

Лекция 4. Передача данных между PHP и JavaScript. AJAX. JSON

При разработке веб-ресурсов с активным использованием javascript, всегда возникает вопрос, как передать переменную из JavaScript в PHP. Поскольку JavaScript выполняется на стороне пользователя, а PHP на стороне сервера, то значение не может быть передано напрямую. Поэтому результат выполнения кода JavaScript будем передавать в PHP через протокол HTTP. Рассмотрим несколько способов, каждый из которых имеет определенные преимущества (удобства) перед другим.

1. Передача данных из PHP в JavaScript

Для этого нужно во время выполнения скрипта PHP просто сформировать JavaScript, чтобы он потом мог выполняться в браузере.

Например:

```
<?php
$var = 'Hello!!!';
print '<script language="javascript">alert("'.$var.'");</script>';
?>
```

Если вы создадите файл PHP с таким кодом и запустите его в браузере будет сообщение в окне alert Hello!!!.

Разумеется нужно помнить что синтаксис должен быть JavaScript. Фактически, когда формируется код JavaScript в PHP разработчик должен знать что PHP сначала что-то генерирует и это что-то должно быть валидным JavaScript.

PHP без проблем генерирует JavaScript, а вот в обратной связи есть определенные сложности. Поэтому придется формировать новый запрос к серверу передавая в нем данные для PHP. Есть два основных метода передачи данных. Реализация может быть как через синхронные, так и через асинхронные запросы. Синхронные вызовут перезагрузку страницы, а асинхронные используют Ajax.

2. Передача данных из JavaScript в PHP

Синхронный метод передачи данных из JavaScript в PHP (с использованием метода GET)

Необходимо разместить в странице нужный JavaScript-код или сгенерировать его на стороне сервера, и передать в браузер пользователя. В этом в качестве примера определим ширину и высоту экрана пользователя (полезно для адаптивной верстки):

```
<?php
// проверяем существование переменных $width и $height
if (isset($_GET['width']) AND isset($_GET['height'])) {
    // если переменные существуют, то выводим полученные значения на экран
    echo 'Ширина экрана: ' . $_GET['width'] . "<br />";
    echo 'Высота экрана: ' . $_GET['height'] . "<br />";
}
// если переменные не существуют, то выполняем следующее
else {
```

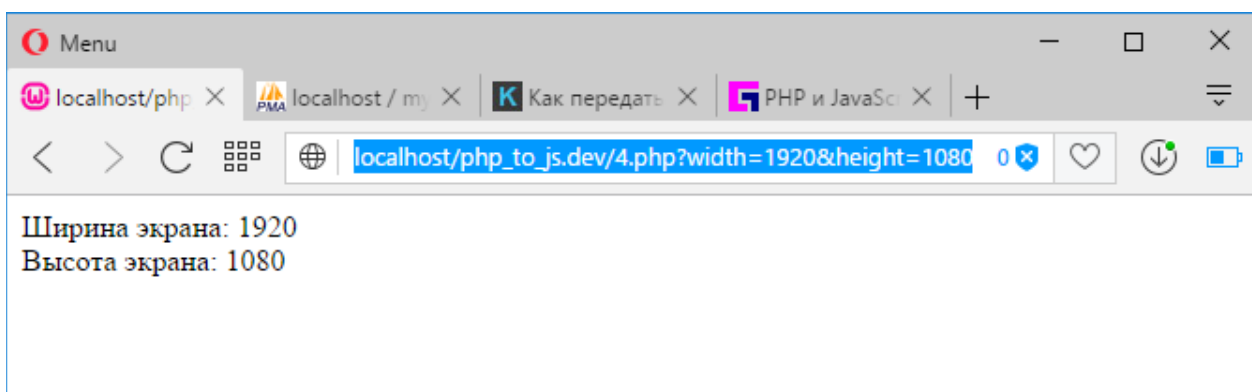
```
// PHP сгенерирует код JavaScript, который обработает браузер
// пользователя и передаст значения обратно PHP-скрипту через
протокол HTTP
echo "<script language='javascript'>";
echo "
location.href=\"\${_SERVER['SCRIPT_NAME']}?\${_SERVER['QUERY_STRING']}\"
    . \"width=\" + screen.width + \"&height=\" + screen.height;\n";
echo "</script>";
}
?>
```

Если браузер пользователя поддерживает JavaScript, то после выполнения скрипта страница обновится и в адресной строке отобразится запрос:

`http://путь_к_скрипту/имя_скрипта.php?width=1920&height=1080`

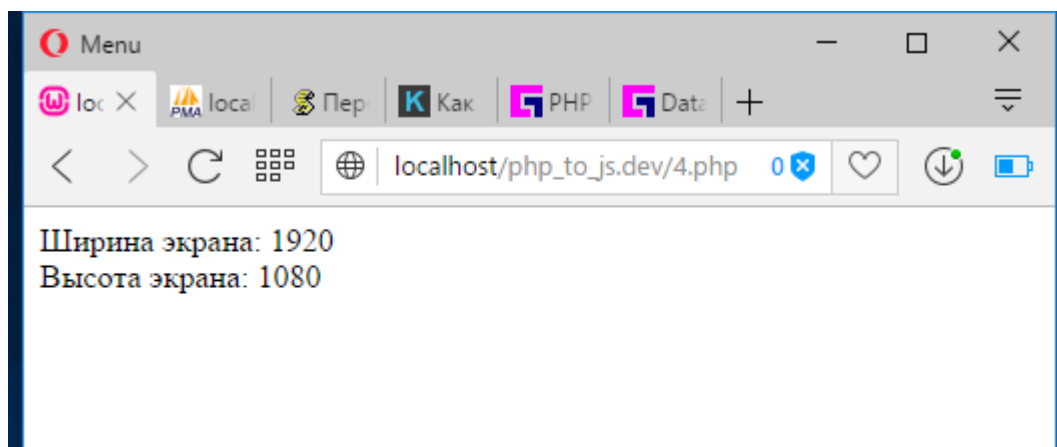
В нашем случае это:

`http://localhost/php_to_js.dev/4.php?width=1920&height=1080`



Знак ? после имени скрипта, очевидно, указывает веб-серверу что будем сделать GET-запрос к PHP-скрипту, после него идет имя переменной и знак равенства, который присваивает переменной следующее за ним значение, определенное JavaScript-функцией. Знак & служит разделителем для переменных в запросе. В результате в PHP будут переданы две переменные: \$width со значением равным ширине экрана (1920) и \$height со значением равным высоте экрана (1080).

А на экране увидим результат выполнения скрипта – ширина и высота экрана пользователя.

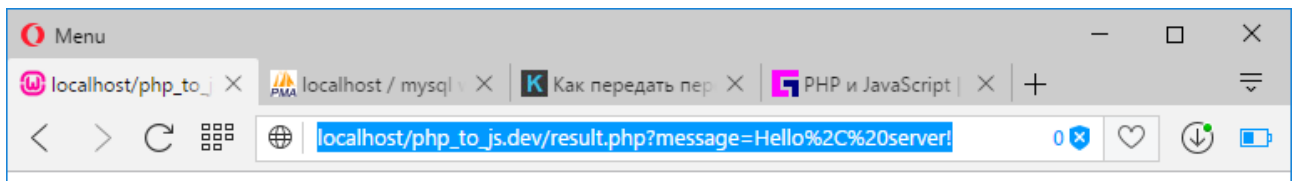


Рассмотрим еще один пример. Основной файл (index.php) будет содержать требуемый код js, который будет формировать url-ссылку со значениями переменных и вызывать файл result.php.

```
<script language="javascript">
    var message = encodeURIComponent('Hello, server!');
    window.location.href = 'result.php?message='+message;
</script>
```

Файл result.php будет в минимальном варианте выглядеть следующим образом.

```
<?php
echo $_GET["message"];
?>
```



В последнем представленном примере был использован метод (JS) *encodeURIComponent*, который заменяет все символы, кроме: символов латинского алфавита, десятичных цифр и - _ . ! ~ * ' ().

```
var encoded = encodeURIComponent(str)
```

Соответственно str — это компонент URI

Это необходимо, чтобы избежать некорректных запросов к серверу. Поэтому метод *encodeURIComponent* следует вызывать на каждом введенном пользователем параметре, который будет передан как часть URI.

Например, пользователь может ввести "me&time=5" в переменной label. Без использования *encodeURIComponent* соответствующая компонента запроса будет иметь вид label=me&time=5. Вместо одного значения label появилось два значения: label=me и time=5.

Window.location Получает/устанавливает URL окна и его компоненты. Значением этого свойства является объект типа Location.

Объект Location

Метод *toString* этого объекта возвращает URL, а различные свойства позволяют получить/установить отдельные компоненты адреса.

Возможен иной способ. Пишем код JavaScript и выполняем его на стороне пользователя, а потом передаем результат через HTTP-протокол в PHP:

```
<script language="javascript"><!--
query='width=' + screen.width + '&height=' + screen.height;
//--></script>
```

Здесь было присвоено переменной query запрос со значениями ширины и высоты экрана пользователя, как и в предыдущем примере. В результате выполнения этого кода переменной query будет присвоена строка width=1920&height=1080 (при разрешении экрана 1920x1080).

Теперь нужно передать переменную query из JavaScript в PHP. Передавать запрос будем PHP-скрипту с именем script.php. Для этого воспользуемся HTML-тегом , который вставляет изображения в HTML-страницу. Вместо картинки мы укажем имя нашего PHP-скрипта и присоединим к нему переменную query с запросом:

```
<script language="JavaScript"><!--
document.write('');
//--></script>
```

Таким способом можно передавать запросы любому скрипту. Результат выполнения этих двух блоков JavaScript-кода браузер пользователя превратит в следующий HTML-код:

```

```

Браузер обратится к серверу за картинкой по указанному адресу и, в результате, сделает GET-запрос к script.php с нужными нам параметрами. Теперь мы можем обработать полученные переменные в нашем script.php:

```
<?php
// проверяем наличие переменных $width и $height
if (isset($_GET['width']) AND isset($_GET['height'])) {
    // Здесь пишем код, который выполнится, если переменные
    // $width и $height существуют. Их, например, можно записать в
    // текстовый файл или добавить в базу данных
}
else {
    // Здесь пишем код, который выполнится, если переменные не
    // существуют или можно вообще не использовать else {}
}
?>
```

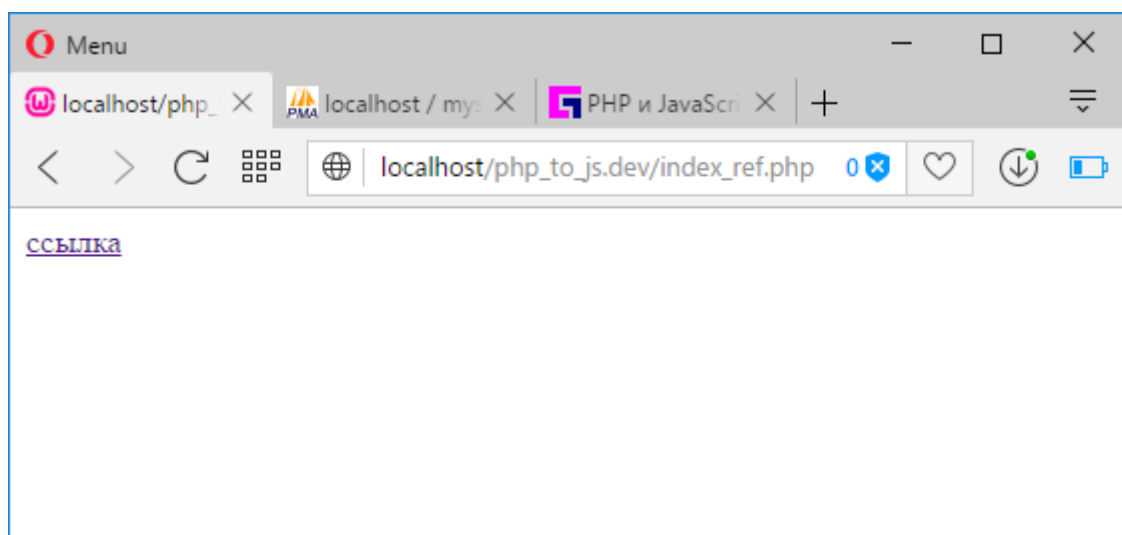
Например:

```
<?php
// проверяем наличие переменных $width и $height
if (isset($_GET['width']) AND isset($_GET['height'])) {
    echo 'ширина экрана = ' . $_GET['width'];
    echo "<br>";
    echo 'высота экрана = ' . $_GET['height'];
}
else {
    echo "Данные об размере экрана не получены";
}
?>
```

Данные также можно передавать PHP-скрипту после нажатия пользователем на ссылку:

```
<script language="javascript"><!--
query='width=' + screen.width + '&height=' + screen.height;
//--></script>

<script language="javascript"><!--
document.write("<a href=script.php?" + query + ">ссылка</a>");
--></script>
```

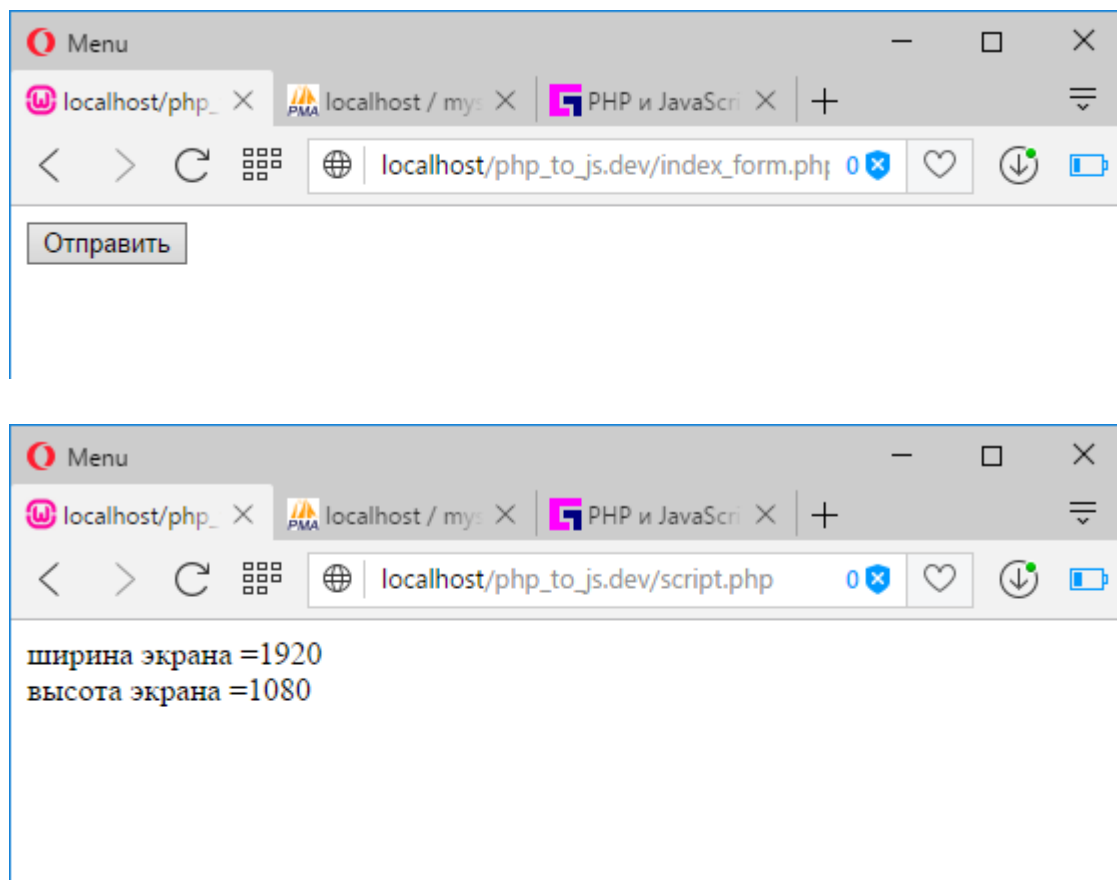


Также можно воспользоваться полями формы, например, `input type="hidden"` – так уже делалось при реализации регистрации. В таком случае необходимо лишь найти способ оправки формы (данных, хранящиеся в тех же скрытых полях будут считаны), например, по нажатию на кнопку.

```
<script language="javascript"><!--
width=screen.width;
height=screen.height;
//--></script>

<script language="javascript"><!--
document.write('<form name="form1" action="script.php" method="get">');
document.write('<input type="hidden" name="width" value="' + width + '
">');
document.write('<input type="hidden" name="height" value="' + height + '
">');
document.write('<input type="submit" value="Отправить"></form>');
//--></script>
```

В случае использования формы каждую переменную необходимо передавать в отдельном поле. В выше представленном скрипте были использованы скрытые поля формы, имена которых соответствуют именам переменных в PHP-скрипте. Файл script.php по сути остается неизменным и результаты передачи данных выводятся на экран.

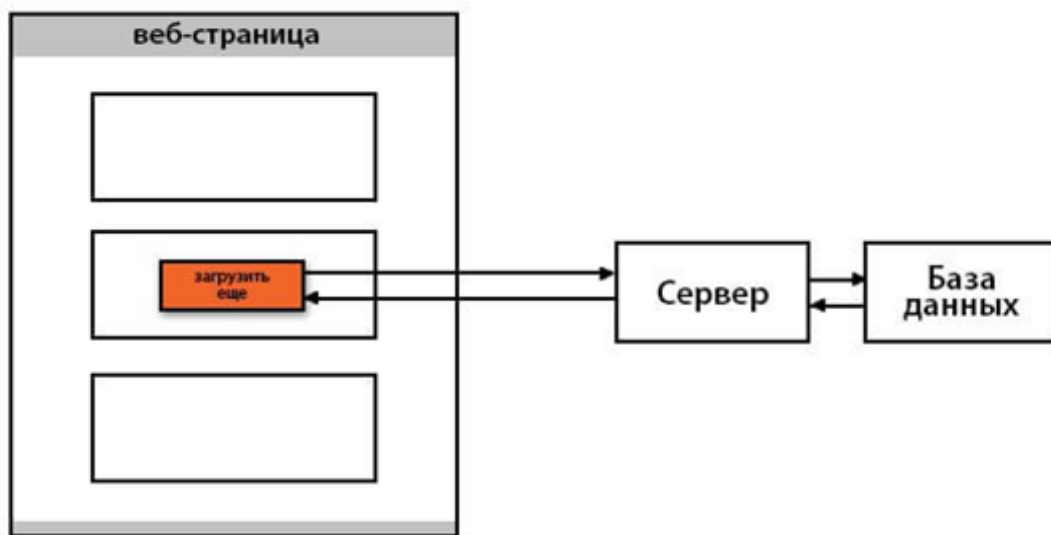


Отметим, что формально любой из этих способов можно использовать для того, чтобы передать значение из JavaScript не только в PHP, но и в Perl, ASP или в любой другой язык программирования, который выполняется на стороне веб-сервера.

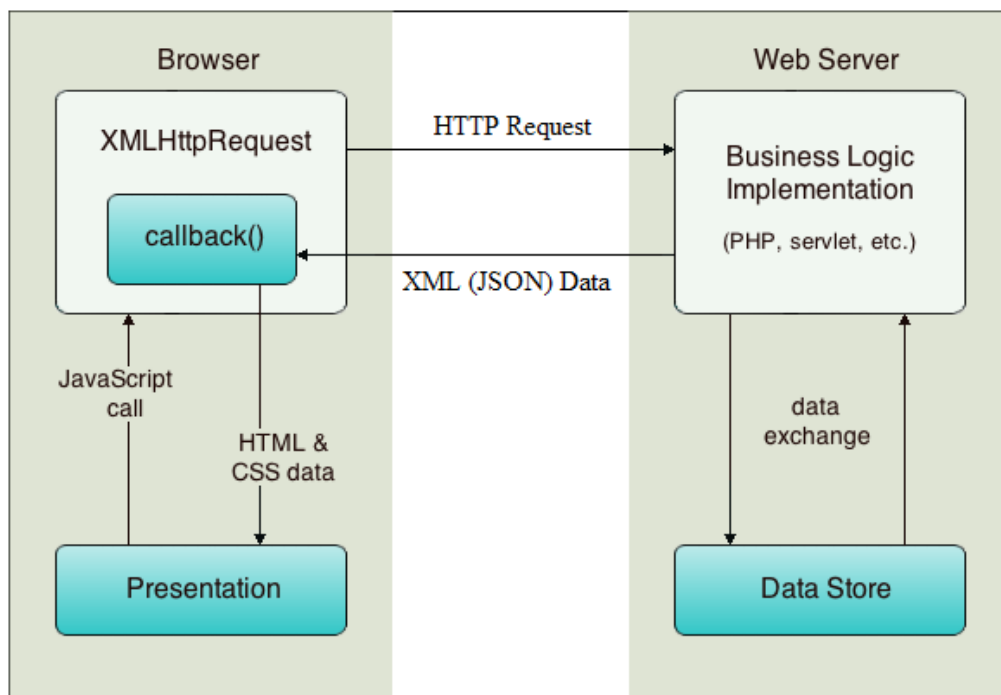
3. Асинхронный метод передачи данных из JavaScript в PHP (с использование ajax)

Асинхронная передача переменной из JavaScript в PHP подразумевает использование Ajax (страница браузера не перезагружается, а обновляется лишь ее фрагмент).

AJAX базируется на технологии обращения к серверу без перезагрузки страницы (XMLHttpRequest, создание дочерних фреймов или тега script) или использовании DHTML, позволяющего динамически изменять содержимое. Формат передачи данных – XML или JSON. В коде web-страниц широко используется JavaScript для прозрачного обмена данными клиента с сервером. Пользователи взаимодействуют со стандартными HTML элементами, динамическое поведение которых описывается на JavaScript.



Аjax использует объект XMLHttpRequest для асинхронной передачи запросов и ответов между клиентом и сервером. На следующем рисунке представлена блок-схема операций связи, происходящих между клиентом и сервером.



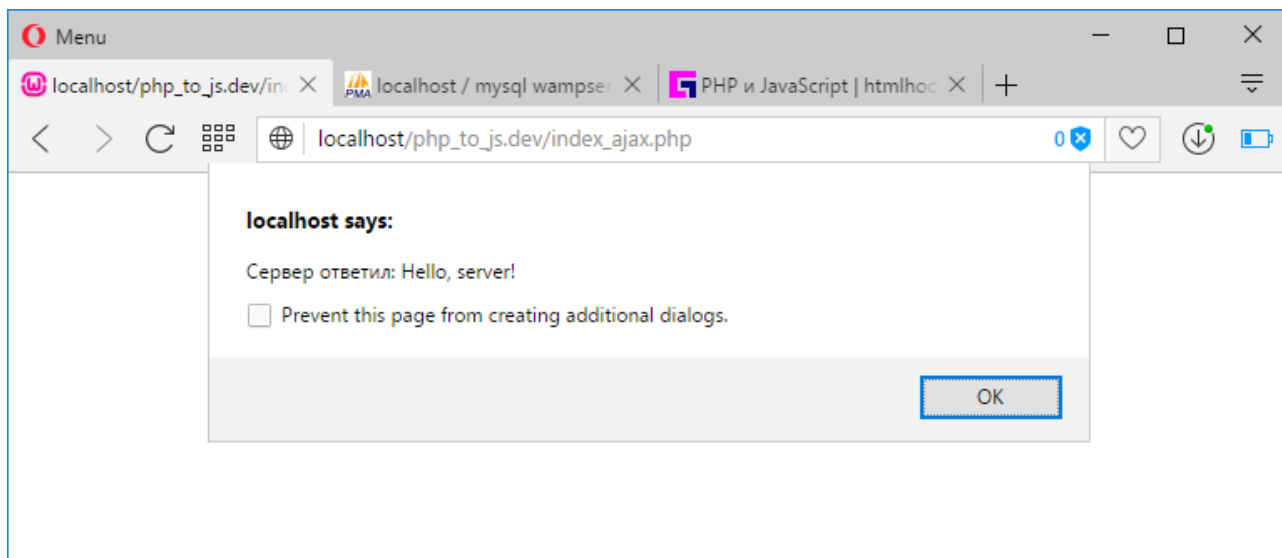
Проще всего реализовать асинхронную передачу данных, используя библиотеку jQuery. Рассмотрим на простейшем примере. Стартовый файл index_ajax.php

```
<script type="text/javascript" src="js/jquery-3.2.0.min.js"></script>
<script type="text/javascript">
  var message = 'Hello, server!';
  $.get('result.php', {message:message}, function(data) {
    alert('Сервер ответил: '+data);
  });
</script>
```

вызываемый файл result.php

```
<?php
    echo $_GET["message"];
?>
```

При такой реализации не нужно применять функцию `encodeURIComponent` - её применит jQuery. Необходимо только описать функцию, которая принимает ответ сервера и складывает его в переменную `data`. Если вызываемый файл (`result.php`) возвращает просто значение переменной `$_GET[message]`, то увидим окно `alert` с текстом Сервер ответил: Hello, server!



Для реализации технологии асинхронного обмена данными чаще всего используется метод `$.ajax` или `jQuery.ajax`:

`$.ajax(свойства)` или `$.ajax(url [, свойства])`

Второй параметр был добавлен в версии 1.5 jQuery.

url – адрес запрашиваемой страницы;

properties – свойства запроса.

Полный список параметров приведен в [документации jQuery](#).

Далее рассмотрим несколько наиболее часто используемых параметров.

success (функция) – данная функция вызывается после успешного завершения запроса.

Функция получает от 1 до 3 параметров (в зависимости от используемой версии библиотеки). Но первый параметр всегда содержит возвращаемые с сервера данные.

data (объект/строка) – пользовательские данные, которые передаются на запрашиваемую страницу.

dataType (строка) – возможные значения: `xml`, `json`, `script` или `html`. Описание типа данных, которые ожидаются в ответе сервера.

type (строка) – тип запроса. Возможные значения: `GET` или `POST`. По умолчанию: `GET`.

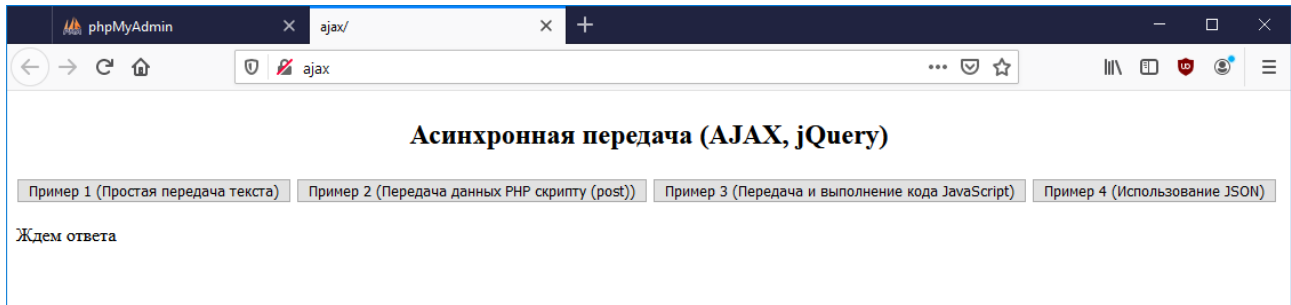
url (строка) – адрес URL для запроса.

Рассмотрим несколько примеров:

- простая передача текста;
- передача данных PHP скрипту (`post`);

- передача и выполнение кода JavaScript;
- использование для передачи JSON.

Рассмотрим следующий пример. Пусть имеется какая-то страница, на которой нужно обновить данные без перезагрузки страницы. В JavaScript с помощью jQuery делаем простой аякс-запрос к серверу и выводим данные в браузер. Общий вид окна примера пусть будет следующим:



Т.е. на основной странице размещаются 4 кнопки, каждая из которых демонстрирует отдельный пример.

```
<button name="sample1" class="sample1">Пример 1 (Простая передача
текста)</button>
<button name="sample2" class="sample2">Пример 2 (Передача данных PHP
скрипту (post))</button>
<button name="sample3" class="sample3">Пример 3 (Передача и выполнение
кода JavaScript)</button>
<button name="sample4" class="sample4">Пример 4 (Использование
JSON)</button>
```

Результаты выполнения асинхронного запроса будем выводить в отдельный div.

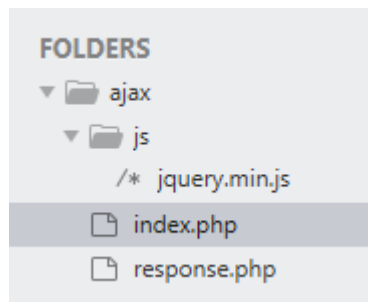
```
<div class="results">Ждем ответа</div>
```

Асинхронный запрос реализуется посредством специальной функции.

```
$.ajax({
    url: 'response.php',
    success: function(data) {
        $('<div class="results">Ждем ответа</div>').html(data);
    }
});
```

Внешний вид примера представлен на следующем скриншоте

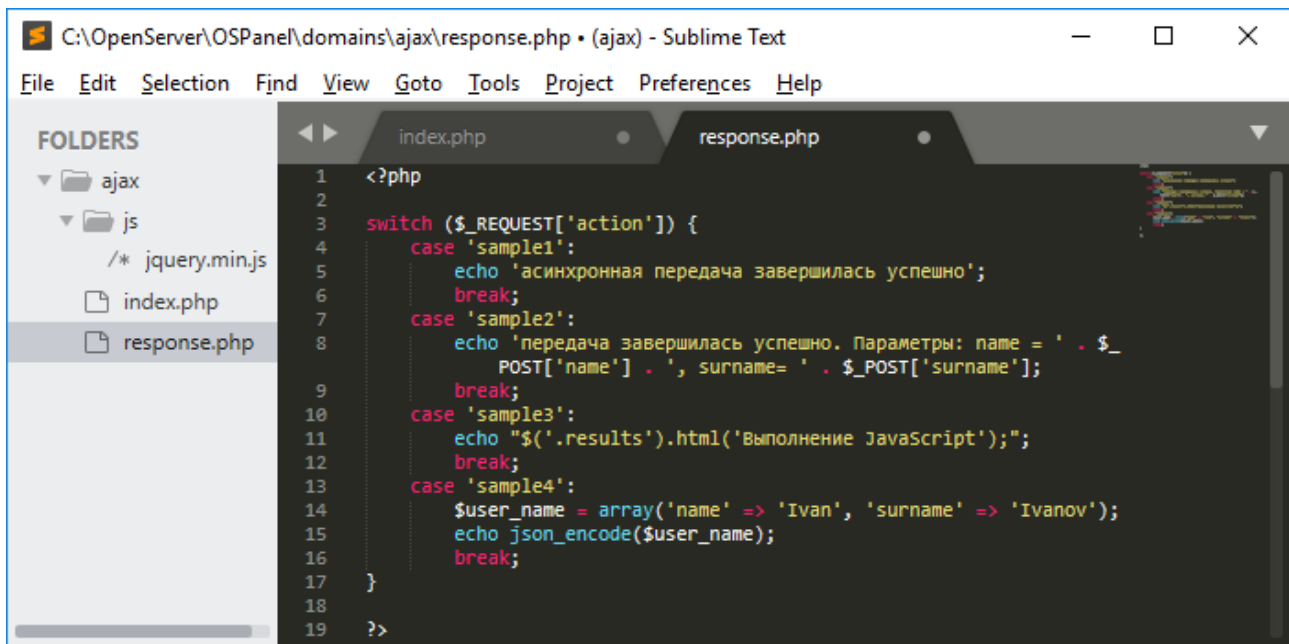
Общая структура проекта с примерами использования AJAX будет следующей.



Файл `index.php`, он же стартовый файл, отвечает за отображение соответствующих кнопок для выполнения асинхронного запроса, точнее вызова соответствующих функций, которые также содержатся в данном файле.

A screenshot of a Sublime Text editor window. The title bar shows the file path: `C:\OpenServer\OSPanel\domains\ajax\index.php`. The editor has a menu bar with 'File', 'Edit', 'Selection', 'Find', 'View', 'Goto', 'Tools', 'Project', 'Preferences', and 'Help'. On the left, there's a 'FOLDERS' sidebar showing the same directory structure as the first image. The main editor area shows the code for `index.php`. The code starts with a DOCTYPE declaration and an HTML lang attribute. It includes a meta charset and a script tag for jQuery. The body contains a header with a title 'Асинхронная передача (AJAX, jQuery)' and a div with four buttons. Each button has a unique class and name. Below the buttons, there's a script block with four click event handlers. Each handler calls `$.ajax()` with specific parameters (url, type, data, dataType) and success callbacks. The first handler calls `response.php?action=sample1`, the second `response.php?action=sample2` with POST data, the third `response.php?action=sample3` with dataType 'script', and the fourth `response.php?action=sample4` with dataType 'json'. The script ends with a closing tag for the body and html. The `response.php` file is also open in the editor, but its content is not visible in this screenshot.

Файл `response.php` отвечает за отображение результатов асинхронного запроса.



```
1 <?php
2
3 switch ($_REQUEST['action']) {
4     case 'sample1':
5         echo 'асинхронная передача завершилась успешно';
6         break;
7     case 'sample2':
8         echo 'передача завершилась успешно. Параметры: name = ' . $_
9             POST['name'] . ', surname= ' . $_POST['surname'];
10        break;
11     case 'sample3':
12         echo "($_.results').html('Выполнение JavaScript');";
13         break;
14     case 'sample4':
15         $user_name = array('name' => 'Ivan', 'surname' => 'Ivanov');
16         echo json_encode($user_name);
17         break;
18 }
19 ?>
```

Пример 1

Для простой передачи данных обычно используют функцию со следующей структурой.

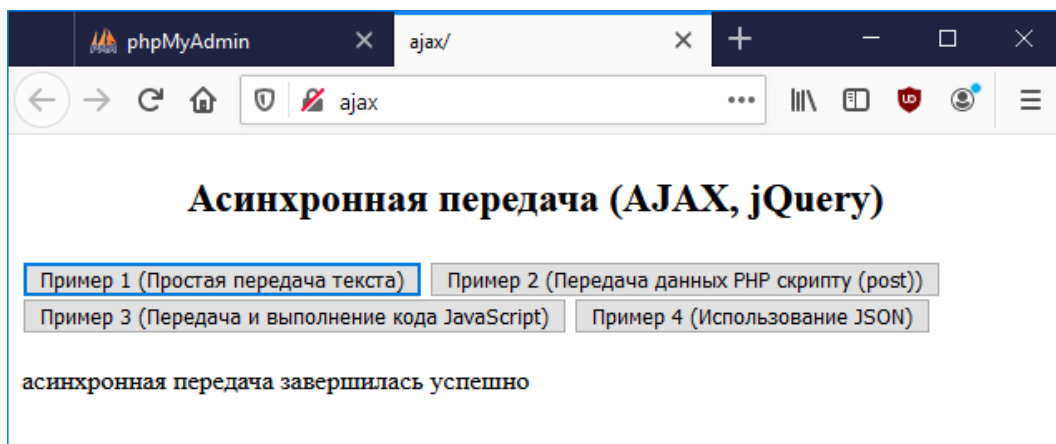
```
<script language="javascript" type="text/javascript">
    $('sample1').click( function() {
        $.ajax({
            url: 'response.php?action=sample1',
            success: function(data) {
                $('results').html(data);
            }
        });
    });
</script>
```

Для ответа, как уже отмечалось, будем использовать элемент div .result

```
<div class="results">Ждем ответа</div>
```

Сервер просто возвращает соответствующую строку

```
echo 'асинхронная передача завершилась успешно';
```



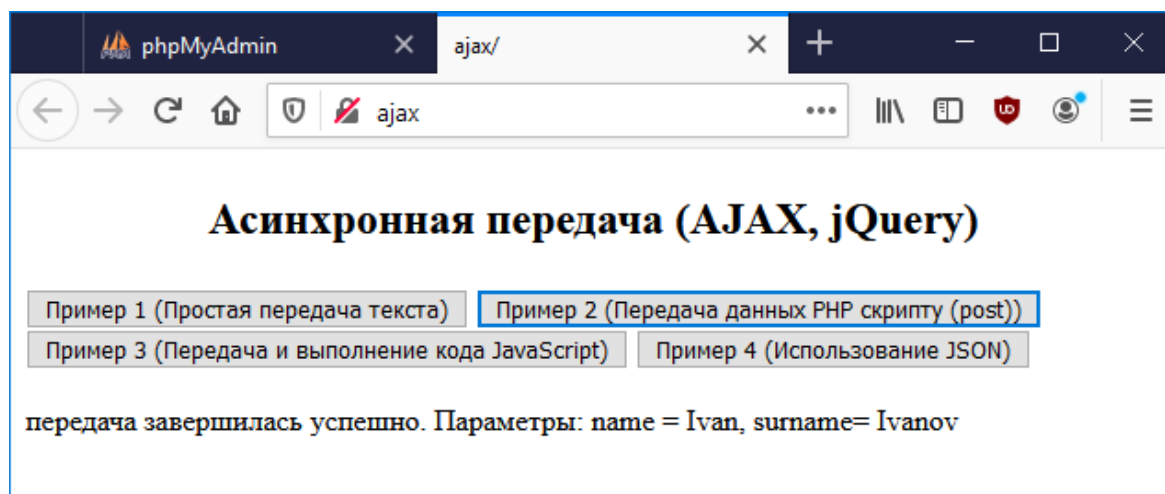
Пример 2

Для передачи данных PHP скрипту, например, методом post, обычно используют функцию со следующей структурой.

```
<script language="javascript" type="text/javascript">
    $('.sample2').click( function() {
        $.ajax({
            type: 'POST',
            url: 'response.php?action=sample2',
            data: 'name=Ivan&surname=Ivanov',
            success: function(data){
                $('.results').html(data);
            }
        });
    });
</script>
```

Сервер возвращает строку со вставленными в нее переданными данными.

```
echo 'передача завершилась успешно. Параметры: name = ' .
$_POST['name'] . ', surname= ' . $_POST['surname'];
```



Пример 3

Для передачи данных и выполнения кода JavaScript обычно используют функцию со следующей структурой.

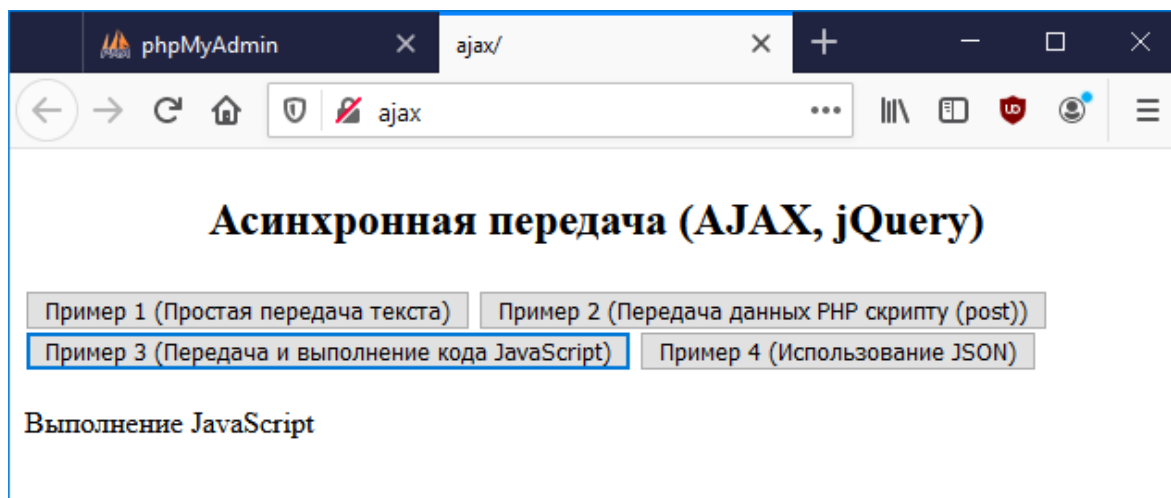
```
<script language="javascript" type="text/javascript">
    $('.sample3').click( function() {

        $.ajax({
            dataType: 'script',
            url: 'response.php?action=sample3',
        });

    });
</script>
```

Сервер выполняет код.

```
echo "$('.results').html('Выполнение JavaScript');";
```



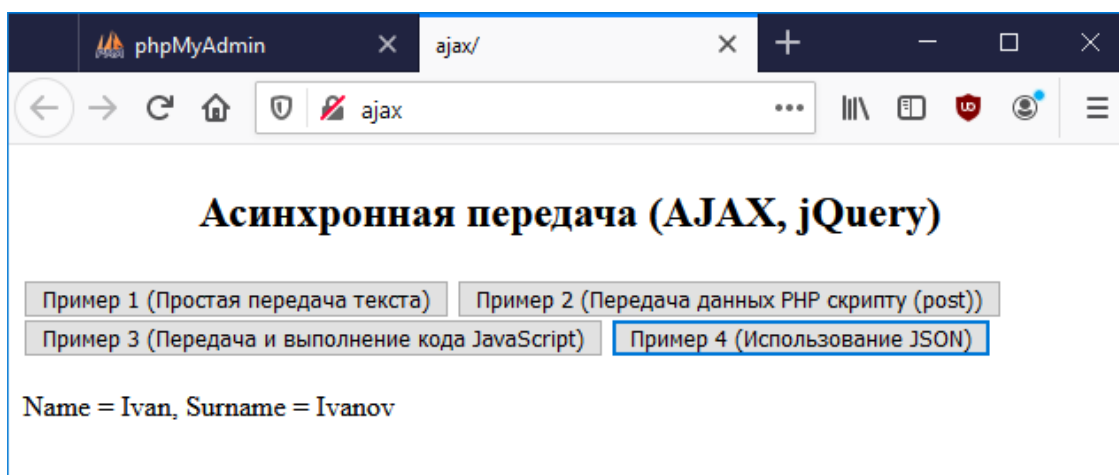
Пример 4

Для передачи данных с использованием формата JSON обычно используют функцию со следующей структурой.

```
$('.sample4').click( function() {  
    $.ajax({  
        dataType: 'json',  
        url: 'response.php?action=sample4',  
        success: function(jsondata){  
            alert('Data tranfer is OK!');  
            $('.results').html('Name = ' + jsondata.name + ', Surname = '  
+ jsondata.surname);  
        }  
    });  
});
```

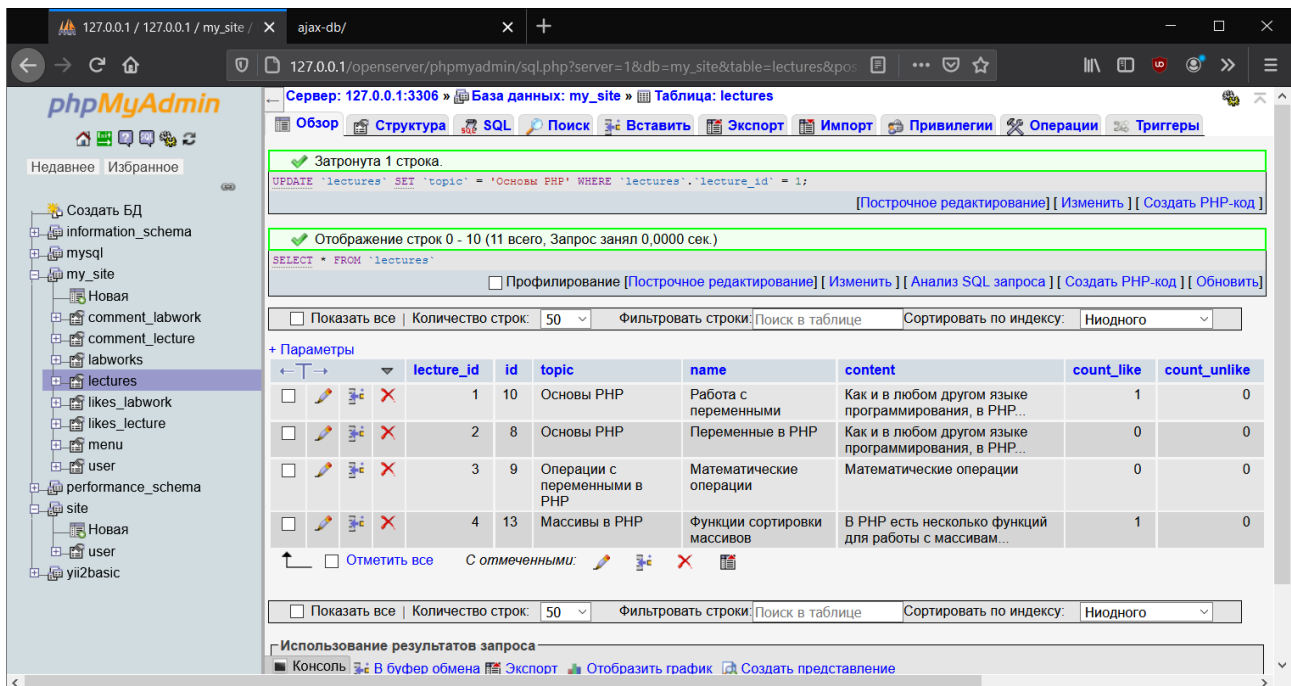
Сервер должен возвращать данные в формате JSON.

```
$user_name = array('name' => 'Ivan', 'surname' => 'Ivanov');  
echo json_encode($user_name);
```



Рассмотрим еще один пример, позволяющий извлекать асинхронно и БД некоторую информацию и выводить ее в соответствующий div.

Пусть имеется в БД таблица со следующей структурой и содержанием – таблица lectures в БД my_site.

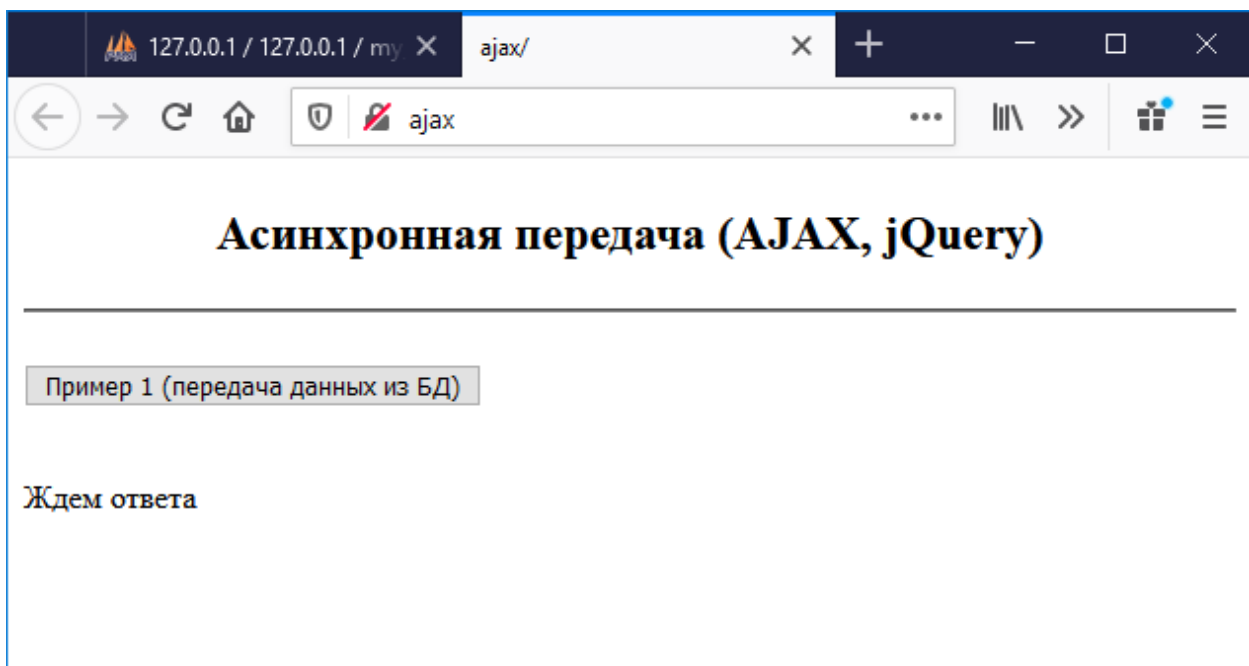


The screenshot shows the phpMyAdmin interface for the 'my_site' database. The 'lectures' table is selected, and its structure and data are displayed. The table has columns: lecture_id, id, topic, name, content, count_like, and count_unlike. The data is as follows:

lecture_id	id	topic	name	content	count_like	count_unlike
1	10	Основы PHP	Работа с переменными	Как и в любом другом языке программирования, в PHP...	1	0
2	8	Основы PHP	Переменные в PHP	Как и в любом другом языке программирования, в PHP...	0	0
3	9	Операции с переменными в PHP	Математические операции	Математические операции	0	0
4	13	Массивы в PHP	Функции сортировки массивов	В PHP есть несколько функций для работы с массивам...	1	0

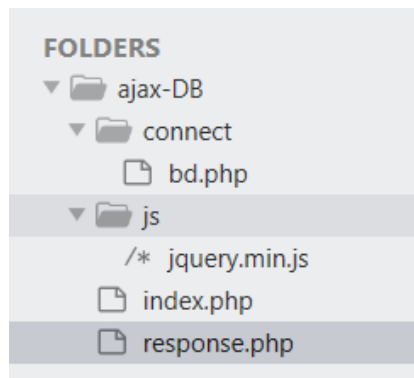
Пусть необходимо в соответствии с указанным пользователем критерии, например, условно lecture_id, вывести в div информацию о теме, названии и содержании лекции, соответствующей введенному (переданному) id.

Внешний вид примера представлен на следующем скриншоте.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1 / 127.0.0.1 / my' and 'ajax/'. The main content area has a title 'Асинхронная передача (AJAX, jQuery)' and a section 'Пример 1 (передача данных из БД)'. Below this section, the text 'Ждем ответа' is visible.

Общая структура проекта с примерами использования AJAX будет следующей.

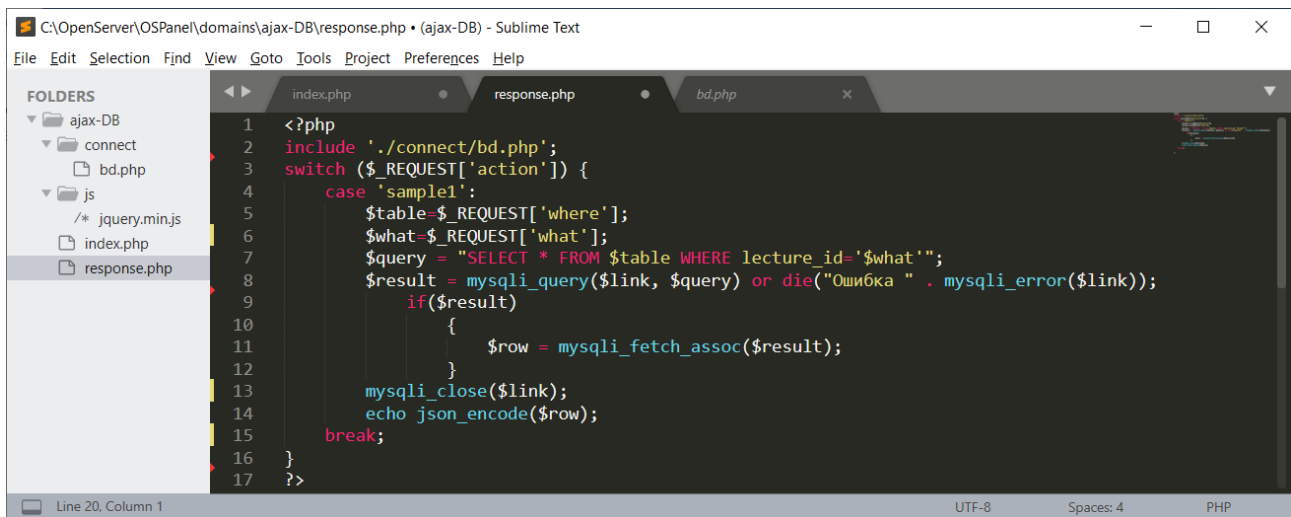


Файл `index.php`, он же стартовый файл, отвечает за отображение соответствующей кнопки для выполнения асинхронного запроса, точнее вызова соответствующих функций, которые также содержатся в данном файле.

```
1 <!DOCTYPE html>
2 <html lang="ru" >
3   <head>
4     <meta charset="utf-8" />
5     <script src="js/jquery.min.js" type="text/javascript"></script>
6   </head>
7   <body>
8     <header>
9       <h2><p align='center'>Асинхронная передача (AJAX, jQuery)</p></h2><hr><BR>
10    </header>
11    <div class="examples">
12      <button name="sample1" class="sample1">Пример 1 (передача данных из БД)</button>
13      <br><br><br>
14      <script language="javascript" type="text/javascript">
15        $('.sample1').click( function() {
16
17          $.ajax({
18            dataType: 'json',
19            url: 'response.php?action=sample1&where=lectures&what=1',
20            success: function(jsondata){
21              alert('Data tranfer is OK!');
22              $('.results').html('<b>Тема лекции:</b> ' + jsondata.topic + '<br><b>
                Название лекции:</b> ' + jsondata.name+ '<br><b>Материалы лекции:</b> '
                + jsondata.content);
23            }
24          });
25        });
26      </script>
27      <div class="results">Ждем ответа</div>
28    </div>
29  </body>
30 </html>
```

Файл `response.php` отвечает за отображение результатов асинхронного запроса, а за подключение к БД отвечает файл `bd.php`.

```
1 <?php
2 $host = "localhost"; // адрес сервера
3 $database = "my_site"; // имя базы данных
4 $user = "root"; // имя пользователя
5 $user_password = ""; // пароль
6 $link = mysqli_connect($host, $user, $user_password, $database) or die("Ошибка
    подключения к БД" . mysqli_error($link));
7 mysqli_set_charset($link, "utf8");
8 ?>
```



```
1 <?php
2 include './connect/bd.php';
3 switch ($_REQUEST['action']) {
4     case 'sample1':
5         $table=$_REQUEST['where'];
6         $what=$_REQUEST['what'];
7         $query = "SELECT * FROM $table WHERE lecture_id='$what'";
8         $result = mysqli_query($link, $query) or die("Ошибка " . mysqli_error($link));
9         if($result)
10         {
11             $row = mysqli_fetch_assoc($result);
12         }
13         mysqli_close($link);
14         echo json_encode($row);
15         break;
16     }
17 }
```

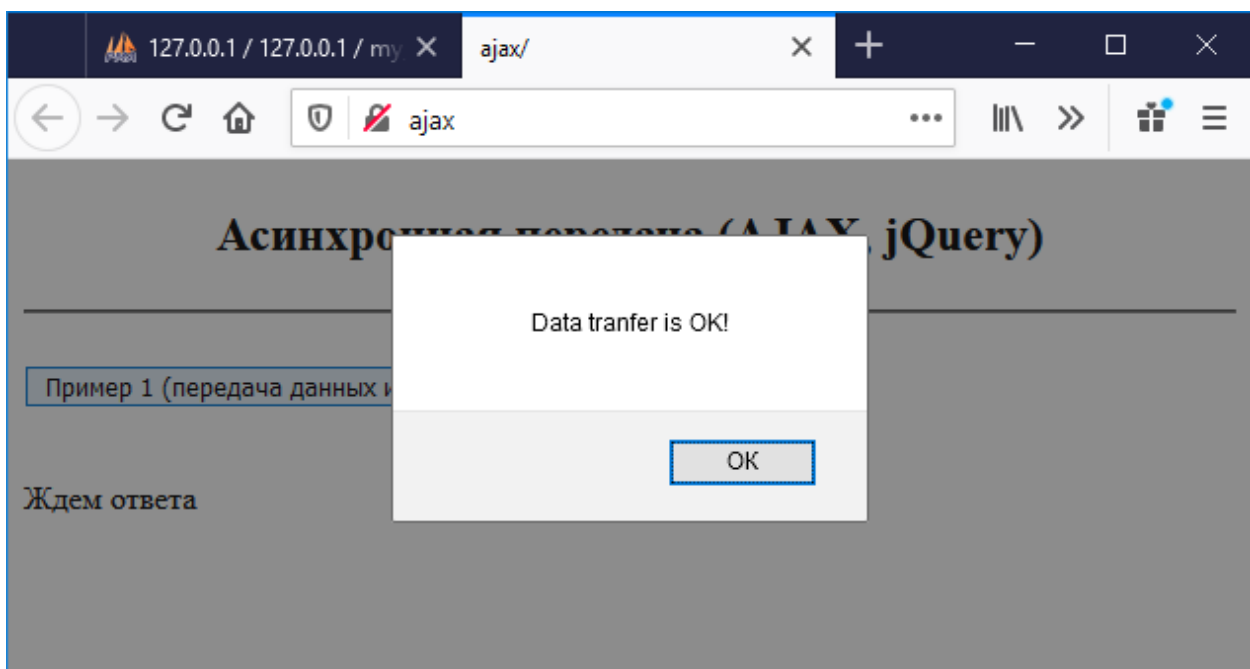
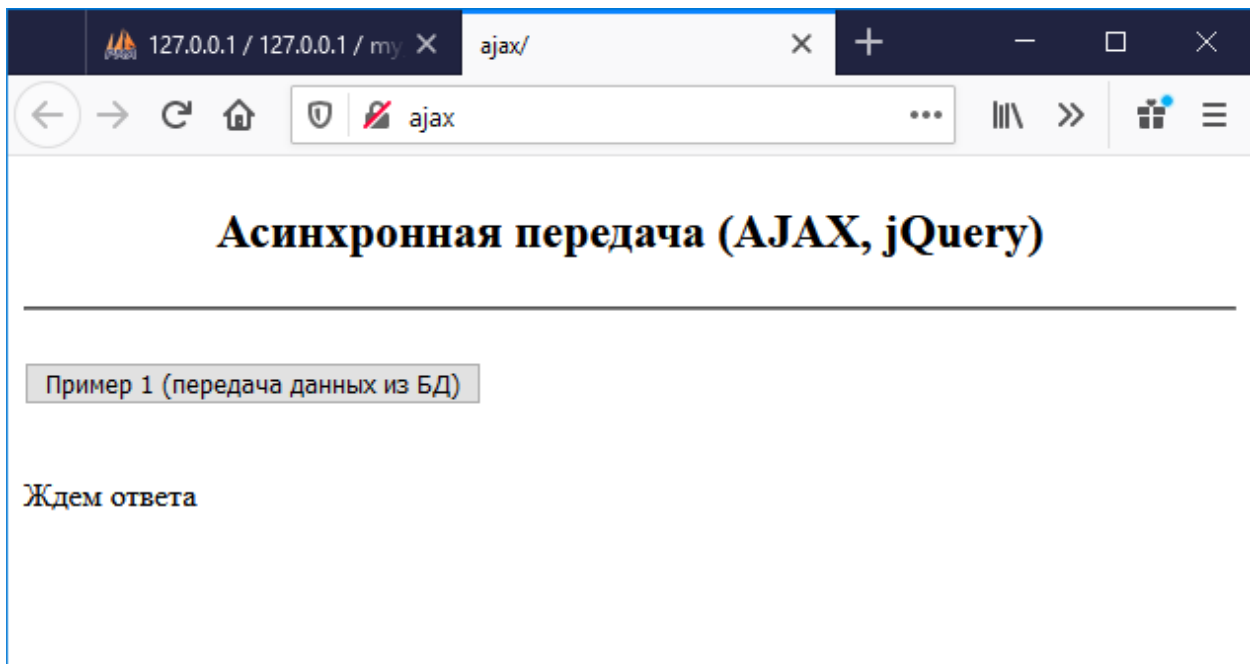
Передачу параметров в файл response.php из ajax-функции, необходимых для запроса к БД, будем осуществлять через передачу переменных по URL-строке. Для передачи данных, извлеченных из БД, будем использовать формат JSON. В таком случае ajax-функция будет иметь, например, такую структуру.

```
<script language="javascript" type="text/javascript">
    $('sample1').click( function() {
        $.ajax({
            dataType: 'json',
            url: 'response.php?action=sample1&where=lectures&what=1',
            success: function(jsondata){
                alert('Data tranfer is OK!');
                $('results').html('<b>Тема лекции:</b> ' + jsondata.topic +
                '<br><b>Название лекции:</b> ' + jsondata.name+ '<br><b>Материалы
                лекции:</b> ' + jsondata.content);
            }
        });
    });
</script>
```

Сервер должен извлекать данные из БД и возвращать их в формате JSON.

```
include './connect/bd.php';
switch ($_REQUEST['action']) {
    case 'sample1':
        $table=$_REQUEST['where'];
        $what=$_REQUEST['what'];
        $query = "SELECT * FROM $table WHERE lecture_id='$what'";
        $result = mysqli_query($link, $query) or die("Ошибка " .
        mysqli_error($link));
        if($result){
            $row = mysqli_fetch_assoc($result);}
        mysqli_close($link);
        echo json_encode($row);
        break;
}
```


В результате получим следующее, при условии, что необходимо вывести материалы лекции с `lecture_id=1` из таблицы `lectures` (в ajax-функции `url: 'response.php?action=sample1&where=lectures&what=1'`).



127.0.0.1 / 127.0.0.1 / my_site / X ajax-db/ X +

ajax-db

Асинхронная передача (AJAX, jQuery)

Пример 1 (передача данных из БД)

Тема лекции: Основы PHP

Название Лекции: Работа с переменными

Материалы лекции: Как и в любом другом языке программирования, в PHP существует такое понятие, как переменная. При программировании на PHP можно не скупиться на объявление новых переменных. Принципы экономии памяти, которые были актуальны несколько лет назад, сегодня в расчет не принимаются. Однако, при хранении в переменных больших объемов памяти, лучше удалять неиспользуемые переменные, используя оператор `unset`. Вообще, переменная - это область оперативной памяти, доступ к которой осуществляется по имени. Все данные, с которыми работает программа, хранятся в виде переменных (исключение — константа, которая, впрочем, может содержать только число или строку). Такого понятия, как указатель (как в C++), в PHP не существует — при присвоении переменная копируется один-в-один, какую бы сложную структуру она ни имела. Тем не менее, в PHP, начиная с версии 4, существует понятие ссылок — жестких и символических. Имена всех переменных в PHP должны начинаться со знака `$` — так интерпретатору значительно легче "понять" и отличить их, например, в строках. Имена переменных чувствительны к регистру букв: например, `$var` — не то же самое, что `$Var` или `$VAR`: В официальной документации PHP указано, что имя переменной может состоять не только из букв "Латиницы" и цифр, но также и из любых символов, код ASCII которых старше 127, — в частности, и из символов кириллицы, то есть "русских" букв! Однако не рекомендуется применять кириллицу в именах переменных — хотя бы из-за того, что в различных кодировках ее буквы имеют различные коды. Впрочем, можно поэкспериментировать и делать так, как вам будет удобно. Можно сказать, что переменные в PHP — это особые объекты, которые могут содержать в буквальном смысле все, что угодно.

4. Передача сложных объектов из PHP в JavaScript

Формат JSON

JSON - простой, основанный на использовании текста, способ хранить и передавать структурированные данные. С помощью простого синтаксиса вы можете легко хранить все, что угодно, начиная от одного числа до строк, массивов и объектов, в простом тексте. Также можно связывать между собой массивы и объекты, создавая сложные структуры данных.

После создания строки JSON, ее легко отправить другому приложению или в другое место сети, так как она представляет собой простой текст.

JSON имеет следующие преимущества:

- Он компактен.
- Его предложения легко читаются и составляются как человеком, так и компьютером.
- Его легко преобразовать в структуру данных для большинства языков программирования (числа, строки, логические переменные, массивы и так далее)
- Многие языки программирования имеют функции и библиотеки для чтения и создания структур JSON.

Название JSON означает JavaScript Object Notation (представление объектов JavaScript). Как и представляет имя, он основан на способе определения объектов (очень похоже на создание ассоциативных массивов в других языках) и массивов.

Наиболее частое распространенное использование JSON - пересылка данных от сервера к браузеру. Обычно данные JSON доставляются с помощью AJAX, который позволяет обмениваться данными браузеру и серверу без необходимости перезагружать страницу.

Также можно использовать JSON для отправки данных от браузера на сервер, передавая строку JSON в качестве параметра запросов GET или POST.

Данные в формате JSON (RFC 4627) представляют собой:

- JavaScript-объекты { ... } или
- Массивы [...] или
- Значения одного из типов:
- строки в двойных кавычках,
- число,
- логическое значение true/false,
- null.

Почти все языки программирования имеют библиотеки для преобразования объектов в формат JSON.

Есть несколько основных правил для создания строки JSON:

- Строка JSON содержит либо массив значений, либо объект (ассоциативный массив пар имя/значение).
- Массив заключается в квадратные скобки ([и]) и содержит разделенный запятой список значений.
- Объект заключается в фигурные скобки ({ и }) и содержит разделенный запятой список пар имя/значение.
- Пара имя/значение состоит из имени поля, заключенного в двойные кавычки, за которым следует двоеточие (:) и значение поля.

Значение в массиве или объекте может быть:

- Числом (целым или с плавающей точкой)
- Строкой (в двойных кавычках)
- Логическим значением (true или false)
- Другим массивом (заключенным в квадратные скобки)
- Другой объектом (заключенный в фигурные скобки)
- Значение null

Чтобы включить двойные кавычки в строку, нужно использовать обратную косую черту: \". Так же, как и во многих языках программирования, можно помещать управляющие символы и шестнадцатеричные коды в строку, предваряя их обратной косой чертой. Смотрите детали на сайте JSON.

Синтаксис JSON

Формат json обычно записывается в 2-х вариантах:

1. Последовательность значений. Например, последовательность 10, 15 и "test" в формате JSON будут выглядеть так:

```
[10,15,"test"]
```

2. Запись в виде пар **ключ : значение**. Например:

```
{"ФИО":"Иванов Сергей", "Дата рождения":"09.03.1975"}
```

Немного более сложный пример:

```
{
  "ФИО" : "Иванов Иван",
  "Адрес" : {
    "Город" : "Минск",
    "Улица" : "Маяковского",
    "Дом" : "35"
  }
}
```

Работа с JSON на JavaScript

Основные методы для работы с данными в формате JSON в JavaScript – это:

- `JSON.parse` – читает объекты из строки в формате JSON.
- `JSON.stringify` – превращает объекты в строку в формате JSON, используется, когда нужно из JavaScript передать данные по сети.

Метод `JSON.stringify` (Создание строки JSON из переменной)

JavaScript имеет встроенный метод `JSON.stringify()`, который берет переменную и возвращает строку JSON, представляющую ее содержание.

Пример преобразования (сериализации) объекта в JSON-строку:

```
obj = {
  "name": "Вася",
  "age": 35,
  "isAdmin": false
}
alert( JSON.stringify(obj) ); // выведет {"name":"Вася","age":35,"isAdmin":false}
```

При сериализации (преобразовании) объекта в JSON-строку, вызывается метод **toJSON** этого объекта, если он существует. Если метода нет, тогда перечисляются все свойства объекта. Пример преобразования объекта с методом `toJSON`:

```
obj = {
  "name": "Вася",
  "age": 35,
  "isAdmin": false,
  toJSON: function() {
    return this.age;
  }
}
alert( JSON.stringify(obj) ); // выведет 35
```

Также необходимо отметить, что метод `JSON.stringify()` возвращает строку JSON без пробелов. Ее сложнее читать, но зато она более компактна для передачи через сеть.

Метод `JSON.parse` (Создание переменной из строки JSON)

Существует несколько способов разобрать строку JSON в JavaScript, но самый безопасный и надежный - использовать встроенный метод `JSON.parse()`. Он получает строку JSON и возвращает объект или массив JavaScript, который содержит данные. Например:

Простой пример декодирования json-строки в массив с цифрами:

```
str = "[0, 1, 2, 3]";
arr = JSON.parse(str);
alert( arr[1] ); // выведет 1
```

Обе функции `JSON.parse` и `JSON.stringify` имеют дополнительные параметры для уточнения правил преобразований. К тому же `JSON.stringify()` и `JSON.parse()` имеют другие возможности, такие как использование возвратных функций для пользовательской конвертации определённых данных. Такие опции очень удобны для конвертации различных данных в правильные объекты JavaScript.

Работа со строкой JSON в PHP

В языке php начиная с версии 5.2. есть всего четыре функции:

json_decode — Декодирует строку JSON (из строки json-формата получает данные)

json_encode — Возвращает JSON-представление данных (преобразует данные в json-строку)

json_last_error_msg — Возвращает строку с сообщением об ошибке последнего вызова json_encode() или json_decode()

json_last_error — Возвращает последнюю ошибку

Создание строки JSON из переменной PHP

Функция json_encode() принимает переменную PHP и возвращает строку JSON, представляющую содержание переменной value.

```
json_encode ( mixed $value [, int $options = 0 [, int $depth = 512 ]] ) : string
```

На кодирование влияет параметр options и, кроме того, кодирование значений типа float зависит от значения serialize_precision.

Список параметров ¶

value

value - значение, которое будет закодировано. Может быть любого типа, кроме resource.

Функция работает только с кодировкой UTF-8.

options

Битовая маска, составляемая из значений JSON_FORCE_OBJECT, JSON_HEX_QUOT, JSON_HEX_TAG, JSON_HEX_AMP, JSON_HEX_APOS, JSON_INVALID_UTF8_IGNORE, JSON_INVALID_UTF8_SUBSTITUTE, JSON_NUMERIC_CHECK, JSON_PARTIAL_OUTPUT_ON_ERROR, JSON_PRESERVE_ZEROS_FRACTION, JSON_PRETTY_PRINT, JSON_UNESCAPED_LINE_TERMINATORS, JSON_UNESCAPED_SLASHES, JSON_UNESCAPED_UNICODE, JSON_THROW_ON_ERROR.

depth

Устанавливает максимальную глубину. Должен быть больше нуля.

Возвращаемые значения ¶

Возвращает строку (string), закодированную JSON или FALSE в случае возникновения ошибки.

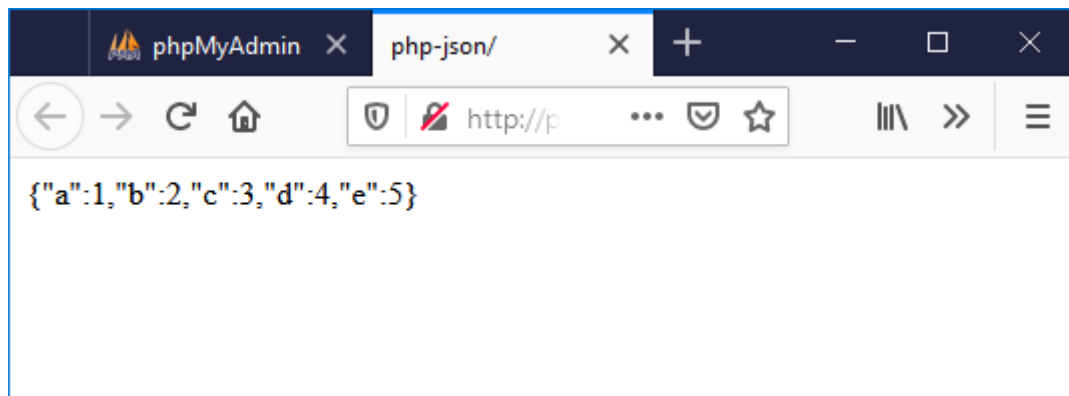
Пример №1 использования json_encode()

```
<?php
$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);

echo json_encode($arr);

?>
```

Результат выполнения данного примера:



Пример №2 использования json_encode() с опциями

```
<?php
$a = array('<foo>', "'bar'", '"baz"', '&blong&', "\xc3\xa9");

echo "Обычно: ", json_encode($a);
echo "<br>";
echo "Теги: ", json_encode($a, JSON_HEX_TAG);
echo "<br>";
echo "Апострофы: ", json_encode($a, JSON_HEX_APOS);
echo "<br>";
echo "Кавычки: ", json_encode($a, JSON_HEX_QUOT);
echo "<br>";
echo "Амперсанды: ", json_encode($a, JSON_HEX_AMP);
echo "<br>";
echo "Юникод: ", json_encode($a, JSON_UNESCAPED_UNICODE);
echo "<br>";
echo "Все: ", json_encode($a, JSON_HEX_TAG | JSON_HEX_APOS |
JSON_HEX_QUOT | JSON_HEX_AMP | JSON_UNESCAPED_UNICODE);
echo "<br>";
echo "<br>";

$b = array();

echo "Отображение пустого массива как массива: ", json_encode($b);
echo "<br>";
echo "Отображение неассоциативного массива как объекта: ",
json_encode($b, JSON_FORCE_OBJECT);
echo "<br>";
echo "<br>";

$c = array(array(1,2,3));

echo "Отображение неассоциативного массива как массива: ",
json_encode($c);
echo "<br>";
echo "Отображение неассоциативного массива как объекта: ",
json_encode($c, JSON_FORCE_OBJECT);
```

```

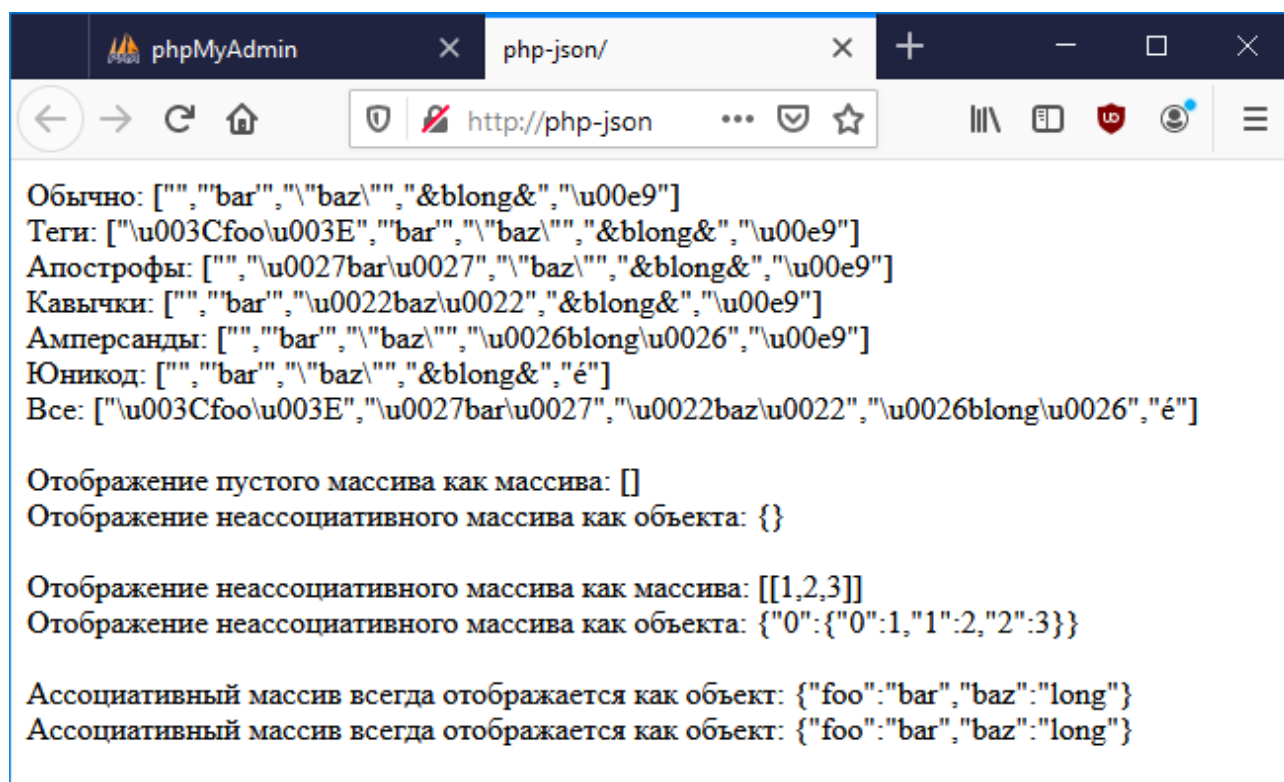
echo "<br>";
echo "<br>";

$d = array('foo' => 'bar', 'baz' => 'long');

echo "Ассоциативный массив всегда отображается как объект: ",
json_encode($d);
echo "<br>";
echo "Ассоциативный массив всегда отображается как объект: ",
json_encode($d, JSON_FORCE_OBJECT);
?>

```

Результат выполнения данного примера:



Создаем переменную из строки JSON

Для преобразования строки JSON в переменную PHP используется метод `json_decode()`.

```

json_decode ( string $json [, bool $assoc = FALSE [, int $depth =
512 [, int $options=0 ]]] ) : mixed

```

Принимает закодированную в JSON строку и преобразует ее в переменную PHP.

Список параметров ¶

`json`

Строка (string) `json` для декодирования.

Эта функция работает только со строками в кодировке UTF-8.

`assoc`

Если TRUE, возвращаемые объекты будут преобразованы в ассоциативные массивы.

`depth`

Указывает глубину рекурсии.

options

Битовая маска из констант JSON_BIGINT_AS_STRING, JSON_INVALID_UTF8_IGNORE, JSON_INVALID_UTF8_SUBSTITUTE, JSON_OBJECT_AS_ARRAY, JSON_THROW_ON_ERROR. Поведение этих констант описаны в конце лекции.

Возвращаемые значения ¶

Возвращает данные json, преобразованные в соответствующие типы PHP. Значения true, false и null возвращаются как TRUE, FALSE и NULL соответственно. NULL также возвращается, если json не может быть преобразован или закодированные данные содержат вложенных уровней больше, чем допустимый предел для рекурсий.

Пример №1 использования json_decode()

```
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

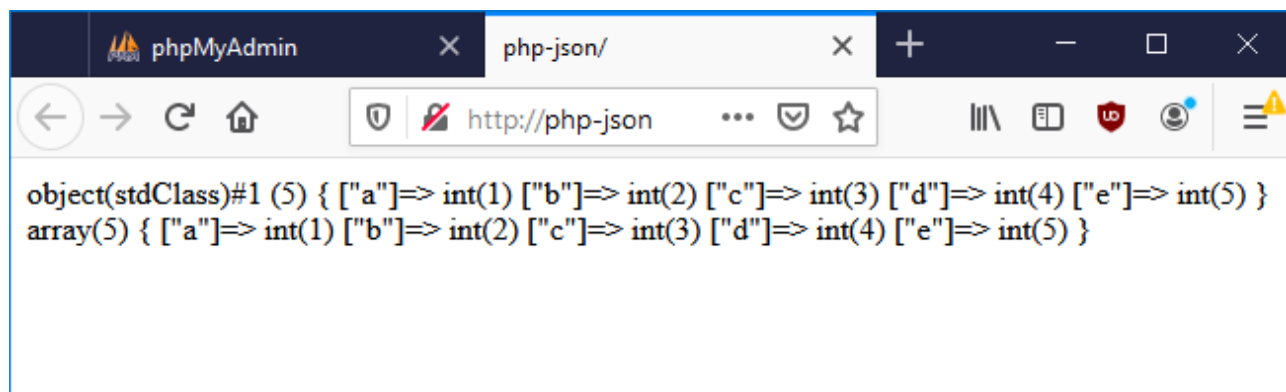
var_dump(json_decode($json));
var_dump(json_decode($json, true));

?>
```

Результат выполнения данного примера:

```
object(stdClass)#1 (5) {
    ["a"] => int(1)
    ["b"] => int(2)
    ["c"] => int(3)
    ["d"] => int(4)
    ["e"] => int(5)
}

array(5) {
    ["a"] => int(1)
    ["b"] => int(2)
    ["c"] => int(3)
    ["d"] => int(4)
    ["e"] => int(5)
}
```



Пример #2 Доступ к свойствам объектов с неправильными именами

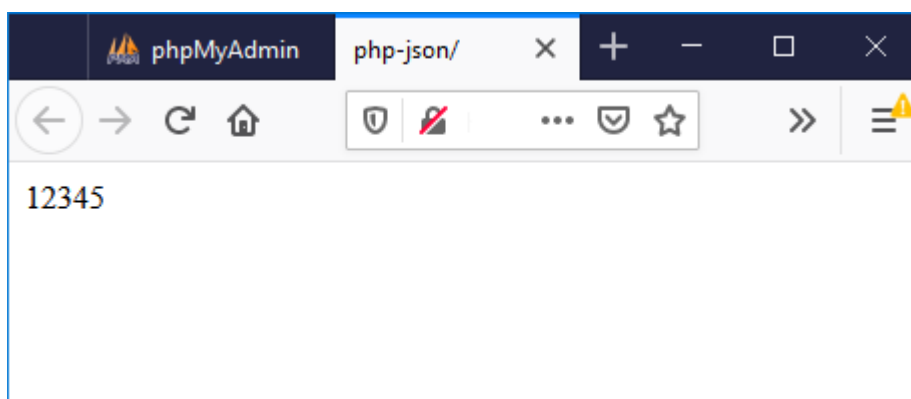
Доступ к элементам объекта, которые содержат символы, недопустимые в соответствии с соглашением об именах PHP (то есть дефис), может быть выполнен путем обрамления имени элемента фигурными скобками и апострофами.

```
<?php

$json = '{"foo-bar": 12345}';

$obj = json_decode($json);
print $obj->{'foo-bar'}; // 12345

?>
```

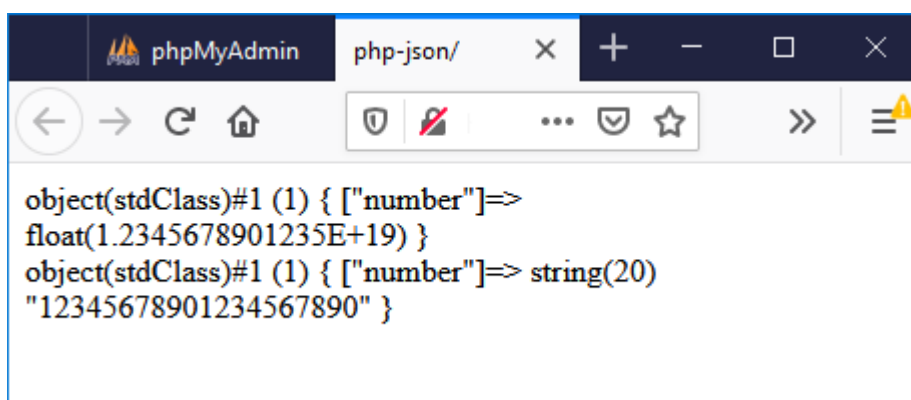


Пример №3 json_decode() с большими целыми числами

```
<?php
$json = '{"number": 12345678901234567890}';

var_dump(json_decode($json));
echo "<br>";
var_dump(json_decode($json, false, 512, JSON_BIGINT_AS_STRING));

?>
```



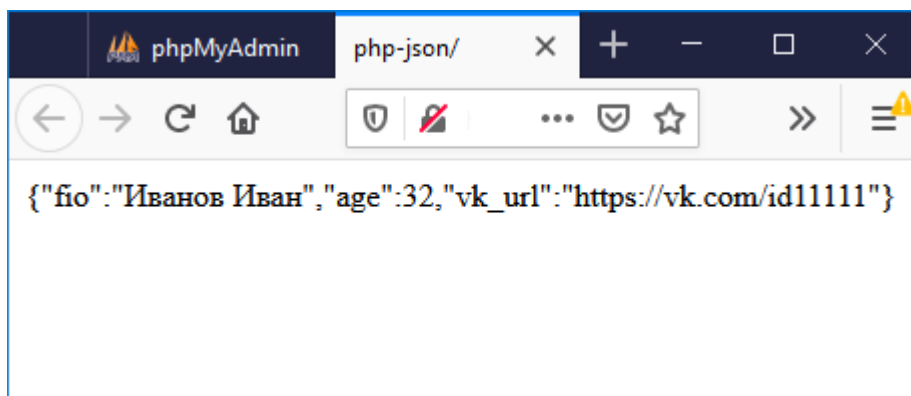
В целом необходимо отметить, что Спецификация JSON - это не JavaScript, а его подмножество. В случае ошибки декодирования можно использовать `json_last_error()` для определения ее причины.

Практические особенности работы с JSON на PHP

Важно отметить, что при кодировании в JSON-формат данных **на русском языке**, функция `json_encode` преобразует русские символы в юникод, т.е. заменяет их на `\uXXXX` и таким образом, json-строка становится не читабельной для человека (но понятной для браузера). Если нужно, чтобы преобразования в юникод не происходило (например, при отладке кода), можно просто использовать опцию `JSON_UNESCAPED_UNICODE`.

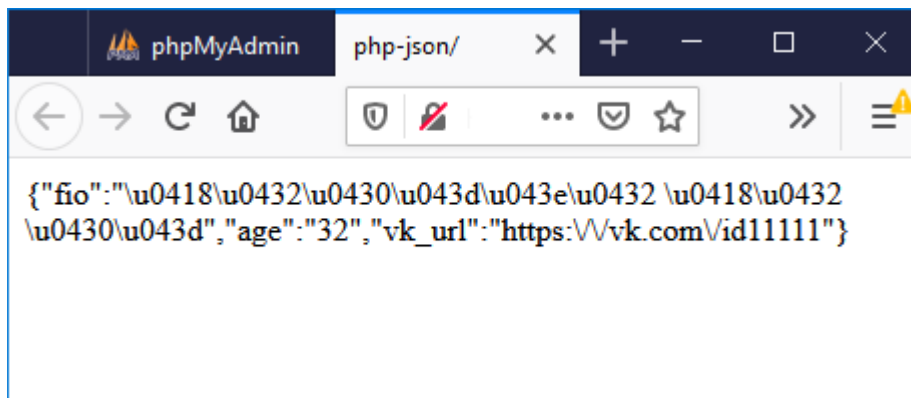
Так же, чтобы при кодировании не добавлялись слэши для экранирования и чтобы строки с числами кодировались как числа, можно использовать `JSON_UNESCAPED_SLASHES` и `JSON_NUMERIC_CHECK`. В итоге, чтобы json-строка была читабельной для человека, сделаем, например, так:

```
<?php
$arr = array(
    'fio' => 'Иванов Иван',
    'age' => '32',
    'vk_url' => 'https://vk.com/id11111'
);
echo json_encode($arr, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES |
JSON_NUMERIC_CHECK);
?>
```



А вот если бы скрипт выглядел следующим образом

```
<?php
$arr = array(
    'fio' => 'Иванов Иван',
    'age' => '32',
    'vk_url' => 'https://vk.com/id11111'
);
echo json_encode($arr);
?>
```



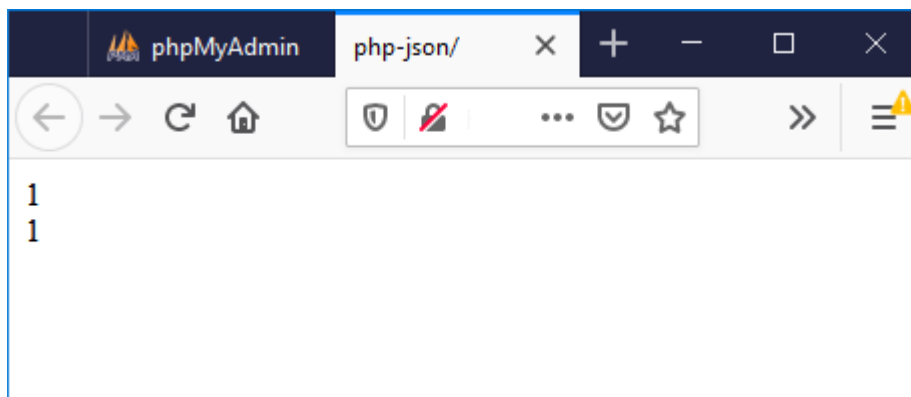
И еще один момент: если нужно чтобы при декодировании json-строки функция `json_decode` возвращала именно массив, просто добавьте второй параметр в функцию равный `true`.

```
$json_str = '{ "a":1, "b":2, "c":3 }';

$obj = json_decode($json_str); // получим объект
echo $obj->a; // выведет 1

echo "<br>";

$arr = json_decode($json_str, true); // получим ассоциативный массив
echo $arr['a']; // выведет 1
```



Передача данных с использованием формата JSON

Для генерации JavaScript из PHP, как уже отмечалось, нужно придерживаться синтаксиса JavaScript. Проще всего это сделать, воспользовавшись форматом JSON и функцией `json_encode()`, она превратит объект или массив PHP в строку, и при выполнении покажет объект JavaScript.

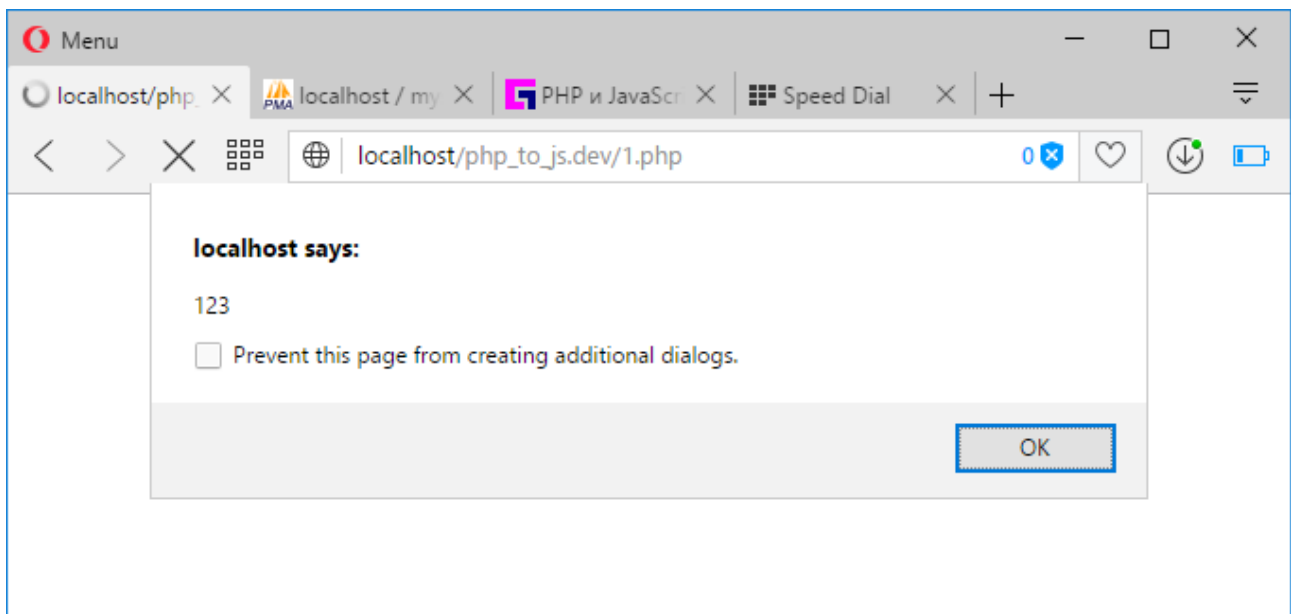
```
<?php
$arr = array('test'=>'123','key'=>'value');
$json_obj = json_encode($arr);
echo "<script language='javascript'>var obj=$json_obj;
alert(obj.test);</script>";
?>
```

Тут могут возникнуть сложности с понимаем происходящих процессов. При этом важно понимать последовательность выполнения операций. Сперва выполняется PHP, затем JavaScript.

После выполнения PHP получится:

```
<script language="javascript">var obj={"test":"123","key":"value"};
alert(obj.test);</script>
```

Что выведет окно alert с текстом 123.



Выводы

Важно помнить, что PHP генерирует JavaScript, поэтому передавать данные очень просто. Нужно формировать JavaScript так же, как формируется html с соблюдением синтаксиса JavaScript.

Для передачи данных из JavaScript в PHP всегда используется дополнительный запрос. Например, может формировать и отправлять данные веб-форм. Понимая это, можно довольно просто перекидывать любые данные из браузера на сервер и обратно.

Для передачи сложных данных целесообразно использовать формат JSON.

Список констант для работы с форматом JSON

JSON_ERROR_NONE (integer)

Не произошло никаких ошибок. Доступна начиная с PHP 5.3.0.

JSON_ERROR_DEPTH (integer)

Была превышена максимальная глубина стека. Доступно с PHP 5.3.0.

JSON_ERROR_STATE_MISMATCH (integer)

Неверный или поврежденный JSON. Доступно с PHP 5.3.0.

JSON_ERROR_CTRL_CHAR (integer)

Ошибка управляющих символов, вероятно, из-за неверного кодирования. Доступно с PHP 5.3.0.

JSON_ERROR_SYNTAX (integer)

Синтаксическая ошибка. Доступно с PHP 5.3.0.

JSON_ERROR_UTF8 (integer)

Поврежденные символы UTF-8, вероятно, из-за неверного кодирования. Доступно с PHP 5.3.3.

JSON_ERROR_RECURSION (integer)

Объект или массив, переданный в функцию `json_encode()` включает рекурсивные ссылки и не может быть закодирован. Если была передана опция `JSON_PARTIAL_OUTPUT_ON_ERROR`, то на месте рекурсивных ссылок будет выведен `NULL`. Доступно с PHP 5.5.0.

JSON_ERROR_INF_OR_NAN (integer)

Значение, переданное в функцию `json_encode()`, включает либо `NAN`, либо `INF`. Если была указана константа `JSON_PARTIAL_OUTPUT_ON_ERROR`, то вместо указанных особых значений будет выведен `0`. Доступно с PHP 5.5.0.

JSON_ERROR_UNSUPPORTED_TYPE (integer)

В функцию `json_encode()` было передано значение неподдерживаемого типа, например, `resource`. Если была указана константа `JSON_PARTIAL_OUTPUT_ON_ERROR`, то вместо неподдерживаемого значения будет выводиться `NULL`. Доступно с PHP 5.5.0.

JSON_ERROR_INVALID_PROPERTY_NAME (integer)

В строке переданной в `json_decode()` был ключ, начинающийся с символа `\u0000`. Доступно с PHP 7.0.0.

JSON_ERROR_UTF16 (integer)

Один непарный суррогат UTF-16 в экранированной последовательности Unicode в строке JSON, переданной в `json_decode()`. Доступно с PHP 7.0.0.

Можно комбинировать следующие константы для передачи в `json_decode()`.

JSON_BIGINT_AS_STRING (integer)

Декодирует большие целые числа в качестве исходного значения строки. Доступно с PHP 5.4.0.

JSON_OBJECT_AS_ARRAY (integer)

Преобразует объекты JSON в массив PHP. Эта опция может быть задана автоматически, если вызвать `json_decode()` указав вторым параметром `TRUE`. Доступно с PHP 5.4.0.

Следующие константы можно комбинировать для использования в `json_encode()`.

JSON_HEX_TAG (integer)

Все `<` и `>` кодируются в `\u003C` и `\u003E`. Доступно с PHP 5.3.0.

JSON_HEX_AMP (integer)

Все `&` кодируются в `\u0026`. Доступно с PHP 5.3.0.

JSON_HEX_APOS (integer)

Все символы `'` кодируются в `\u0027`. Доступно с PHP 5.3.0.

JSON_HEX_QUOT (integer)

Все символы `"` кодируются в `\u0022`. Доступно с PHP 5.3.0.

JSON_FORCE_OBJECT (integer)

Выдавать объект вместо массива при использовании неассоциативного массива. Это полезно, когда принимающая программа или код ожидают объект, а массив пуст. Доступно с PHP 5.3.0.

JSON_NUMERIC_CHECK (integer)

Кодирование строк, содержащих числа, как числа. Доступно с PHP 5.3.3.

JSON_PRETTY_PRINT (integer)

Использовать пробельные символы в возвращаемых данных для их форматирования. Доступно с PHP 5.4.0.

JSON_UNESCAPED_SLASHES (integer)

Не экранировать /. Доступно с PHP 5.4.0.

JSON_UNESCAPED_UNICODE (integer)

Не кодировать многобайтовые символы Unicode (по умолчанию они кодируются как \uXXXX). Доступно с PHP 5.4.0.

JSON_PARTIAL_OUTPUT_ON_ERROR (integer)

Позволяет избежать возникновения ошибок при использовании функции json_encode. Осуществляет подстановку значений по умолчанию вместо неcodируемых. Доступно с PHP 5.5.0.

JSON_PRESERVE_ZERO_FRACTION (integer)

Гарантирует, что значение типа float будет преобразовано именно в значение типа float в случае, если дробная часть равна 0. Доступно с PHP 5.6.6.

JSON_UNESCAPED_LINE_TERMINATORS (integer)

Символы конца строки не будут экранироваться, если задана константа JSON_UNESCAPED_UNICODE. Поведение будет таким же, какое оно было до PHP 7.1 без этой константы. Доступно с PHP 7.1.0.

Следующие константы можно комбинировать для использования в json_encode().

JSON_INVALID_UTF8_IGNORE (integer)

Игнорировать некорректные символы UTF-8. Доступно с PHP 7.2.0.

JSON_INVALID_UTF8_SUBSTITUTE (integer)

Преобразовывать некорректные символы UTF-8 в \0xfffd (Символ Юниокда 'REPLACEMENT CHARACTER') Доступно с PHP 7.2.0.

JSON_THROW_ON_ERROR (integer)

Выбрасывается исключение JsonException в случае возникновения ошибок вместо установки глобального состояния ошибки, которое может быть получено с помощью функции json_last_error() и json_last_error_msg().

Константа JSON_PARTIAL_OUTPUT_ON_ERROR имеет приоритет
над JSON_THROW_ON_ERROR. Доступно с PHP 7.3.0.