.com/webhook',
        slap_honey: ''
        /* These Slapform Name Triggers exist but aren't applicable to AJAX submissions */
        // slap_redirect: 'https://yourwebsite.com',
        // slap_captcha: false,


      },
      success: function (response) {
        console.log('Got data: ', response);
        if (response.meta.status == 'success') {
          console.log('Submission was successful!');
          // window.location.href = 'https://google.com' // Uncomment this line if you want to
redirect the user upon a successful submission
        } else if (response.meta.status == 'fail') {
          console.log('Submission failed with these errors: ', response.meta.errors);
        }
      }
    });
  });
</script>
```

Доктора:

- Умение гуглить не делает из вас специалиста

Программисты: