

Южно-Уральский  
государственный  
университет

Национальный  
исследовательский  
университет

# КРОК

## Серверная часть клиент-серверного взаимодействия. Основы РНР

КРОК

Челябинск, ул. Карла Маркса, д. 38

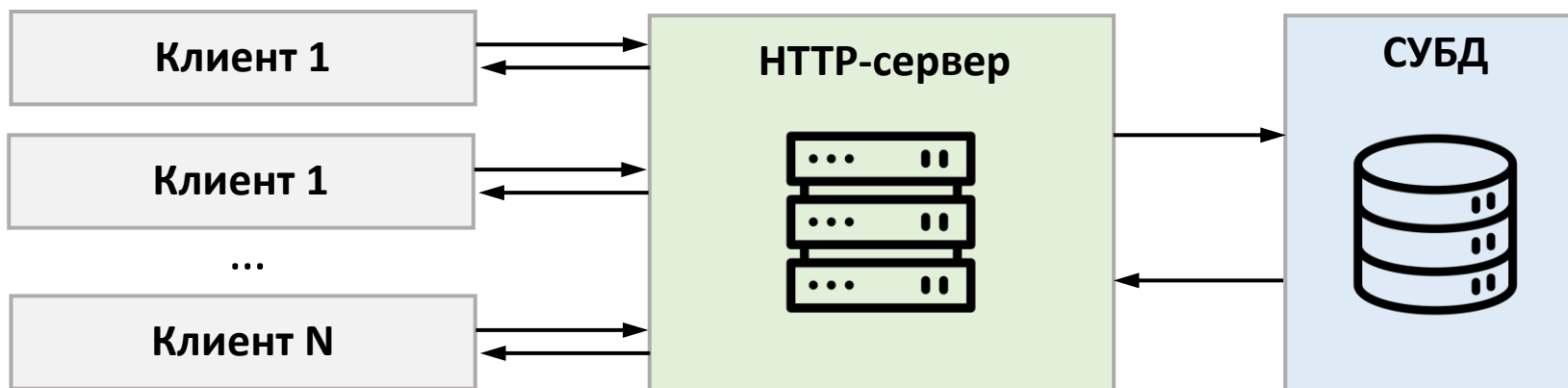
Смирнов Анатолий  
Технический менеджер


Кузнецов Сергей  
Старший инженер-разработчик


Фоменко Алексей  
Младший инженер-разработчик



- Обработка HTTP-запроса
- Отправка HTTP-ответа
- Аутентификация и авторизация
- Обработка и валидация данных HTTP-запроса
- Взаимодействие с СУБД



 <b>Аутентификация</b>
Проверка подлинности (логин/пароль)

 <b>Авторизация</b>
Проверка доступа (роли)



Самые распространенные форматы передачи данных:

- JSON (JavaScript Object Notation)
- XML (eXtensible Markup Language)

## JSON (REST)

```
{
  "Header": {
    "Authorization": "25dc1a8a-642c"
  },
  "Body": {
    "UpdateHomeCallRequest" : {
      "idHomeCallRequest": 29974,
      "homeCallStatus": 3,
      "guid": "C9015DAA-1F92-4043-82A3-E395F5E483F0"
    }
  }
}
```

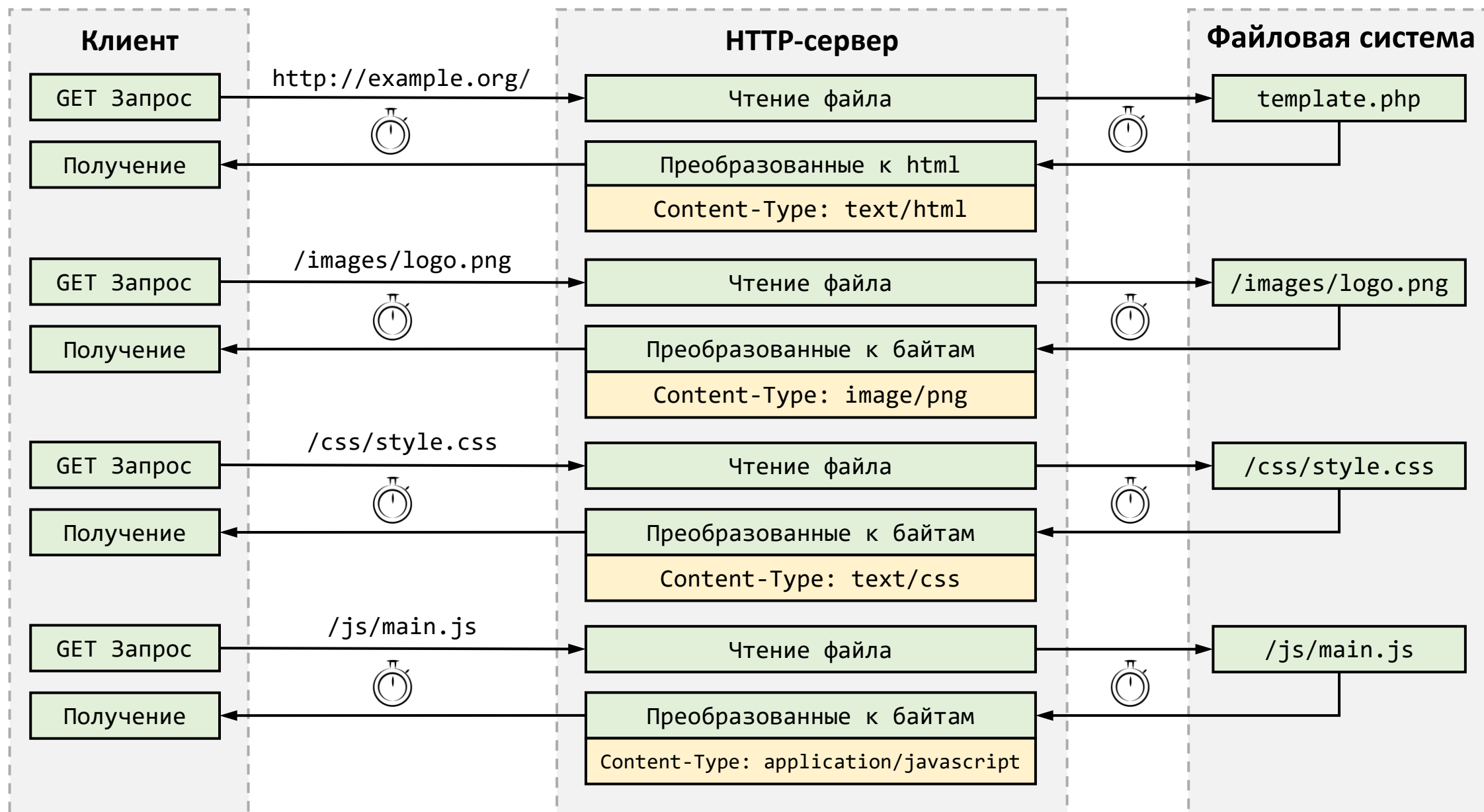
Длина: 189 символов

## XML (SOAP)

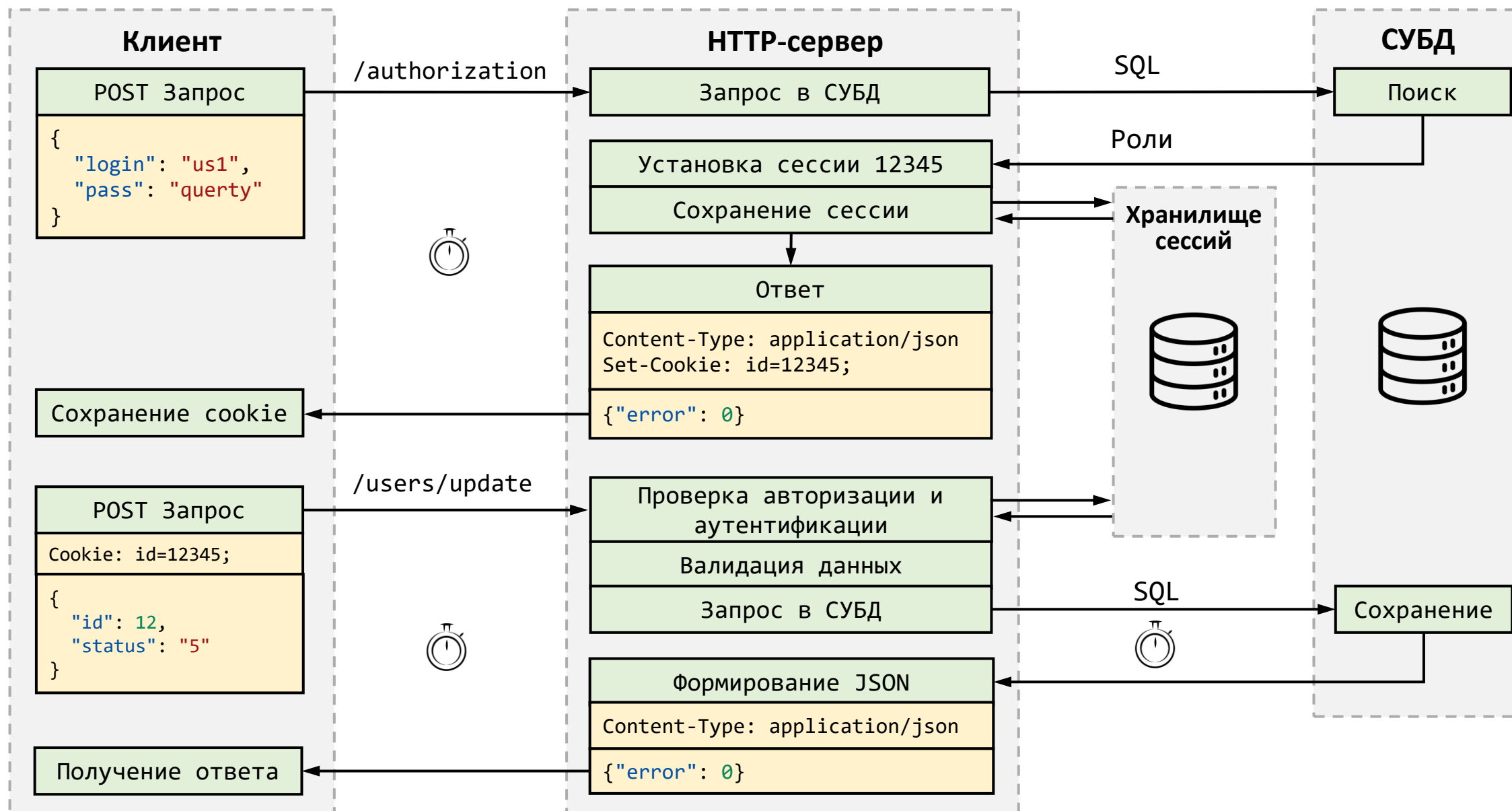
```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1="http://tempuri.org/">
  <SOAP-ENV:Header>
    <SOAP-ENV:Authorization>
      25dc1a8a-642c
    </SOAP-ENV:Authorization>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns1:UpdateHomeCallRequest>
      <ns1:idHomeCallRequest>29974</ns1:idHomeCallRequest>
      <ns1:homeCallStatus>3</ns1:homeCallStatus>
      <ns1:guid>
        C9015DAA-1F92-4043-82A3-E395F5E483F0
      </ns1:guid>
    </ns1:UpdateHomeCallRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Длина: 486 символов

# Пример работы HTTP-сервера



# Пример работы HTTP-сервера

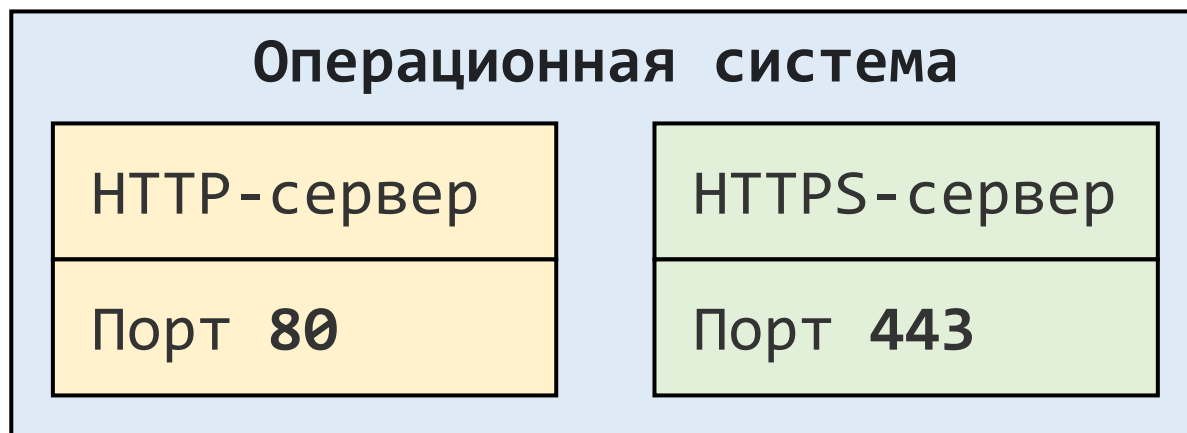


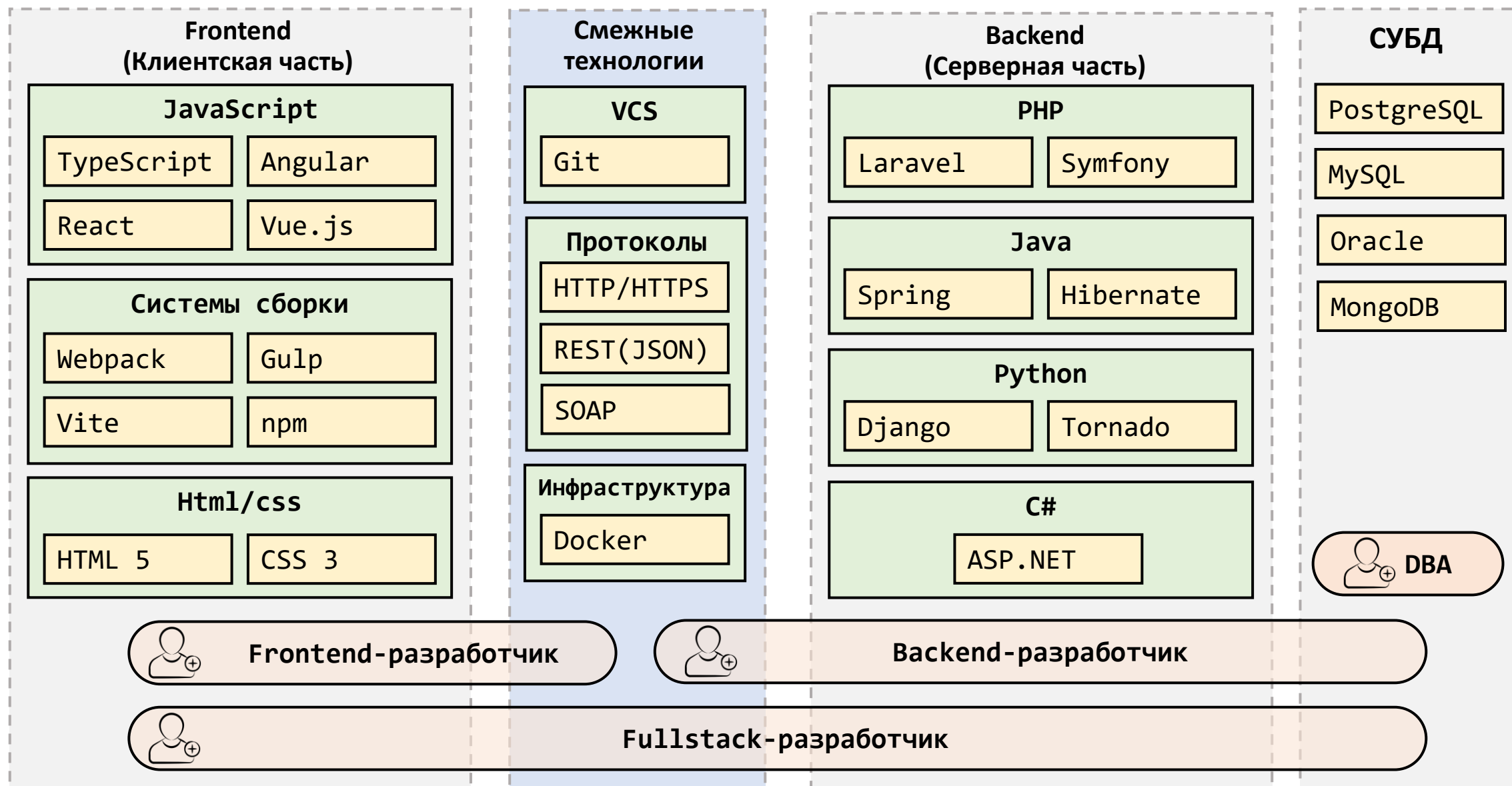


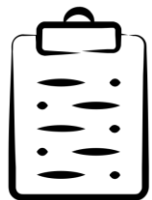
Сам по себе протокол HTTP не предполагает использование шифрования для передачи информации.

Для упаковки передаваемых данных используется криптографический протокол SSL или TLS.

Название этого протокола **HTTPS** (*HyperText Transfer Protocol Secure*).







Фронтенд (англ. frontend) — это разработка пользовательских функций и интерфейса. К ним относится всё, что пользователи видят на сайте или в приложении, и с чем можно взаимодействовать: картинки, выпадающие списки, меню, анимация, карточки товаров, кнопки, чекбоксы, интерактивные элементы.



Backend-разработчик — программист, который пишет серверный код, отвечает за реакцию ресурса на действия пользователя и выдачу информации.

Backend разработчики имеют дело со всем, что относится к СУБД, архитектуре и программной логике серверного приложения.



# Пример запуска HTTP-сервера. Python

---

```
from http.server import HTTPServer, BaseHTTPRequestHandler
```

```
class Handler(BaseHTTPRequestHandler):  
    def do_GET(self):  
        self.send_response(200)  
        self.end_headers()  
        self.wfile.write(b'Hello, world, python!')
```

```
httpd = HTTPServer(('localhost', 80), Handler)  
httpd.serve_forever()
```

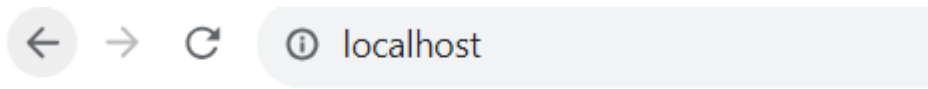
A screenshot of a web browser's address bar. It features navigation icons (back, forward, refresh) on the left and a search icon on the right. The text 'localhost' is entered in the address field.

Hello, world, python!

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

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.end('Hello World, nodeJS');
});

server.listen(80, 'localhost');
```

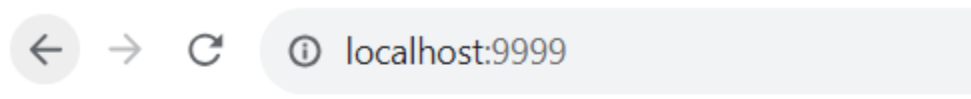


Hello World, nodeJS

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintf(w, "Hello World, Golang!")
    })
    http.ListenAndServe(":9999", nil)
}
```



Hello World, Golang!



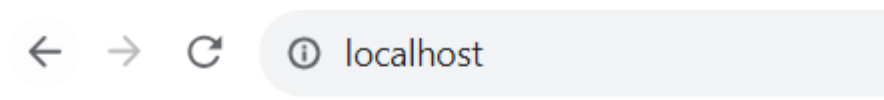
При указании порта, отличного от 80, его также надо указывать в браузере

# Пример запуска HTTP-сервера. Java

```
import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.HttpHandler;
import com.sun.net.httpserver.HttpExchange;

import java.io.OutputStream;
import java.io.IOException;
import java.net.InetSocketAddress;

public class server {
    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress(80), 0);
        server.createContext("/", new MyHandler());
        server.setExecutor(null);
        server.start();
    }
    static class MyHandler implements HttpHandler {
        @Override
        public void handle(HttpExchange t) throws IOException {
            String response = "Hello, World, Java!";
            t.sendResponseHeaders(200, response.length());
            OutputStream os = t.getResponseBody();
            os.write(response.getBytes());
            os.close();
        }
    }
}
```



Hello, World, Java!

```
<?php
    echo "Hello, world, PHP!";
?>
```

A screenshot of a web browser's address bar. It features navigation icons (back, forward, refresh) on the left and the text 'localhost' on the right, preceded by an information icon.

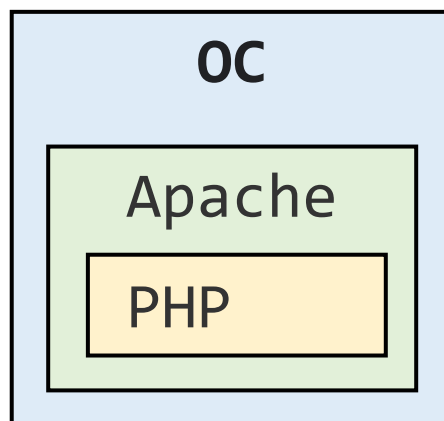
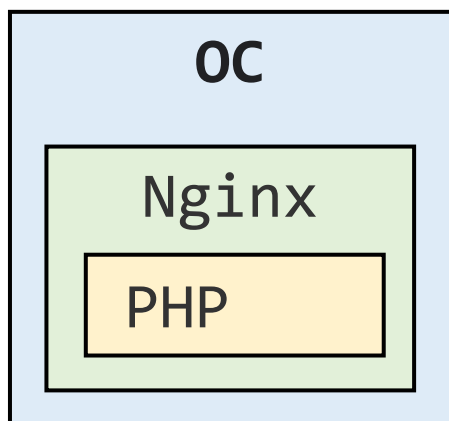
Hello, world, PHP!

Пример	Комментарий
Python	Однопоточный
NodeJS	Однопоточный
Java	Однопоточный
Golang	Многопоточный
PHP*	Многопоточный (многопроцессорный)



PHP не может быть веб-сервером! Необходим внешний веб-сервер для работы с PHP. Обычно это либо Nginx, либо Apache HTTP Server.

- Nginx: более быстрый для статического контента
- Apache: легче конфигурировать





PHP: Hypertext Preprocessor - PHP: Препроцессор Гипертекста, создан в 1994 году.

Является распространённым интерпретируемым языком общего назначения с открытым исходным кодом. PHP создавался специально для ведения веб-разработок и код на нем может внедряться непосредственно в HTML-код.



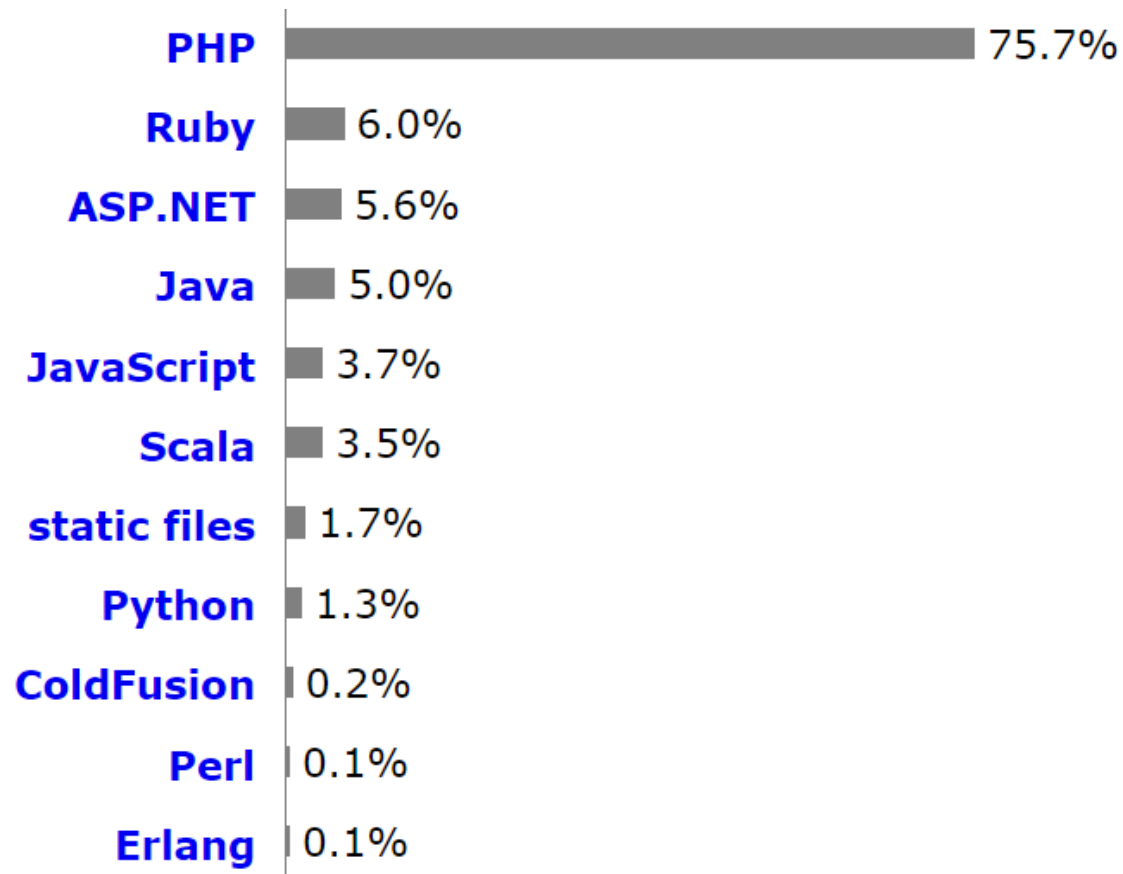
Особенности:

- Простота использования
- Популярность
- Нестрогая типизация
- Си-подобный синтаксис



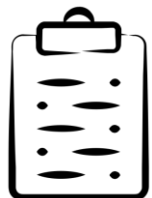


Из анализа языка, проведенного W3 Techs на 10 миллионах лучших веб-сайтов по всему миру (обновляется ежедневно):



[https://w3techs.com/technologies/overview/programming\\_language](https://w3techs.com/technologies/overview/programming_language)





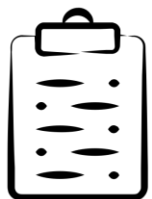
Код PHP заключается в специальный тэг:

```
<?php  
// Код на PHP  
?>
```



Все переменные в данном языке должны начинаться со знака \$. Тип при этом не указывается:

```
$a = 5;
```



- Точка с запятой обязательна.
- Отступы не важны



Комментарии:

```
<?php
```

```
    $a = 5; # Однострочный комментарий в стиле Unix
```

```
    $b = 4; // Однострочный комментарий
```

```
    $c /* Многострочный  
        комментарий
```

```
    */ = 3;
```

```
?>
```



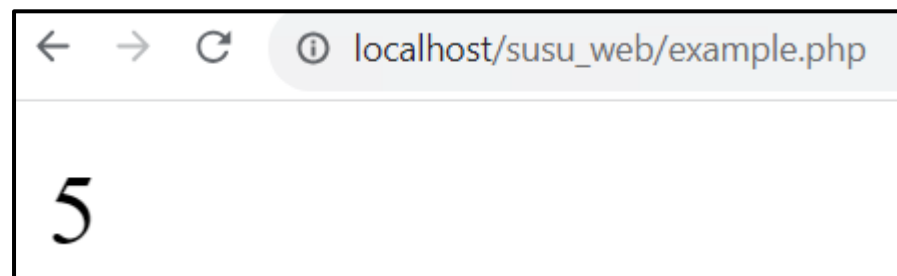
Для вывода данных используется оператор echo:

```
<?php
```

```
    $a = 5;
```

```
    echo $a;
```

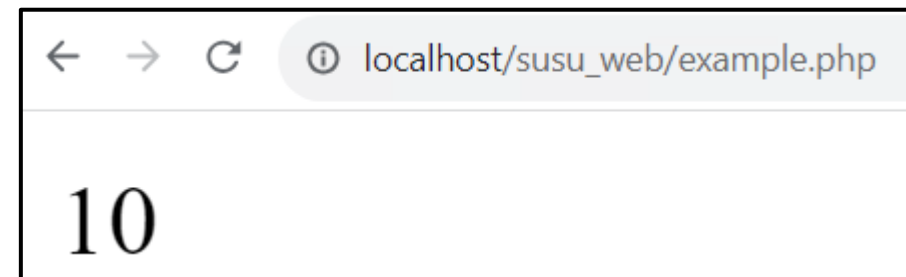
```
?>
```





Ссылка одной переменной на другую задается символом амперсанда &:

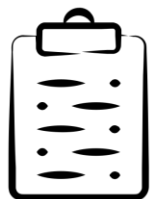
```
<?php
    $a = 5;
    $b = &$a;
    $b = 10;
    echo $a; // выведет 10
?>
```





Целочисленное значение:

```
$a = 5;
```



Числа с плавающей точкой (float или double):

```
$b = 21.5;
```

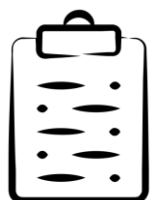
```
$c = 5.1e4; // 5.1 * 10^4
```



Переменные логического типа:

```
$d = true;
```

```
$e = false;
```



Специальное значение null:

```
$a = null;
```



Используются одинарные или двойные кавычки:

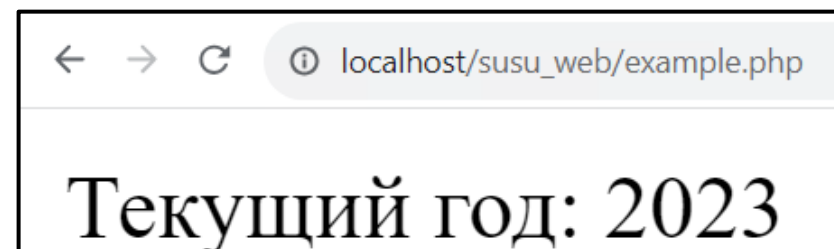
```
$string_1 = 'Простая строка';
```



В случае двойных кавычек PHP сможет распознать строке управляющие последовательности и производить обработку переменных:

```
<?php
    $string_2 = "Простая строка";

    $year = 2023;
    $string_3 = "Текущий год: $year";
    echo $string_3;
?>
```

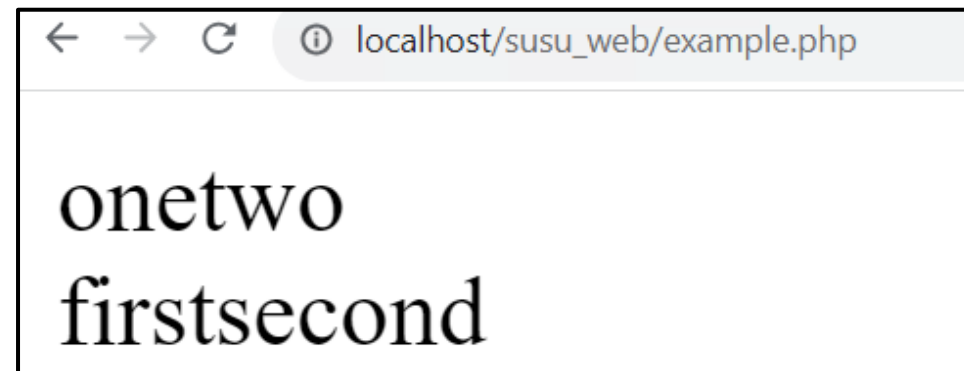




В качестве оператора конкатенации (объединения строк), в PHP используется оператор «точка» . (не «+»)

```
<?php
$string_one = "one";
$string_two = "two";
$string_result_one = $string_one . $string_two; // onetwo
$string_result_two = "first" . "second"; // firstsecond

echo $string_result_one;
echo "<br/>";
echo $string_result_two;
?>
```

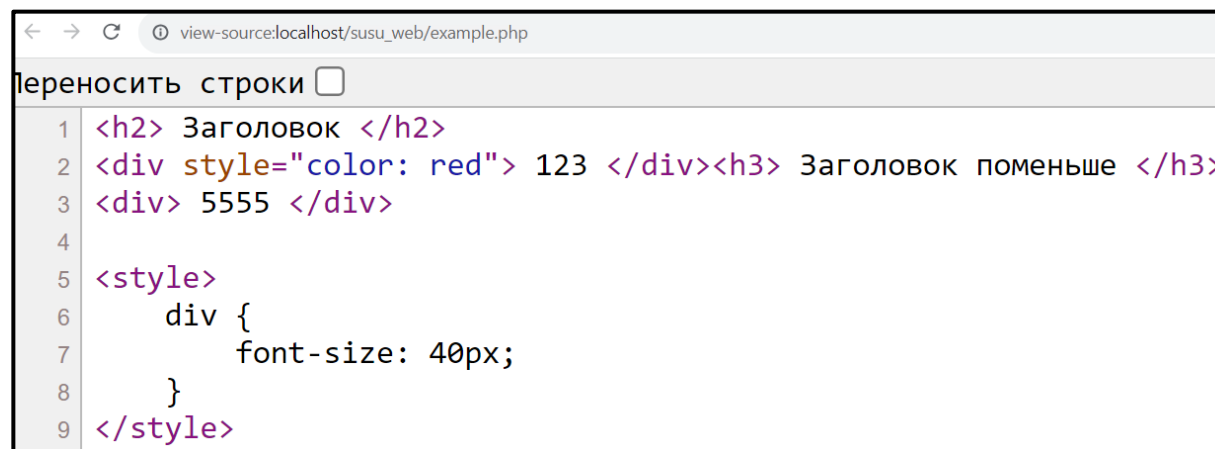
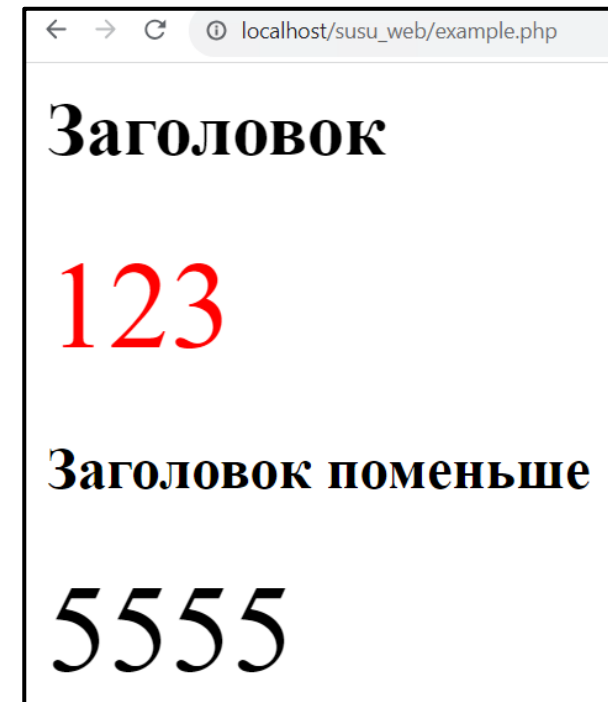




- PHP встраивается в html-верстку
- echo может выводить html-верстку

```
<h2> Заголовок </h2>
<?php
    $a = 5555;
    echo '<div style="color: red"> 123 </div>';
?>
<h3> Заголовок поменьше </h3>
<div> <?php echo $a?> </div>
```

```
<style>
    div {
        font-size: 40px;
    }
</style>
```



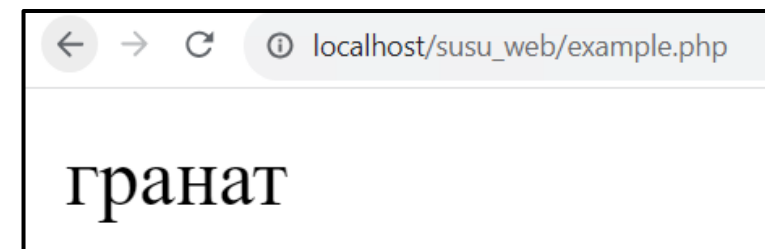


- Массивы делятся на две группы: ассоциативные и индексируемые.
- Индексируемый массив - это упорядоченный набор данных, в котором каждому значению массива задается порядковый номер (индекс). По умолчанию индекс начинается с нуля.

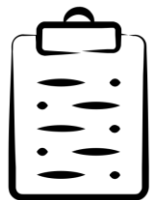
```
<?php
// Полный синтаксис
$array_3 = array("яблоко", "груша", "гранат");

// Сокращенный синтаксис
$array_4 = ["яблоко", "груша", "гранат"];

// Получение значения по ключу:
echo $array_3[2]; // гранат
?>
```



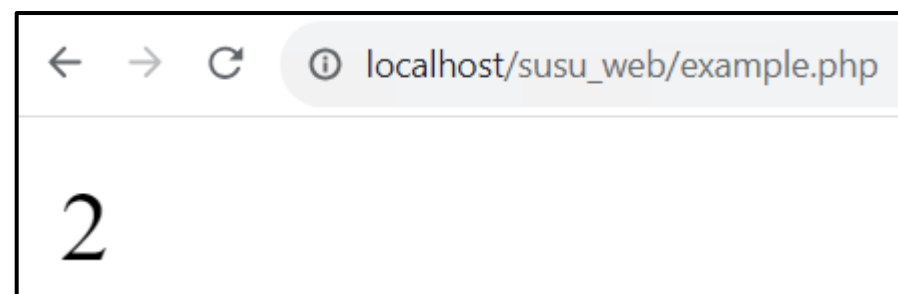




Ассоциативные массивы - это набор данных в формате ключ => значение. Ключ может быть в формате int или string, а значение может быть любым типом данных.

```
<?php
// Полный синтаксис
$array_1 = array(
    "one"    => 1,
    "two"    => 2,
    "three"  => 3
);
// Сокращенный синтаксис
$array_2 = [
    "one"    => 1,
    "two"    => 2,
    "three"  => 3
];
// Получение значения по ключу:
$value_1 = $array_1["two"]; // 2
echo $value_1;

?>
```



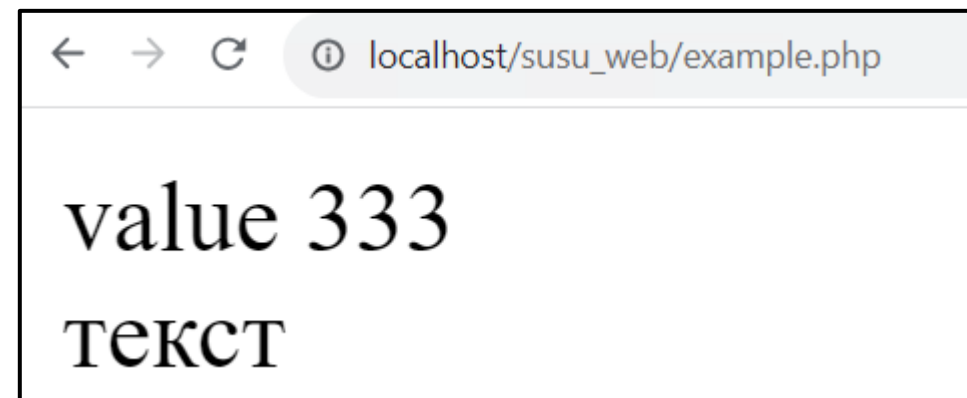


Другой пример ассоциативных массивов:

```
<?php
// Полный синтаксис
$array_11 = array(
    "one"    => "value 1",
    "two"    => "value 333",
    "three"  => "value 444"
);

// Сокращенный синтаксис
$array_22 = [
    "one" => "value 1",
    7    => "текст",
    3    => 8
];

// Получение значения по ключу:
echo $array_11["two"]; // value 333
echo "<br/>";
echo $array_22[7]; // текст
?>
```





Для вывода всех значений массива в удобной форме используется `print_r` либо `json_encode`:

```
<?php
    $a = [1, 2, 66, 33];
    //echo $a; // Будет ОШИБКА

    echo json_encode($a);
    echo '<br/>';

    echo print_r($a);
    echo '<br/>';

    echo '<pre>';
    echo print_r($a);
    echo '</pre>';

?>
```

```
localhost/susu_web/example.php

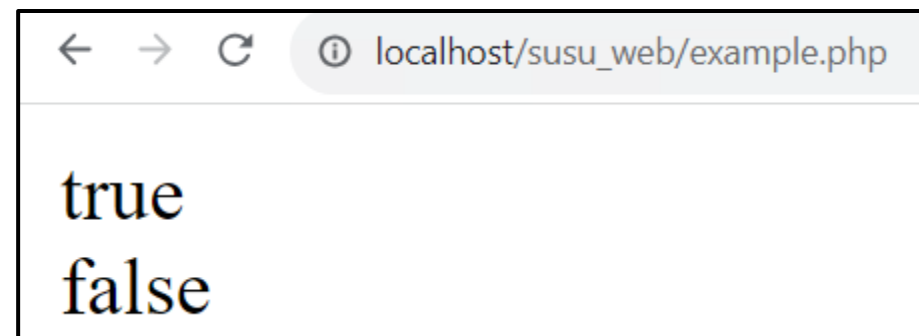
[1,2,66,33]

Array ( [0] => 1 [1] => 2 [2] => 66 [3] => 33 ) 1

Array
(
    [0] => 1
    [1] => 2
    [2] => 66
    [3] => 33
)
1
```

\$a == \$b	Равно	true если \$a равно \$b после преобразования типов.	
\$a === \$b	Тождественно равно	true если \$a равно \$b и имеет тот же тип.	
\$a and \$b	И	true, если и \$a, и \$b true.	Самый низкий приоритет
\$a or \$b	Или	true, если или \$a, или \$b true.	
! \$a	Отрицание	true, если \$a не true.	
\$a && \$b	И	true, если и \$a, и \$b true.	
\$a    \$b	Или	true, если или \$a, или \$b true.	

```
<?php
    echo json_encode(1 == '1');
    echo "<br/>";
    echo json_encode(1 === '1');
?>
```



```
localhost/susu_web/example.php

true
false
```



Для условных конструкций используются ключевые слова: if, else и elseif:

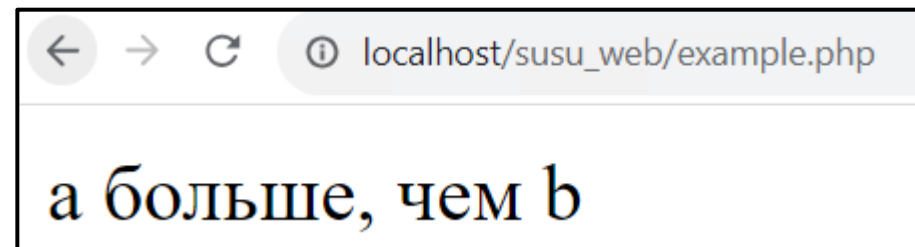
```
<?php
```

```
    $a = 5;
```

```
    $b = 4;
```

```
    if ($a > $b) {  
        echo "a больше, чем b";  
    } elseif ($a == $b) {  
        echo "a равен b";  
    } else {  
        echo "a меньше, чем b";  
    }  
}
```

```
?>
```





Синтаксисы циклов for, while и do while совпадают с синтаксисом языка C:

```
<?php
    for($i = 0; $i < 6; $i++) {
        echo "<div>$i</div>";
    }

    echo "<br>";

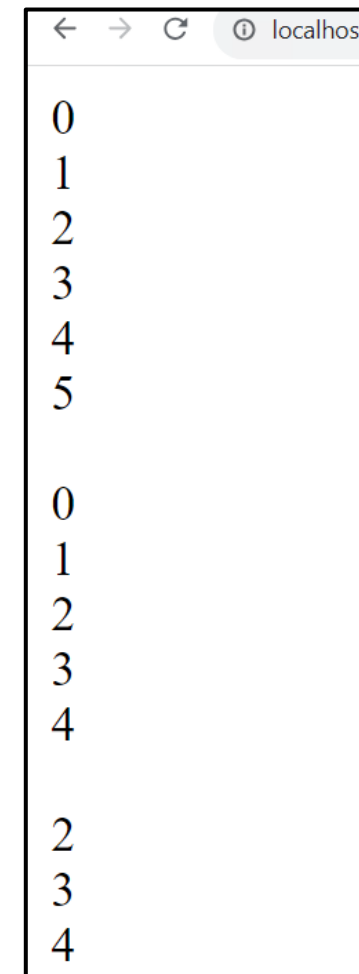
    $var = 0;
    while($var < 5) {
        echo "<div>$var</div>";
        $var++;
    }

    echo "<br>";

?>
```

```
<?php
    $var2 = 2;
    do {
        echo "<div>$var2</div>";
        $var2++;
    } while ($var2 < 5);

?>
```

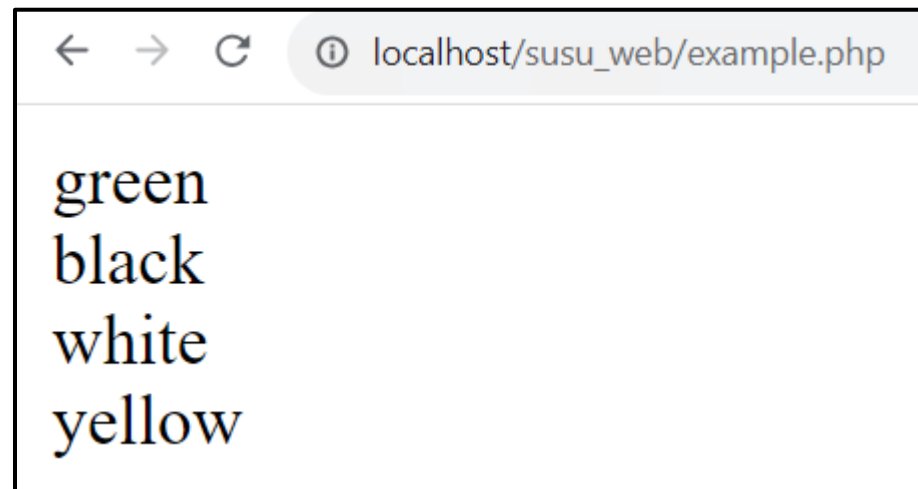




Для упрощенного перебора массивов в PHP существует цикл foreach:

```
<?php
    $colors = ["green", "black", "white", "yellow"];

    foreach($colors as $value) {
        echo "<div>$value</div>";
    }
?>
```





В foreach можно получить доступ и к ключу:

```
<?php
    $colors = ["green", "black", "white", "yellow"];

    foreach($colors as $my_key => $value) {
        echo "<div>Индекс: $my_key, значение: $value</div>";
    }
?>
```

localhost/susu\_web/example.php

```
Индекс: 0, значение: green
Индекс: 1, значение: black
Индекс: 2, значение: white
Индекс: 3, значение: yellow
```



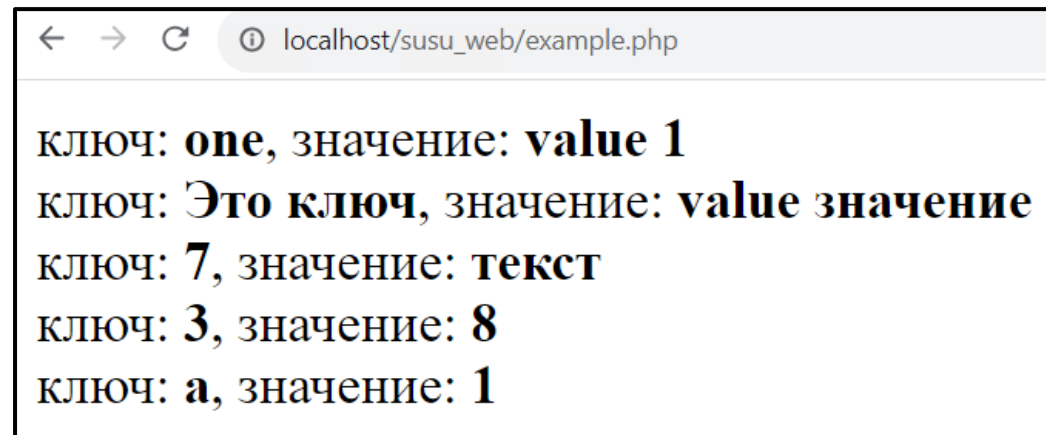


В foreach можно перебирать ассоциативный массив:

```
<?php
$array1 = [
    "one" => "value 1",
    "Это ключ" => "value значение",
    7 => "текст",
    3 => 8,
    "a" => true
];

foreach($array1 as $my_key => $value) {
    echo "<div>ключ: <b>$my_key</b>, значение: <b>$value</b></div>";
}

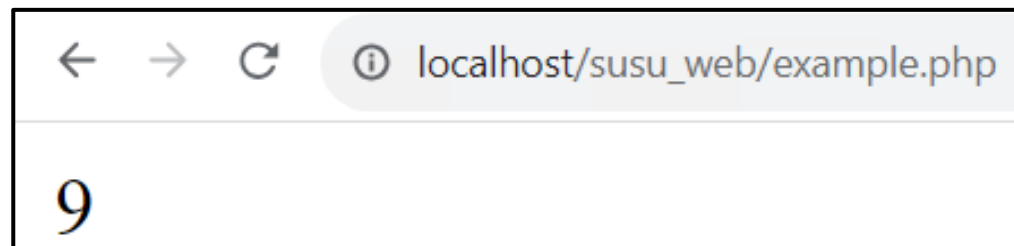
?>
```





Для создания функции в PHP используется ключевое слово `function`

```
<?php
    function sumNumbers($num1, $num2) {
        return $num1 + $num2;
    }
    echo sumNumbers(5, 4);
?>
```

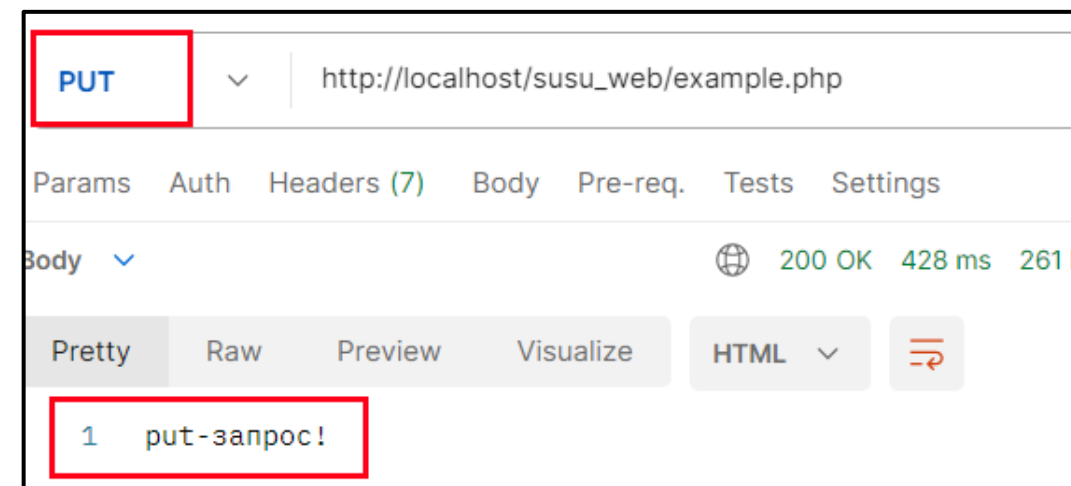
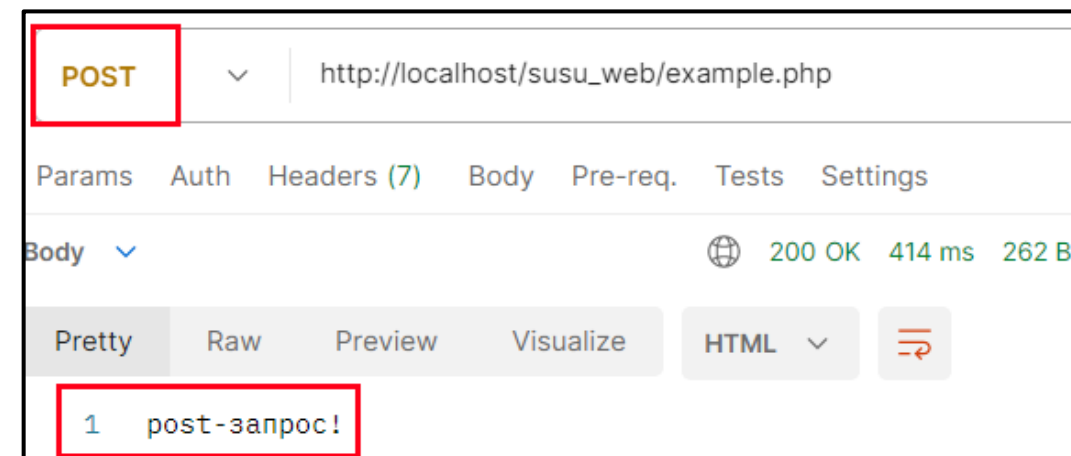
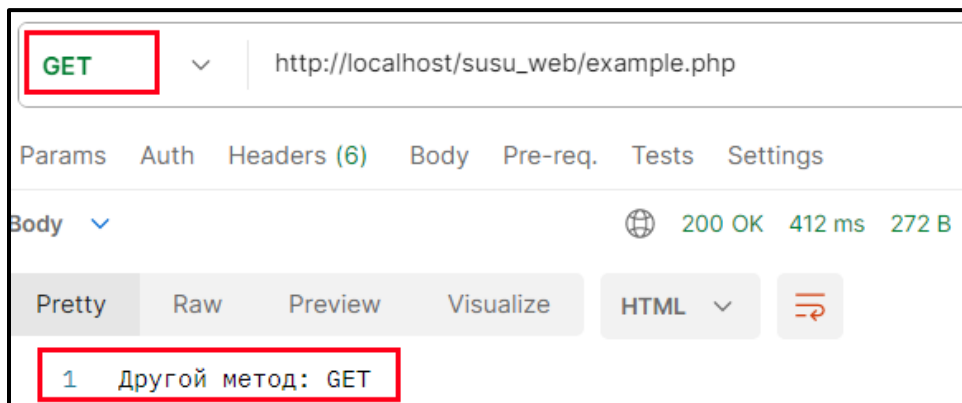




Используется ассоциативный массив `$_SERVER`:

```
<?php
    $method = $_SERVER['REQUEST_METHOD'];

    if ($method == 'POST') {
        echo "post-запрос!";
    } elseif ($method == 'PUT') {
        echo "put-запрос!";
    } else {
        echo "Другой метод: " . $method;
    }
?>
```





Используется ассоциативный массив \$\_GET:

```
<?php
    foreach ($_GET as $key => $value) {
        echo "Ключ: <b>" . $key . "</b>, значение: <b>"
            . $value . "</b><br/>";
    }
?>
```

← → ↻ ⓘ localhost/susu\_web/example.php?name=Иван&еще\_ключ=12345

Ключ: **name**, значение: **Иван**

Ключ: **еще\_ключ**, значение: **12345**

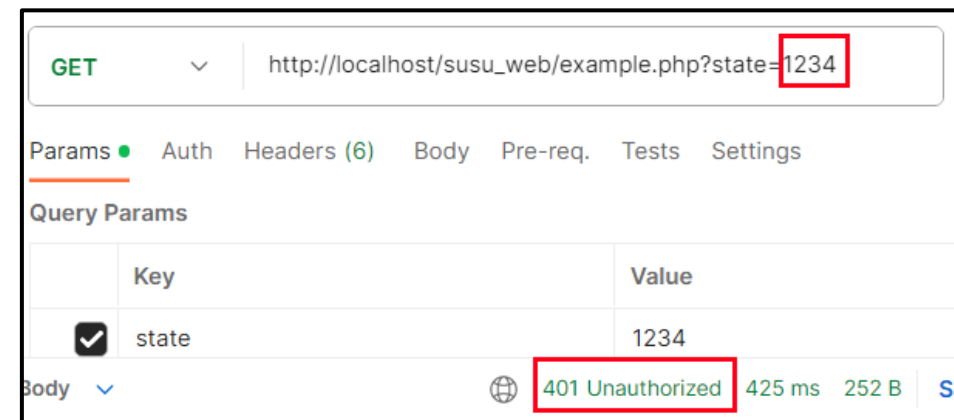
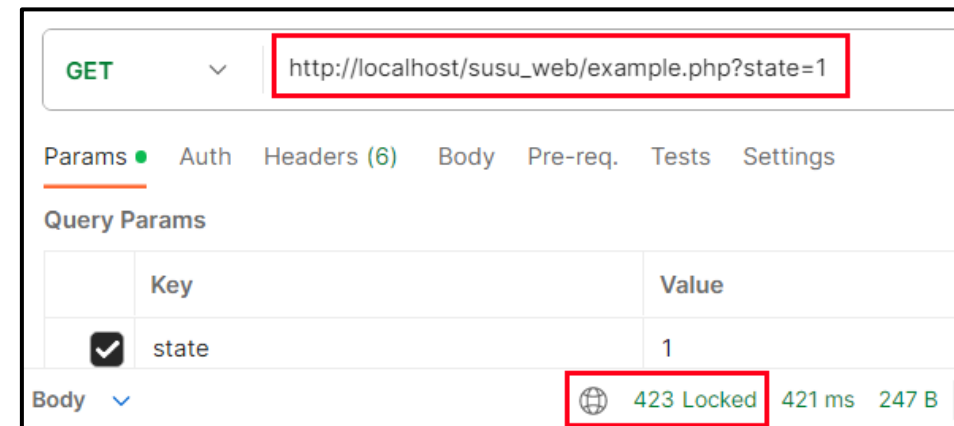
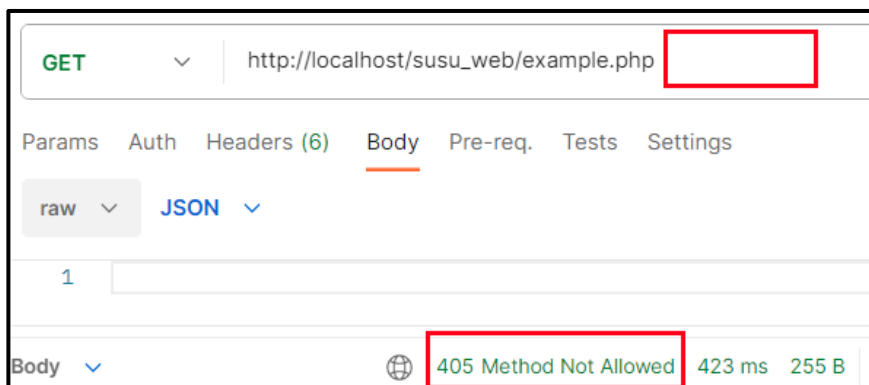


Функция `http_response_code(<Код>)` возвращает клиенту любой код состояния HTTP:

```
<?php
if (!array_key_exists("state", $_GET)) {
    http_response_code(405);
    die();
}

if ($_GET["state"] == 1) {
    http_response_code(423);
} else {
    http_response_code(401);
}

?>
```





Используем функции `file_get_contents` и `json_decode`:

```
<?php
if ($_SERVER['REQUEST_METHOD'] != 'POST') {
    http_response_code(403);
    die();
}

$entityBody = file_get_contents('php://input');

if(!$entityBody) {
    http_response_code(404);
    die();
}

$post = json_decode($entityBody, true);

foreach ($post as $value) {
    echo "<div>$value</div>";
}

?>
```

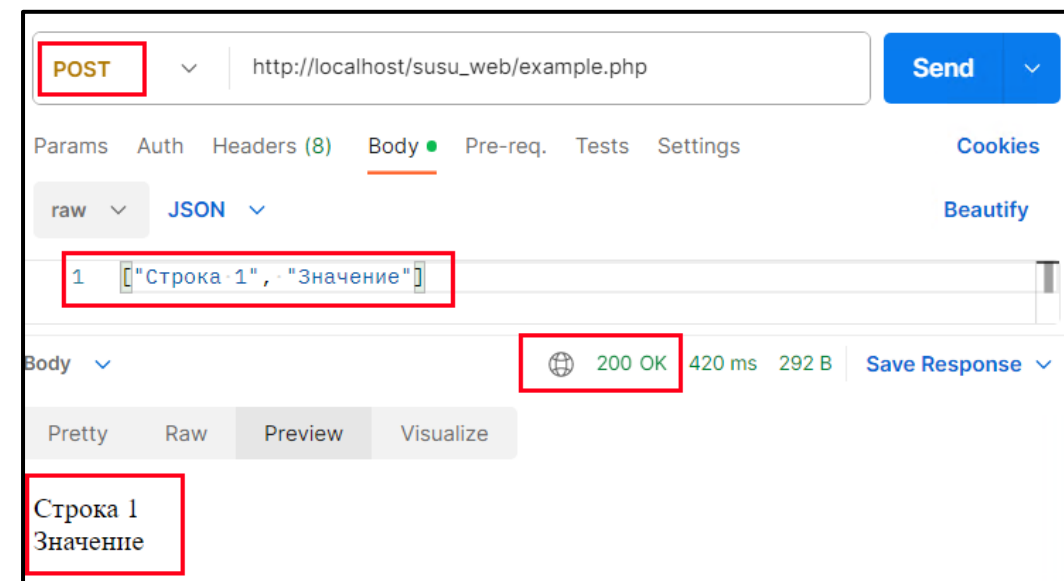


Доступ к localhost запрещен

У вас нет прав для просмотра этой страницы.

HTTP ERROR 403

Перезагрузить



# PHP. Получение тела POST-запроса

```
<?php
    if ($_SERVER['REQUEST_METHOD'] != 'POST') {
        http_response_code(403);
        die();
    }

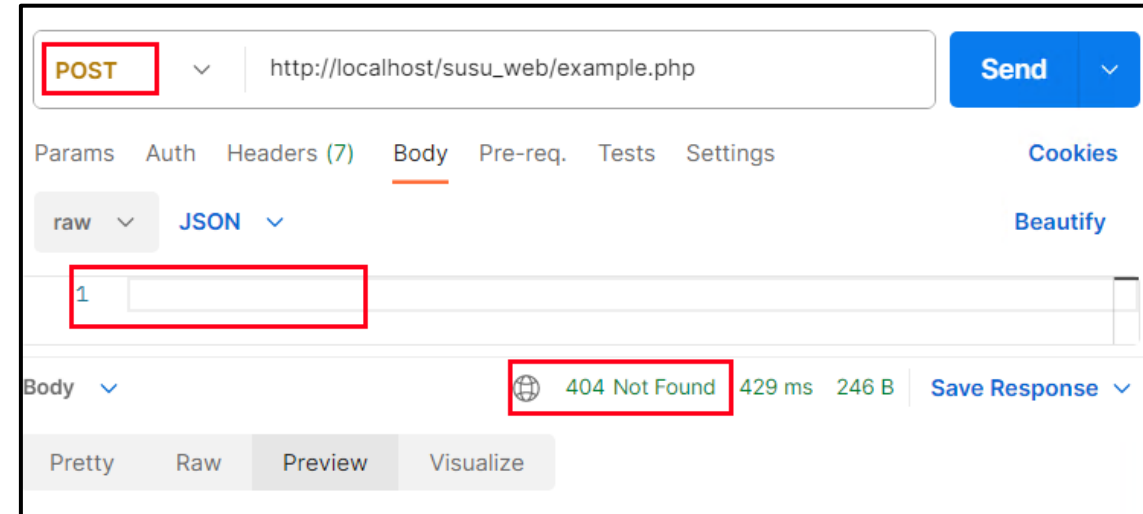
    $entityBody = file_get_contents('php://input');

    if(!$entityBody) {
        http_response_code(404);
        die();
    }

    $post = json_decode($entityBody, true);

    foreach ($post as $value) {
        echo "<div>$value</div>";
    }

?>
```



# Пример 1

Реализовать функцию аналогичную стандартной `in_array`: `in_array_user($значение, $массив)`. Если в массиве есть значение, то вернуть `true`, иначе `false`.

```
<?php
function in_array_user($value, $array) {
    for($i = 0; $i < count($array); ++$i) {
        if ($array[$i] === $value) {
            return true;
        }
    }

    return false;
}

$a = [1, 2, 66, 33, 54, 8, 9];

echo "in_array:<br/>";
echo json_encode(in_array(2, $a)) . "<br/>";
echo json_encode(in_array(4, $a)) . "<br/><br/>";

echo "in_array_user:<br/>";
echo json_encode(in_array_user(2, $a)) . "<br/>";
echo json_encode(in_array_user(4, $a)) . "<br/>";

?>
```

← → ↻ ⓘ localhost/susu\_web/1.php

in\_array:

true

false

in\_array\_user:

true

false



# Пример 2

Реализовать функцию, аналогичную стандартной `array_diff`: `array_diff_user($массив1, $массив2)`, которая вернет массив, в котором будут все данные из массива `$массив1`, если они не присутствуют в массиве `$массив2`.

```
<?php
function array_diff_user($array1, $array2) {
    $temp = [];

    foreach ($array1 as $value) {
        if (!in_array($value, $array2)) {
            array_push($temp, $value);
        }
    }

    return $temp;
}

$a = [1, 2, 66, 33, 54, 8, 9];
$b = [3, 5, 66, 8];

echo '<pre>';
echo print_r(array_diff($a, $b));
echo '</pre>';

echo '<pre>';
echo print_r(array_diff_user($a, $b));
echo '</pre>';
?>
```

← → ↺ localhost/susu\_web/2.php

```
Array
(
    [0] => 1
    [1] => 2
    [3] => 33
    [4] => 54
    [6] => 9
)
```

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 33
    [3] => 54
    [4] => 9
)
```

# Пример 3

Вывести таблицу из трех столбцов.

```
<style>
    table, th, td {
        border: 1px solid black;
        border-collapse: collapse;
    }
</style>
<table>
    <thead>
        <th>
            <td>Индекс</td>
            <td>Имя</td>
            <td>Значение</td>
        </th>
    </thead>
<?php
    $a = ["Иван", "Алексей", "Мария"];
    $b = [111, 333, 444];

    foreach ($a as $index => $value) {
        echo "
        <tr>
            <td>$index</td>
            <td>$value</td>
            <td>$b[$index]</td>
        </tr>";
    }
?>
</table>
```

← → ↻ ⓘ localhost/susu\_web/3.php

Индекс	Имя	Значение
0	Иван	111
1	Алексей	333
2	Мария	444

← → ↻ ⓘ view-source:localhost/susu\_web/3.php

Переносить строки ☐

```
1 <style>
2     table, th, td {
3         border: 1px solid black;
4         border-collapse: collapse;
5     }
6 </style>
7
8 <table>
9     <thead>
10        <td>Индекс</td>
11        <td>Имя</td>
12        <td>Значение</td>
13    </thead>
14
15    <tr>
16        <td>0</td>
17        <td>Иван</td>
18        <td>111</td>
19    </tr>
20    <tr>
21        <td>1</td>
22        <td>Алексей</td>
23        <td>333</td>
24    </tr>
25    <tr>
26        <td>2</td>
27        <td>Мария</td>
28        <td>444</td>
29    </tr></table>
```

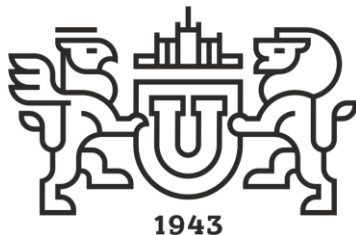
# Пример 3 (продолжение)

The screenshot shows the Chrome DevTools Network tab with the 'Headers' sub-tab selected. The request is for '3.php' with a status of 200 OK. The 'Response Headers' section is expanded, showing details like 'Content-Type: text/html; charset=UTF-8' and 'Server: Apache/2.4.57 (Win64) PHP/8.2.10'.

Name	Value
Request URL:	http://localhost/susu_web/3.php
Request Method:	GET
Status Code:	200 OK
Remote Address:	[::1]:80
Referrer Policy:	strict-origin-when-cross-origin
<b>Response Headers</b>	
Connection:	Keep-Alive
Content-Length:	592
Content-Type:	text/html; charset=UTF-8
Date:	Thu, 12 Oct 2023 10:56:06 GMT
Keep-Alive:	timeout=5, max=100
Server:	Apache/2.4.57 (Win64) PHP/8.2.10
X-Powered-By:	PHP/8.2.10
<b>Request Headers</b>	
Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding:	gzip, deflate, br

The screenshot shows the same request in the Network tab, but with the 'Response' sub-tab selected. The response body is displayed as HTML code, showing a table with three columns: 'Индекс', 'Имя', and 'Значение'. The table contains three rows of data.

```
1 <style>
2   table, th, td {
3     border: 1px solid black;
4     border-collapse: collapse;
5   }
6 </style>
7
8 <table>
9   <thead>
10    <td>Индекс</td>
11    <td>Имя</td>
12    <td>Значение</td>
13  </thead>
14
15    <tr>
16      <td>0</td>
17      <td>Иван</td>
18      <td>111</td>
19    </tr>
20    <tr>
21      <td>1</td>
22      <td>Алексей</td>
23      <td>333</td>
24    </tr>
25  </table>
```



Южно-Уральский  
государственный  
университет

Национальный  
исследовательский  
университет

# КРОК

## Спасибо за внимание!



КРОК

Челябинск, ул. Карла Маркса, д. 38