Работа с загрузкой фалов в PHP

Загрузка файла PHP 5

PHP легко загружать файлы на сервер.
Однако с легкостью возникает опасность, поэтому всегда будьте осторожны при разрешении загрузки файлов!

Во-первых, убедитесь, что PHP настроен на разрешение загрузки файлов.
В вашем файле «php.ini» найдите директиву file_uploads и установите для неё значение:

```
file_uploads = On
```

Затем создайте форму HTML, которая позволяет пользователям выбирать файл изображения, который они хотят загрузить:

Загрузка файла PHP 5

```
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```

Загрузка файла PHP 5

Некоторые правила для HTML-формы выше:

1. Убедитесь, что в форме используется метод = "post"
2. Форма также нуждается в следующем атрибуте: enctype = "multipart / form-data". Он указывает, какой тип содержимого использовать при отправке формы

Без вышеуказанных требований загрузка файла не будет работать.

Другие примечания:

1. Атрибут type = "file" тега <input> показывает поле ввода в качестве элемента выбора файла, с кнопкой «Обзор» рядом с элементом управления вводом
2. Вышеуказанная форма отправляет данные в файл под названием «upload.php»

Загрузка файла PHP 5

Файл upload.php содержит код для загрузки файла:

```php
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
?>
```

Загрузка файла PHP 5

1. **$target_dir** = "uploads /" - указывает каталог, в который будет помещен файл
2. **$target_file** указывает путь к файлу, который будет загружен
3. **$uploadOk** = 1 еще не используется (будет использоваться позже)
4. **$imageFileType** содержит расширение файла
5. Затем проверяется, является ли файл фактическим или поддельным изображением

Загрузка файла PHP 5

Теперь мы можем добавить некоторые ограничения.

Сначала мы проверим, существует ли файл в папке «uploads». Если это так, отображается сообщение об ошибке, а $uploadOk устанавливается в 0:

```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

Загрузка файла PHP 5

- Теперь мы хотим проверить размер файла. Если файл больше 500 КБ, отображается сообщение об ошибке, а $uploadOk устанавливается в 0:

```
 // Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
```

Загрузка файла PHP 5

Приведенный ниже код позволяет пользователям загружать файлы JPG, JPEG, PNG и GIF. Все остальные типы файлов выдают сообщение об ошибке перед установкой $uploadOk в 0:

```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
```

```php
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

```php
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```

Работа с cookies PHP

PHP 5 Cookies

cookie часто используется для идентификации пользователя. cookie - это небольшой файл, который сервер встраивает в компьютер пользователя. Каждый раз, когда компьютер запрашивает страницу с браузером, он отправляет cookie тоже.

С помощью PHP вы можете создавать и извлекать значения cookie.

cookie создается с помощью функции setcookie ().

Синтаксис

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Требуется только параметр *name* . Все остальные параметры являются необязательными.

PHP 5 Cookies

В следующем примере создается cookie с именем «user» со значением «John Doe». Срок действия cookie истекает через 30 дней (86400 * 30). «/» Означает, что cookie доступен на всем веб-сайте (в противном случае выберите каталог, который вы предпочитаете).

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Затем мы извлекаем значение cookie «user» (используя глобальную переменную $ _COOKIE). Мы также используем функцию isset (), чтобы выяснить, установлен ли cookie

PHP 5 Cookies

Чтобы изменить cookie, просто установите (снова) cookie с помощью функции setcookie ():

```php
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

PHP 5 Cookies

Чтобы удалить cookie, используйте функцию setcookie () с датой истечения в прошлом:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

PHP 5 Cookies

В следующем примере создается небольшой скрипт, который проверяет, включены ли cookie. Сначала попробуйте создать тестовый cookie с помощью функции setcookie (), затем подсчитайте переменную массива $ _COOKIE:

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

Сеансы PHP 5

Сеансы PHP 5

- Сессия - способ хранения информации (в переменных), которая будет использоваться на нескольких страницах.
- В отличие от файла cookie, информация не сохраняется на компьютере пользователя.
- Когда вы работаете с приложением, вы открываете его, делаете некоторые изменения и закрываете его. Это очень похоже на сеанс. Компьютер знает, кто вы. Он знает, когда вы запускаете приложение и когда закончите. Но в Интернете есть одна проблема: веб-сервер не знает, кто вы и что вы делаете, потому что HTTP-адрес не поддерживает состояние.
- Переменные сеанса решают эту проблему, сохраняя информацию пользователя, которая будет использоваться на нескольких страницах (например, имя пользователя, любимый цвет и т. Д.). По умолчанию переменные сеанса сохраняются до тех пор, пока пользователь не закроет браузер.
- Так; Переменные сеанса содержат информацию об одном пользователе и доступны для всех страниц в одном приложении.

Сеансы PHP 5 (Начало сеанса PHP)

Теперь давайте создадим новую страницу под названием «demo_session1.php».

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Сессия начинается с функции session_start ().

Переменные сеанса задаются с помощью глобальной переменной PHP: $ _SESSION.

Сеансы PHP 5(Получить значения переменных сеанса PHP)

Затем мы создаем еще одну страницу под названием «demo_session2.php».

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on
previous page
echo "Favorite color is
" . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is
" . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

На этой странице мы получим доступ к информации о сеансе, которую мы установили на первой странице («demo_session1.php»).

Обратите внимание, что переменные сеанса не передаются отдельно для каждой новой страницы, вместо этого они извлекаются из сеанса, который мы открываем в начале каждой страницы (session_start ()).

Также обратите внимание, что все значения переменной сеанса хранятся в глобальной переменной $ _SESSION

Сеансы PHP 5(Получить значения переменных сеанса)

Другой способ показать все значения переменной сеанса для сеанса пользователя

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>

</body>
</html>
```

В большинстве сеансов пользовательский ключ на компьютере пользователя выглядит примерно так: 765487cf34ert8dede5a562e4f3a7e12.

Затем, когда сеанс открывается на другой странице, php сканирует компьютер на наличие пользовательского ключа.

Если есть совпадение, php обращается к этому сеансу, если нет, он запускает новый сеанс.

Сеансы PHP 5(Изменение переменной сеанса)

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Сеансы PHP 5(Удаление сеанса)

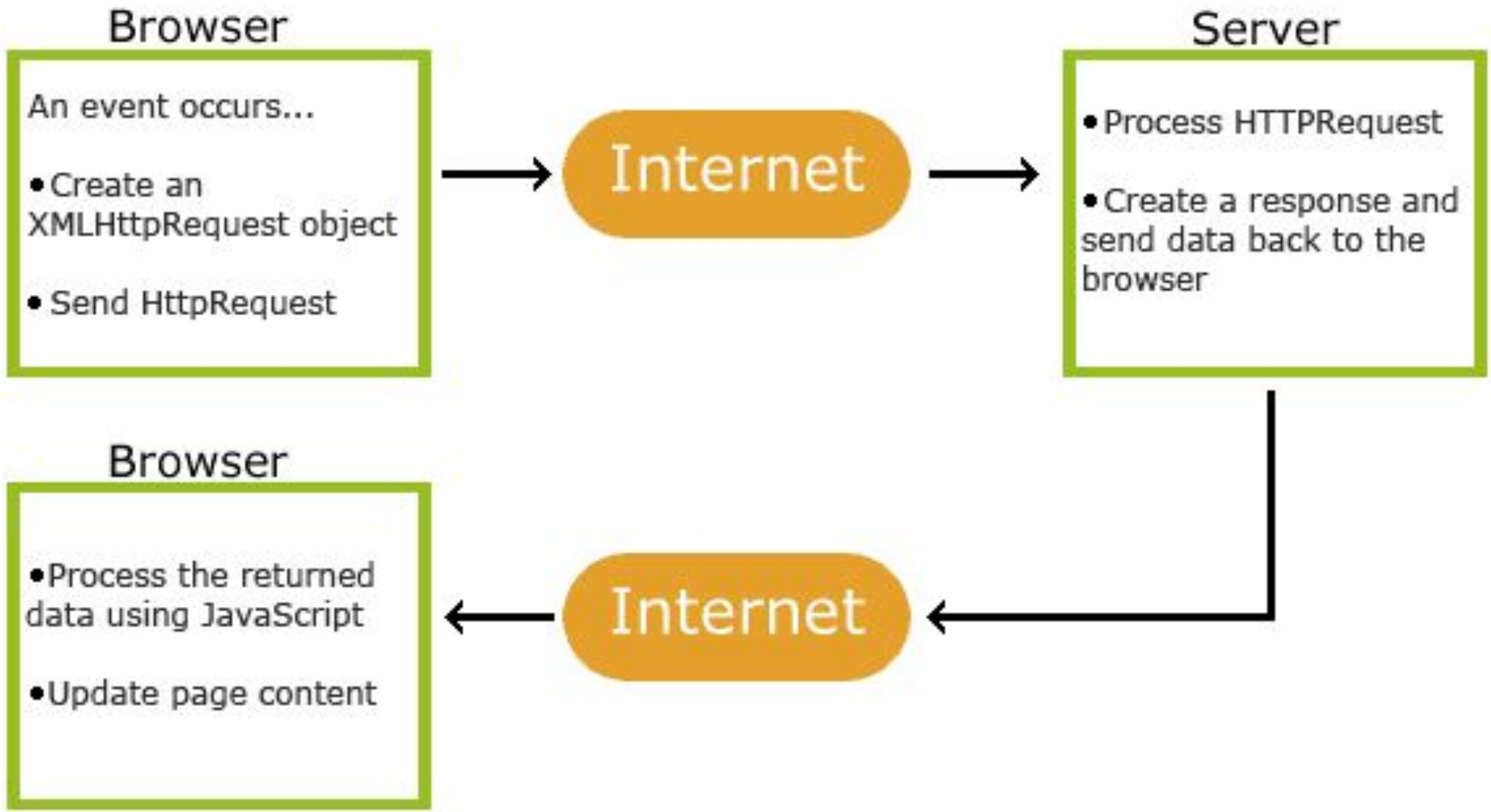
```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

AJAX PHP 5

Введение в AJAX

1. AJAX = асинхронный JavaScript и XML.
2. AJAX - это способ создания быстрых и динамичных веб-страниц.
3. AJAX позволяет обновлять веб-страницы асинхронно, обмениваясь небольшими объемами данных с сервером в фоновом режиме. Это означает, что можно обновлять части веб-страницы без перезагрузки всей страницы.
4. Классические веб-страницы (которые не используют AJAX) должны перезагружать всю страницу, если содержимое должно измениться.
5. Примеры приложений с использованием AJAX: вкладки Google Карты, Gmail, Youtube и Facebook.

Введение в AJAX



Введение в AJAX

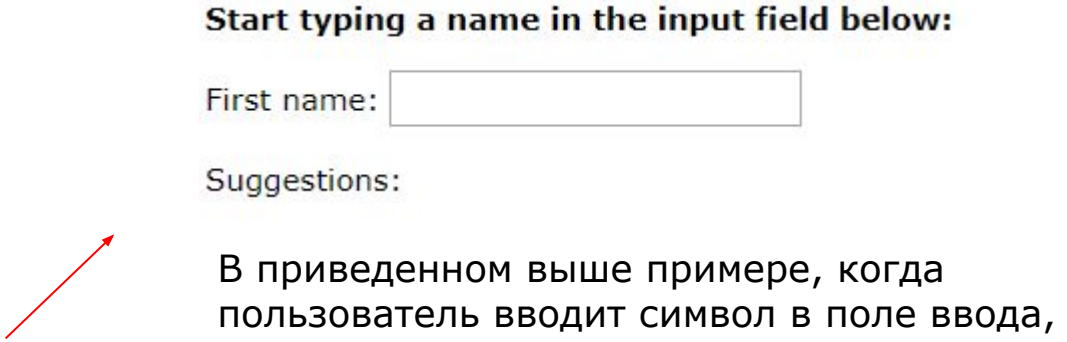
AJAX основан на интернет-стандартах и использует комбинацию:

1. Объект XMLHttpRequest (для обмена данными асинхронно с сервером)
2. JavaScript / DOM (для отображения / взаимодействия с информацией)
3. CSS (для стилизации данных)
4. XML (часто используется как формат для передачи данных)

Приложения AJAX независимы от браузера и платформы!

PHP - AJAX и PHP

```
<html>
<head>
<script>
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("txtHint").innerHTML = this.responseText;
            }
        };
        xmlhttp.open("GET", "gethint.php?q=" + str, true);
        xmlhttp.send();
    }
}
</script>
</head>
<body>

<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```



В приведенном выше примере, когда пользователь вводит символ в поле ввода, выполняется функция «showHint ()». Функция запускается событием onkeyup.

Сначала проверяется, пусто ли поле ввода (str.length == 0). Если это так, очищаем содержимое поля txtHint и выходим из функции. Однако, если поле ввода не пусто, делаем следующее:

1. Создание объекта XMLHttpRequest
2. Создание функции, которая будет выполнена, когда ответ сервера будет готов
3. Отправление запрос на файл PHP (gethint.php) на сервере
4. Обратите внимание, что параметр q добавляется в url (gethint.php?q="+ str)
5. И переменная str содержит значение из поля ввода

PHP - AJAX и PHP

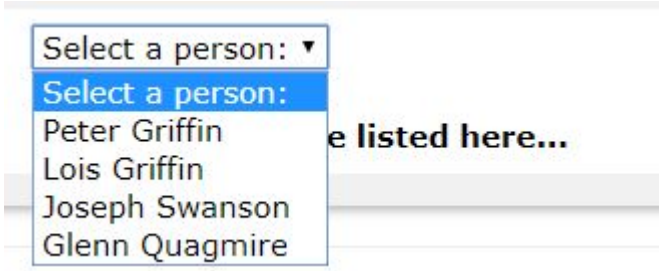
Файл PHP – «gethint.php»

```
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
$a[] = "Johanna";
$a[] = "Kitty";
$a[] = "Linda";
$a[] = "Nina";
$a[] = "Ophelia";
$a[] = "Petunia";
$a[] = "Amanda";
$a[] = "Raquel";
$a[] = "Cindy";
$a[] = "Doris";
$a[] = "Eve";
$a[] = "Evita";
$a[] = "Sunniva";
$a[] = "Tove";
$a[] = "Unni";
$a[] = "Violet";
$a[] = "Liza";
$a[] = "Elizabeth";
$a[] = "Ellen";
$a[] = "Wenche";
$a[] = "Vicky";
```

```
// get the q parameter from URL
$q = $_REQUEST["q"];

$hint = "";

// lookup all hints from array if $q is different from ""
if ($q !== "") {
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name) {
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint === "") {
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
        }
    }
}

// Output "no suggestion" if no hint was found or output correct values
echo $hint === "" ? "no suggestion" : $hint;
?>
```

PHP - AJAX и MySQL

В следующем примере показано, как веб-страница может извлекать информацию из базы данных с помощью AJAX:



id	FirstName	LastName	Age	Hometown	Job
1	Peter	Griffin	41	Quahog	Brewery
2	Lois	Griffin	40	Newport	Piano Teacher
3	Joseph	Swanson	39	Quahog	Police Officer
4	Glenn	Quagmire	41	Quahog	Pilot

PHP - AJAX и MySQL

```
<html>
<head>
<script>
function showUser(str) {
    if (str == "") {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        if (window.XMLHttpRequest) {
            // code for IE7+, Firefox, Chrome, Opera, Safari
            xmlhttp = new XMLHttpRequest();
        } else {
            // code for IE6, IE5
            xmlhttp
= new ActiveXObject("Microsoft.XMLHTTP");
        }
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200)
{

document.getElementById("txtHint").innerHTML = this.response
Text;
            }
        };
        xmlhttp.open("GET","getuser.php?q="+str,true);
        xmlhttp.send();
    }
}
</script>
</head>
<body>
```

```
<form>
<select name="users" onchange="showUser(this.value)">
  <option value="">Select a person:</option>
  <option value="1">Peter Griffin</option>
  <option value="2">Lois Griffin</option>
  <option value="3">Joseph Swanson</option>
  <option value="4">Glenn Quagmire</option>
  </select>
</form>
<br>
<div id="txtHint"><b>Person info will be listed
here...</b></div>

</body>
</html>
```

Сначала проверяем, выбран ли человек. Если ни один человек не выбран (str == ""), очищаем содержимое txtHint и выходим из функции. Если выбран человек, выполните следующие действия:

1. Создание объекта XMLHttpRequest
2. Создайте функцию, которая будет выполнена, когда ответ сервера будет готов
3. Отправка запроса в файл на сервере
4. Обратите внимание, что параметр (q) добавляется в URL-адрес (с содержимым выпадающего списка)

Исходный код в «getuser.php»

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
    width: 100%;
    border-collapse: collapse;
}

table, td, th {
    border: 1px solid black;
    padding: 5px;
}

th {text-align: left;}
</style>
</head>
<body>

<?php
$q = intval($_GET['q']);

$con = mysqli_connect('localhost','peter','abc123','my_db');
if (!$con) {
    die('Could not connect: ' . mysqli_error($con));
}
mysqli_select_db($con,"ajax_demo");
$sql="SELECT * FROM user WHERE id = '".$q."'";
$result = mysqli_query($con,$sql);

echo "<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
<th>Hometown</th>
<th>Job</th>
</tr>";
while($row = mysqli_fetch_array($result)) {
    echo "<tr>";
    echo "<td>" . $row['FirstName'] . "</td>";
    echo "<td>" . $row['LastName'] . "</td>";
    echo "<td>" . $row['Age'] . "</td>";
    echo "<td>" . $row['Hometown'] . "</td>";
    echo "<td>" . $row['Job'] . "</td>";
    echo "</tr>";
}
echo "</table>";
mysqli_close($con);
?>
</body>
</html>
```

1. PHP открывает соединение с сервером MySQL
2. Правильный человек найден
3. Создается таблица HTML, заполняется данными и отправляется обратно в «txtHint»

jQuery - метод Load AJAX ()

Метод jQuery load () - это простой, но мощный метод AJAX.
Метод load () загружает данные с сервера и помещает возвращенные данные в выбранный элемент.

```
$(selector).load(URL,data,callback);
```

- Требуемый URL-адрес указывает URL-адрес, который вы хотите загрузить.
- Необязательный параметр данных указывает набор пар ключей / значений запроса для отправки вместе с запросом.
- Необязательный параметр обратного вызова - это имя функции, которая должна быть выполнена после завершения метода load ().

Содержание нашего файла примера: «demo_test.txt»:

```
<h2>jQuery and AJAX is FUN!!!</h2>
<p id="p1">This is some text in a
paragraph.</p>
```

Следующий код загружает содержимое файла «demo_test.txt» в конкретный элемент :

```
$("#div1").load("demo_test.txt");
```

jQuery - методы AJAX get () и post ()

HTTP-запрос: GET против POST

Два часто используемых метода для запроса-ответа между клиентом и сервером: GET и POST.

1. **GET** - запрашивает данные из указанного ресурса
2. **POST** - отправка данных, подлежащих обработке, на указанный ресурс

GET в основном используется для получения (получения) данных с сервера. **Примечание.** Метод GET может возвращать кэшированные данные.

POST также может использоваться для получения некоторых данных с сервера. Однако метод POST никогда не кэширует данные и часто используется для отправки данных вместе с запросом.

jQuery - метод AJAX get ()

Метод $ .get () запрашивает данные с сервера с запросом HTTP GET.

1. Требуемый параметр URL указывает URL-адрес, который вы хотите запросить.
2. Необязательный параметр обратного вызова - это имя функции, которая будет выполнена, если запрос будет успешным.

В следующем примере используется метод $ .get () для извлечения данных из файла на сервере:

```
$("button").click(function(){
    $.get("demo_test.asp", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

Первым параметром $ .get () является URL-адрес, который мы хотим запросить («demo_test.php»).
Второй параметр - это функция обратного вызова. Первый параметр обратного вызова содержит содержимое запрошенной страницы, а второй параметр обратного вызова содержит статус запроса.

jQuery - метод AJAX post ()

Метод $ .post () запрашивает данные с сервера с помощью HTTP POST-запроса.

```
$.post(URL,data,callback);
```

1. Требуемый параметр URL указывает URL-адрес, который вы хотите запросить.
2. Необязательный параметр данных указывает некоторые данные для отправки вместе с запросом.
3. Необязательный параметр обратного вызова - это имя функции, которая будет выполнена, если запрос будет успешным.

В следующем примере используется метод $ .post () для отправки некоторых данных вместе с запросом:

```
$("button").click(function(){
    $.post("demo_test_post.php",
    {
        name: "Donald Duck",
        city: "Duckburg"
    },
    function(data, status){
        alert("Data: " + data + "\nStatus: " +
status);
    });
});
```

1. Первым параметром $ .post () является URL-адрес, который мы хотим запросить («demo_test_post.php»).
2. Затем мы передаем некоторые данные для отправки вместе с запросом (имя и город).
3. Скрипт PHP в «demo_test_post.php» считывает параметры, обрабатывает их и возвращает результат.
4. Третий параметр - это функция обратного вызова. Первый параметр обратного вызова содержит содержимое запрошенной страницы, а второй параметр обратного вызова содержит статус запроса.

Server Side Events -- события с сервера

Современный стандарт Server-Sent Events позволяет браузеру создавать специальный объект EventSource, который сам обеспечивает соединение с сервером, делает пересоединение в случае обрыва и генерирует события при поступлении данных.

Он, по дизайну, может меньше, чем WebSocket'ы.

С другой стороны, Server Side Events проще в реализации, работают по обычному протоколу HTTP и сразу поддерживают ряд возможностей, которые для WebSocket ещё надо реализовать.

Поэтому в тех случаях, когда нужна преимущественно односторонняя передача данных от сервера к браузеру, они могут быть удачным выбором.

Server Side Events

При создании объекта new EventSource(src) браузер автоматически подключается к адресу src и начинает получать с него события:

```
var eventSource = new EventSource("/events/subscribe");

eventSource.onmessage = function(e) {
  console.log("Пришло сообщение: " + e.data);
};
```

Чтобы соединение успешно открылось, сервер должен ответить с заголовком Content-Type: text/event-stream, а затем оставить соединение висящим и писать в него сообщения в специальном формате:

1. Каждое сообщение пишется после data:. Если после двоеточия есть пробел, то он игнорируется.

2. Сообщения разделяются двумя строками \n\n.

3. Если нужно переслать перевод строки, то сообщение разделяется. Каждая следующая строка пересылается отдельным data:.

Server Side Events

Файл JavaScript-а

```
var eventSource = new EventSource('http://pupkin.ru/stream');

eventSource.onopen = function(e) {
  console.log("Соединение открыто");
};

eventSource.onerror = function(e) {
  if (this.readyState == EventSource.CONNECTING) {
    console.log("Соединение порвалось,
пересоединяемся...");
  } else {
    console.log("Ошибка, состояние: " + this.readyState);
  }
};

eventSource.onmessage = function(e) {
  console.log("Пришли данные: " + e.data);
};
```

Файл PHP

```
<?php

  header('Content-Type: text/event-stream');
  header('Cache-Control: no-cache');

  echo "data: some_data=test";
  flush();

  ?>
```