PHP. Краткое введение.

PHP(*PHP: Hypertext Preprocessor)* – скриптовый язык программирования, созданный для генерации HTML-страниц на стороне веб-сервера.

Основные стандартные типы данных:

- Целый тип (integer)
- Вещественный тип (float, double и real – одно и то же в PHP)
- Логический тип (boolean)
- Строковой тип (string)
- Специальный тип NULL
- Ассоциативные массивы(array)

PHP является языком программирования с динамической типизацией(не требует объявления типа переменных/самих переменных)

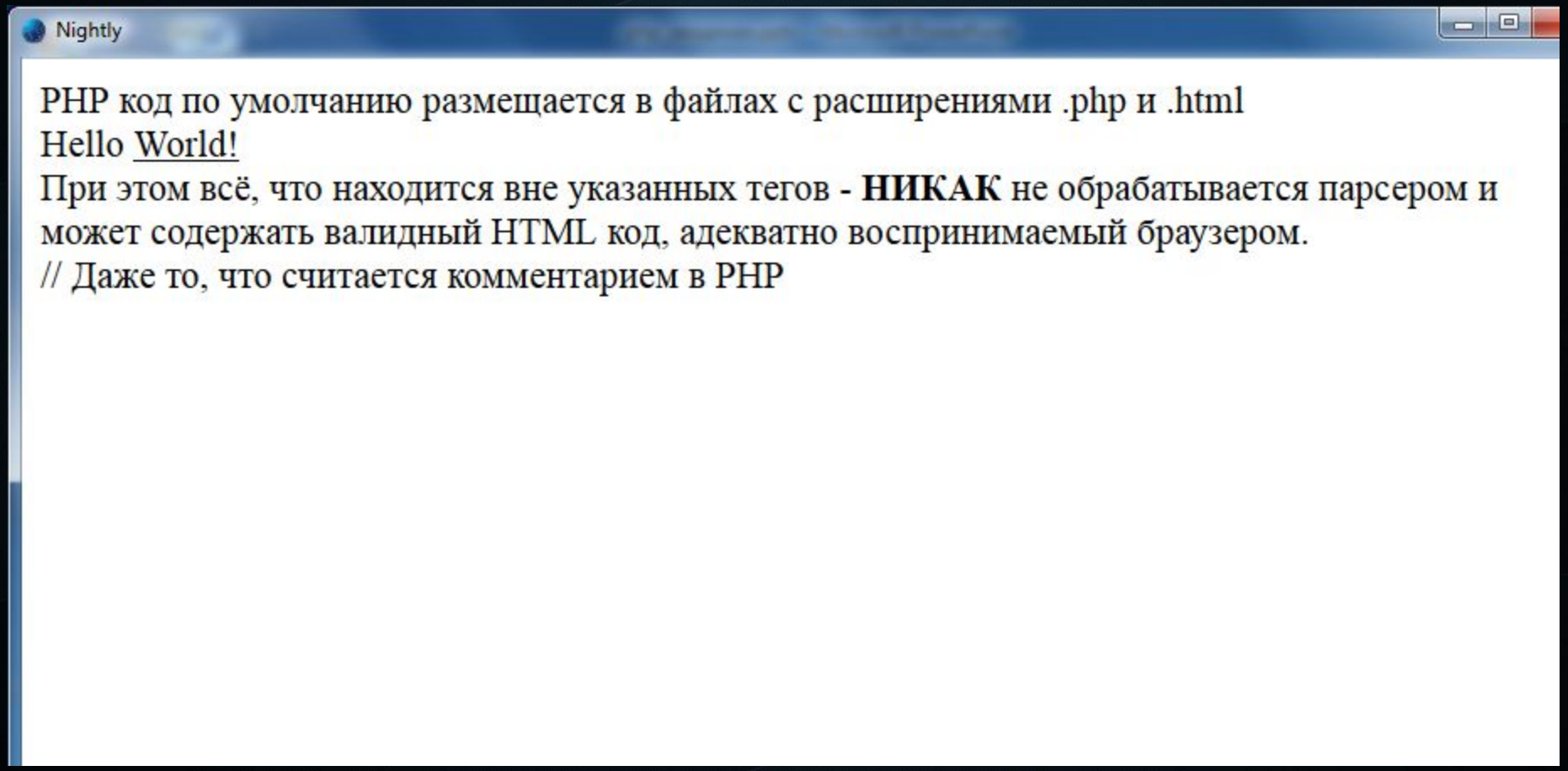
Синтаксис.

Пример:

```
1  PHP код по умолчанию размещается в файлах с расширениями .php
2  и .html</br>
3  <?
4      // Интерпретатор выполняет всё, что находится между
5      // <? .. ? > или <?php .. ? >
6      echo 'Hello <u>World!</u></br>';
7      /*
8          Команда 'echo' не является функцией, поэтому
9              строка выше не обрамлена круглыми скобками.
10     */
11 ?>
12 При этом всё, что находится вне указанных тегов - <b>НИКАК</b> не
   обрабатывается парсером и может содержать валидный HTML код,
   адекватно воспринимаемый браузером. </br>
13 // Даже то, что считается комментарием в PHP
14 <!-- Обычный HTML-комментарий не будет виден в браузере -->
```

Синтаксис.

Результат:



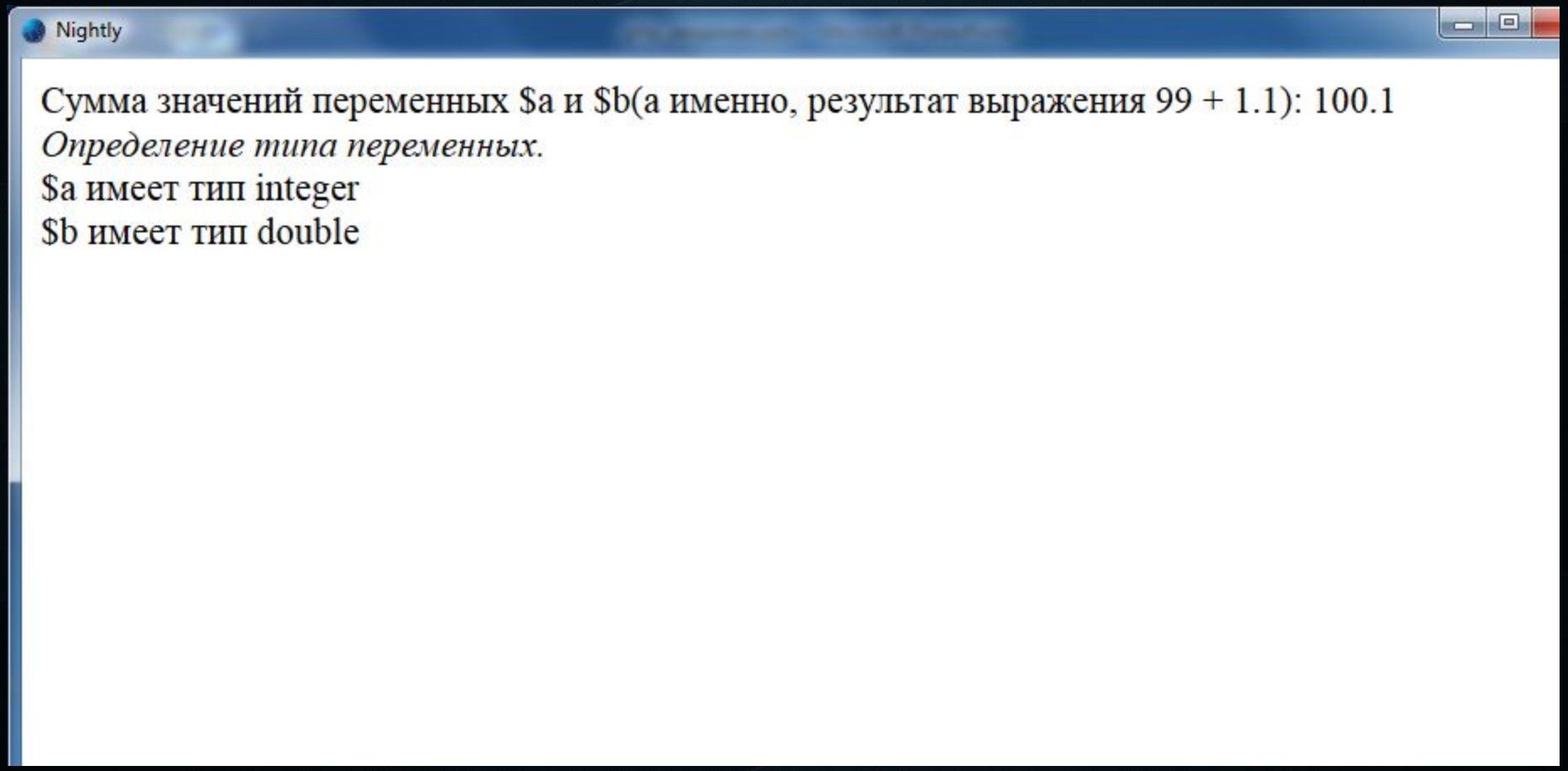
Синтаксис.

Пример:

```php
<?php
    $a = 99;
    $b = (real)1.1;

    $first_str = 'Сумма значений переменных $a и $b(';
    $second_str = "а именно, результат выражения $a + $b): ";

    // Обратите внимание на . (точку) - это оператор склейки строк
    echo $first_str . $second_str . ($a + $b) . '</br>';

    // Стандартные операции над числами: -, /, %, ++, --
    // А так же побитовые операции: and(&&), or(||), xor, !
    // Операции сравнения: >, <, >=, <=, !=(<>), ==, !=, ===, !==

    echo '<i>Определение типа переменных.</i></br>';
    echo '$a имеет тип ' . gettype($a) . '</br>';
    echo '$b имеет тип ' . gettype($b) . '</br>';
?>
```

Синтаксис.

Результат:



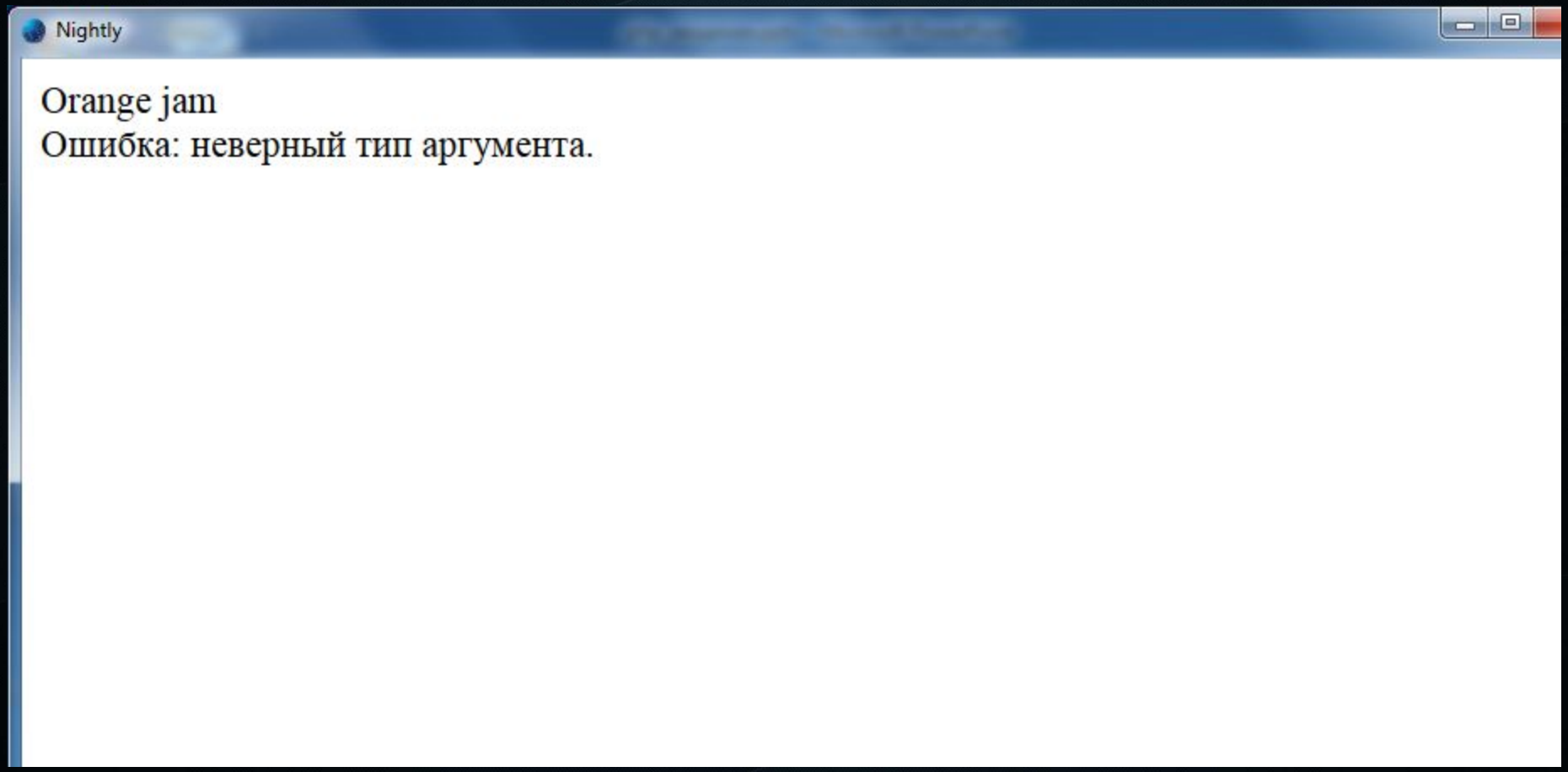
Синтаксис.

Пример:

```php
<?
    function reverse_string($text) {
        if (!is_string($text))
            die('Ошибка: неверный тип аргумента.');

        $len = strlen($text);

        for($i = 0; $i <  $len / 2; $i++) {
            $tmp = $text[$i];
            $text[$i] = $text[$len - $i];
            $text[$len - $i] = $tmp;
        }

        return $text;
    }

    echo reverse_string('maj egnarO') . '</br>';
    echo reverse_string(100) . '</br>';
?>
```

Синтаксис.

Результат:



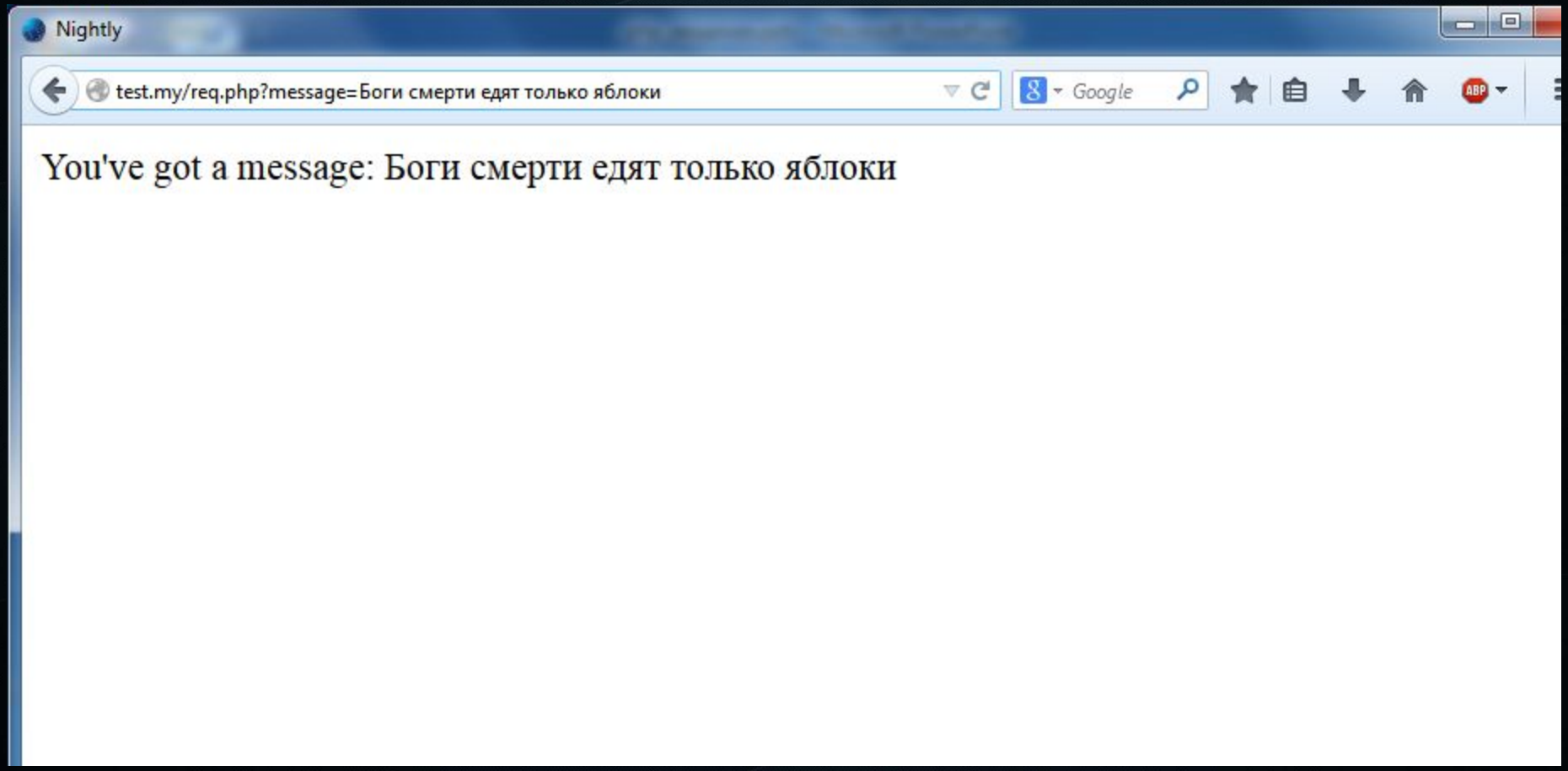
Обработка запросов.

Пример:

```php
<?
    /*
        Суперглобальный массив $_GET содержит все параметры,
        переданные скрипту методом GET. Так же для приёма данных
        существуют суперглобальные массивы $_POST(содержит
        данные, присланные методом POST), и $_REQUEST(сюда
        складываются ВСЕ запросы, присланные обоими методами).
    */

    if ( isset( $_GET['message'] ) ) {
        $message = $_REQUEST['message'];
        echo "You've got a message: $message</br>";
    } else
        echo 'No messages for you</br>';
?>
```

Обработка запросов.

Результат:



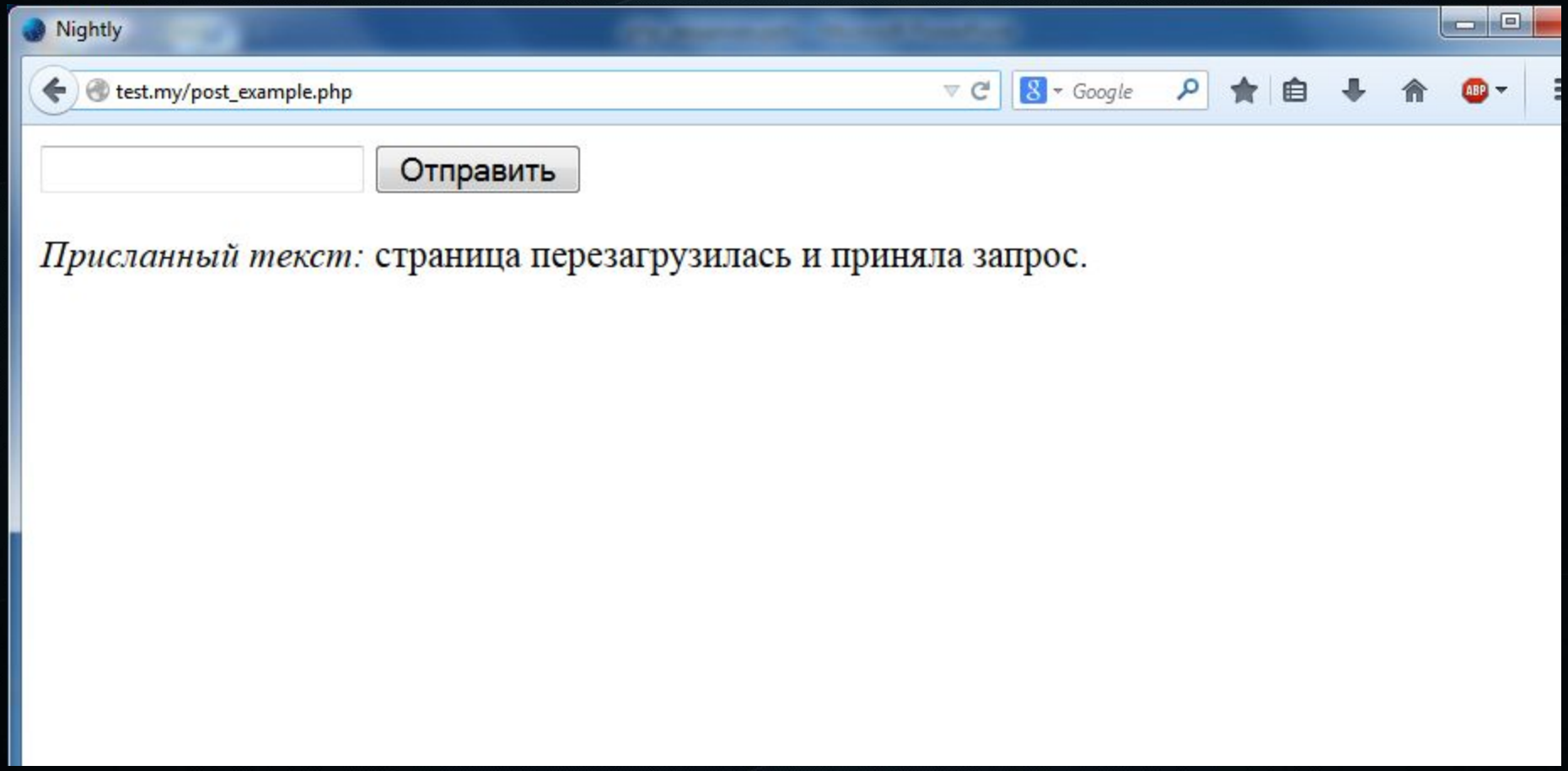
Обработка запросов.

Пример:

```php
<!-- Простейшая HTML-форма, отправляющая POST-запрос. -->
<form action="post_example.php" method="post" accept-charset="
utf-8">
    <input type="text" name="text" value="">
    <input type="submit" name="ok" value="Отправить">
</form>
<?
    if (isset($_REQUEST['text']) && $_REQUEST['text'] != '') {
        $text = $_REQUEST['text'];
        echo "<i>Присланный текст:</i> $text";
    }
?>
```

Обработка запросов.

Результат:



Какому методу отдать предпочтение?

- Если нужен человекочитаемый URL(например, для CEO оптимизации) – выбираем GET.
- Нужно передавать данные, которые не должны быть видимы пользователю (например, логин/пароль) и не должны палиться в логах сервака – выбираем POST.
- Передаётся значительный объём данных или к форме прикреплён файл – выбираем метод POST.

Request-Response.

Данная модель предполагает обмен клиент-сервер сообщениями вида запрос(клиент)-ответ(сервер).

Сообщение состоит из трёх частей: стартовая строка, заголовки, тело. При этом обязательна только стартовая строка.

METHOD URI HTTP/VERSION – так выглядит стартовая строка для запроса. Здесь METHOD – имя метода (GET, POST или другие), URI – идентификатор ресурса, VERSION – используемая версия протокола HTTP

Заголовки – набор пар «имя:значение». Они содержат служебную информацию вроде используемой на странице кодировки и т. д.

Тело содержит передаваемые данные(в случае запроса), и, как правило, содержимое HTML-страницы(в случае ответа).

Request-Response.

Пример запроса:

```
GET /index.php HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru;
   rv: 1.9b5)Gecko/2008050509 Firefox/3.0b5
Accept: text/html
Connection: close
```

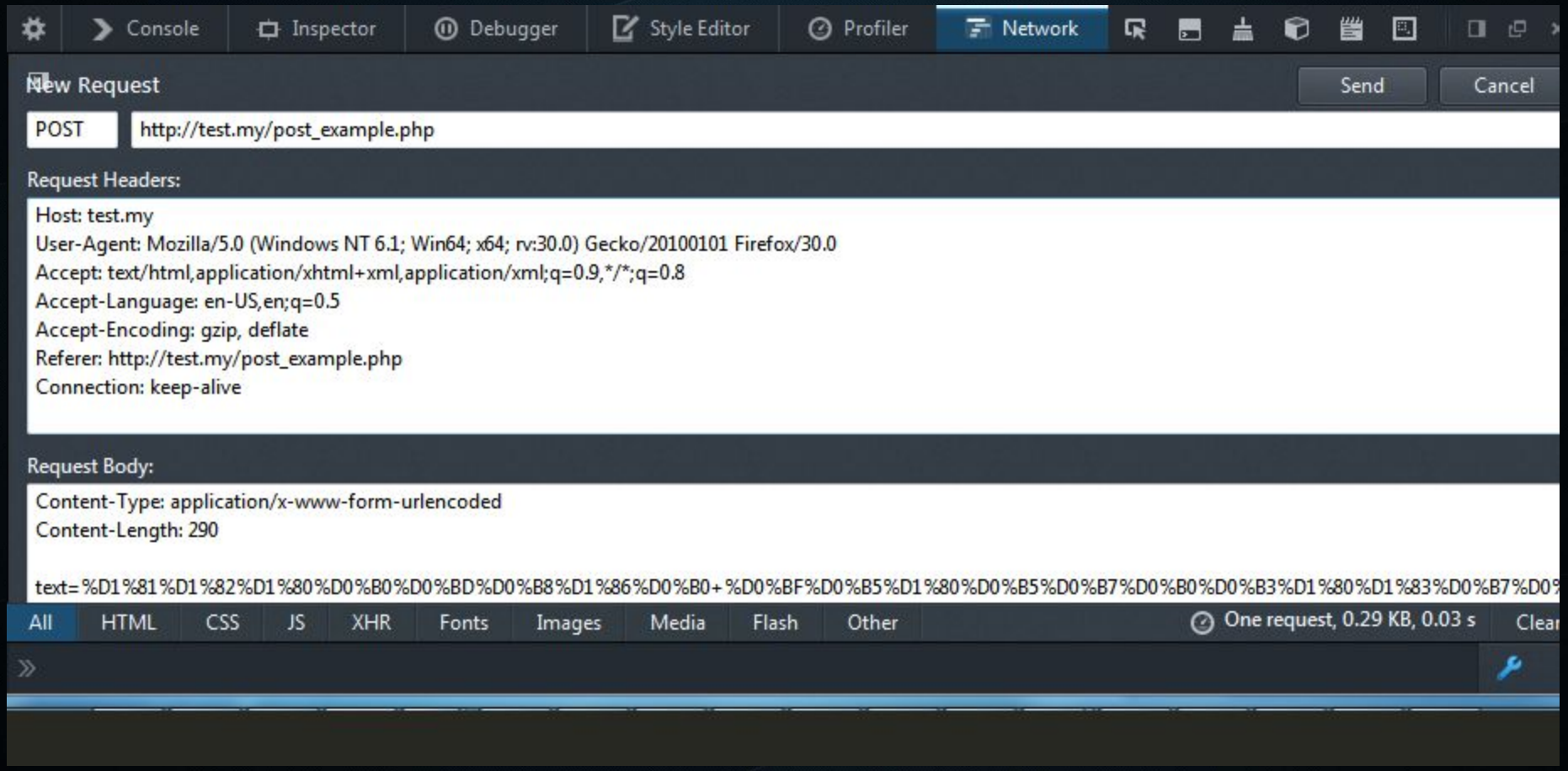
Request-Response.

Пример ответа:

```
HTTP/1.0 200 OK
Server: nginx/0.6.31
Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close
<html>
…. Какой-то html-код ….
</html>
```

Обработка запросов.

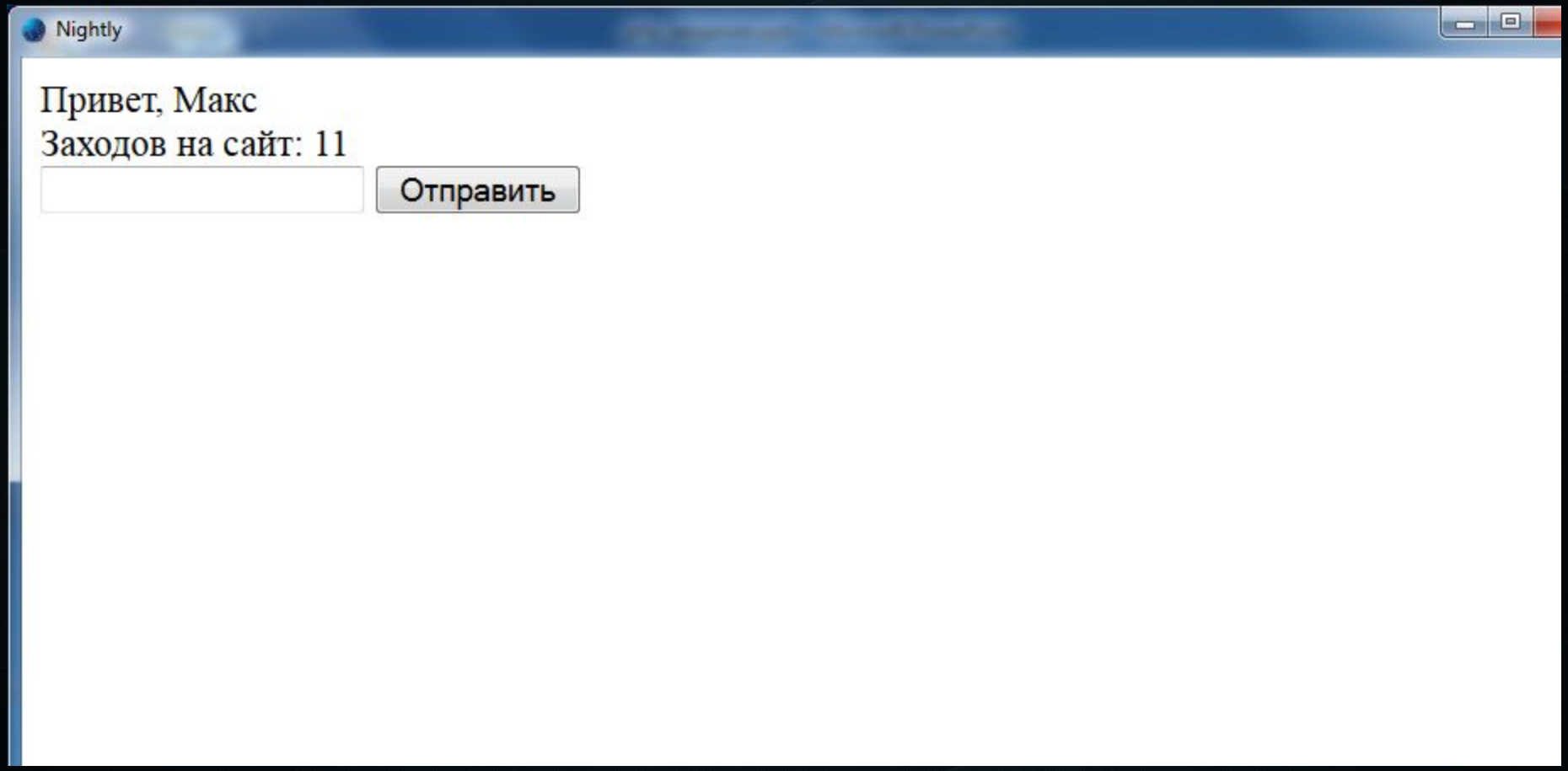
Заголовок и тело запроса как есть:



Cookies.

```php
<?
    // ! Установка кук должна происходить ДО осуществления вывода посредством echo или как-либо иначе
    if (isset($_REQUEST['username']) && $_REQUEST['username'] != '') {
        $username = $_REQUEST['username'];
        setcookie("username", $username, time() + 3600);

        if (!isset($_COOKIE['count']))
            setcookie("count", 0, time() + 3600);
        else
            setcookie("count", $_COOKIE['count'] + 1, time() + 3600);
    }
    // После того, как куки установлены(при необходимости), можем вывести приветствие
    if (isset($_COOKIE['username'])) {
        echo 'Привет, ' . $_COOKIE['username'] . '</br>';

        if (isset($_COOKIE['count']))
           echo 'Заходов на сайт: ' . $_COOKIE['count'] . '</br>';
    }
?>
<form action="cookies.php" method="post" accept-charset="utf-8">
    <input type="text" name="username" value="">
    <input type="submit" name="ok" value="Отправить">
</form>
```

Cookies.



Спасибо за внимание!