Передача данных со страницы на другую страницу - JavaScript

12.05.2014, 16:45. Просмотров 3582. Ответов 4

Метки нет (Все метки)

Приветствую всех)

Поставил себе задачу сделать форму полями в виде анкеты, после нажатия на кнопку отправить, все данные из полей <input> отсылались на другую страницу html.

Сделать нужно обязательно на JS, так как нужно что бы работало в автономном режиме, без интернета, и серверов.

Перекопал все статьи, но ничего не вышло.

Вот мой код(проверка на заполнение полей сделана):HTML5Выделить код

<form action='index.html' method="post">

1. Ваше имя:<br>

<input type="text" name="fnm" value="">

<br>

2. Сообщение:<br>

<textarea name="text"></textarea>

<br>

3. E-Mail<br>

<input type="text" name="email" value="">

<br>

4. Контактный телефон<br>

<input type="text" name="phone" value="">

<br>

3. Введите сумму 10+10<br>

<input type="text" name="summa" id='summa' value=""><br>

<input type="hidden" name="hidden" value="ok">

<input type="submit" value="отправить" name="Submit" onClick="return Formdata(this.form)">

</form>

<script language='Javascript' type='text/javascript'>

function Formdata(data){

/\* если не заполнено поле Ваше имя, длина менее 3-x\*/

if (data.fnm != null && data.fnm.value.length < 3 )

{

alert('Заполните поле "Ваше имя"');

return false;}

/\* если не заполнено поле Сообщение \*/

if (data.text != null && data.text.value.length < 3)

{

alert('Заполните поле "Сообщение"');

return false;}

/\* e-mail Юзера \*/

if(data.email != null && data.email.value.length == 0)

{

alert('поле "E-Mail" пустое');

return false;}

if(data.email != null && data.email.value.length < 6)

{

alert('слишком короткий "E-Mail"');

return false;}

if(!(/^\w+[-\_\.]\*\w+@\w+-?\w+\.[a-z]{2,4}$/.test(data.email.value)) )

{

alert("Введите правильный E-Mail адрес");

return false;}

/\* контактный телефон \*/

if(data.phone != null && data.phone.value.length == 0)

{

alert('поле "Контактный телефон" пустое');

return false;}

if(data.phone != null && data.phone.value.length < 5)

{

alert('поле "Контактный телефон" должно содержать минимум пять символов');

return false;}

if(!(/^[0-9-+()s]+z/.test(data.phone.value+"z")))

{

alert('"Контактный телефон" указан неверно');

return false;}

/\* делаем чтобы поле сумма было равно определенному числу \*/

number = document.getElementById("summa");

if (number.value !== '20')

{

alert('"Сумма" не введена или введена неверно');

return false;}

}</script>

HTML

JavaScript

.NET

XML

Services

CSS

PHP

AJAX

SQL

Кросдоменная передача данных между html-страницами

Вячеслав Гринин, April 29, 2010

Итак, представим себе ситуацию, что на некотором сайте в некоторой форме есть поле, в которое нужно ввести логин пользователя, но не свой собственный, а чужой логин, предположим, пользователя, которого надо добавить в друзья или в черный список. Но посетитель может не помнить наизусть, как пишется этот логин, а потому мы сделаем так, чтобы он мог выбрать его из списка, причем список этот должен открыться в отдельном окне и там помимо списка логинов пользователей должны отображаться еще и их фотографии, ФИО, возраст и т.д. Предположим, что мы даже создали такую страницу со списком пользователей. Возникает вопрос – как передать из одного окна браузера в другое окно некоторые данные (в данном случае это – логин пользователя)?

Справочник по JavaScript и объектной документной модели DOM говорит нам, что для открытия нового окна нужно использовать метод window.open(), а для доступа из “дочернего” окна в “родительское” (то есть то, которое и породило новое окно) нужно использовать указатель opener. Рассмотрим этот факт на примере:

index.htm

view plaincopy to clipboardprint?

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Главное окно</title>

<script>

function openWindow() {

window.open("http://easy4web.ru/samples/transfer/popup.htm",

"contents", "toolbar=no", "status=no");

}

</script>

</head>

<body>

<input type="button" onclick="openWindow()"

value="Показать диалог"/><br />

<input type="text" id="data" />

</body>

</html>

popup.htm

view plaincopy to clipboardprint?

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Диалоговое</title>

<script>

function transferData() {

opener.document.getElementById("data").value =

document.getElementById("inp").value;

window.close();

}

</script>

</head>

<body>

<input type="text" id="inp" name="inp"/>

<input type="button" value="OK" onclick="transferData()"/>

</body>

</html>

В главной странице мы видим кнопку, по нажатии на которую открывается новое окно, и TEXTBOX с айдишником data, в него-то и будет попадать текст, введенный в диалоговом окне.

А вот и диалоговое окно, в нем мы видим TEXTBOX и кнопку по нажатию на которую текст введенный в текстовое поле присваивается текстовому полю data из главного окна. А доступ к нему мы и получаем при помощи указателя opener.

Вот здесь (Передача данных между окнами в рамках одного домена) вы можете протестировать работу этого алгоритма.

Здесь все просто, и я не стал бы писать эту статью, если бы хотел рассказать только про это.

А хочу я теперь рассказать о том, как быть если окна эти расположены в разных доменах. Когда такое может произойти? Предположим, есть Ваш форум, а есть специализированный сервис для загрузки и хранения фотографий и вот теперь владелец форума хочет договориться с владельцем фото-сервиса, что на форму он разместит кнопку “Добавить изображение”, которая будет открывать окно созданное в рамках фото-сервиса, в этом окне пользователь загрузит фотографии, а затем по нажатию кнопки “ОК” ссылки на фотографии скопируются в окно редактирования сообщения на форуме. Как видим, здесь есть два окна в разных доменах и нам нужно передать текст из одного окна в другое.

“Что тут сложного?” – спросите вы. А давайте попробуем.

Невозможность передачи данных между окнами в разных доменах

При попытке передачи данных между окнами получим ошибку Error: Access is denied.. Происходит это потому что прежде чем получить доступ к любому методу или свойству объекта opener браузер сравнит домен, в котором существует этот объект и домен, из которого происходит вызов собственно метода или свойства объекта opener. И если доменные имена не совпадают, будет возбуждено исключение “Доступ запрещен”.

Решить эту проблему можно. И мы решим ее без использования каких-либо серверных технологий, только силами JavaScript. Это становится возможным, если мы узнаем еще вот какую тонкость. При изменении свойства location.href в порожденном окне в него загрузится страница, заданная ссылкой, но значение указателя opener не изменится, он так и будет продолжать указывать на породившее его главное окно. А теперь представим себе, что мы в диалоговое окно загрузили все ту же главную страницу, или любую другую но с того же домена, где лежит главная страница, а после этого обратились к объекту opener, теперь домены диалогового окна и объекта opener снова совпадают, а значит исключение возбуждено не будет.

А теперь – время для исходников, иллюстрирующих пример:

index.htm

view plaincopy to clipboardprint?

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Главное окно</title>

<script>

function openWindow() {

var par = "?loc=" + location.href + "&id=data";

window.open("http://easyapi.ru/easy4web/transfer/popup2.htm" + par,

"contents", "toolbar=no", "status=no");

}

function getDataFromUrl() {

var txt\_id, data;

var url = location.href;

var query = url.split("?")[1];

if(query) {

var params = query.split("&");

for(var i = 0; i < params.length; i++) {

var keyval = params[i].split("=");

if(keyval[0] == "data") {

data = keyval[1];

}

if(keyval[0] == "id") {

txt\_id = keyval[1];

}

}

}

if(txt\_id && data) {

opener.document.getElementById(txt\_id).value = data;

close();

}

}

getDataFromUrl();

</script>

</head>

<body>

<input type="button" onclick="openWindow()"

value="Показать диалог"/><br/>

<input type="text" id="data" />

<script>

getDataFromUrl();

</script>

</body>

</html>

Итак, здесь мы видим, что процедура openWindow() все также открывает диалоговое окно, однако теперь она передает ему параметры: loc – указывающий на URL самой порождающей страницы; id – айдишник текстбокса, в который будет вставлен текст из дочернего окна.

А еще мы видим, что после загрузки контента главной страницы вызывается функция getDataFromUrl(), которая проверяет, есть ли GET-параметры id и data (айдишник текстбокса и текст, который мы в него будем вставлять). Параметры эти при изначальной загрузке страницы не заданы, а сначит при первой загрузке страницы код обновляющий содержимое текстбокса не выполнится. А когда он выполнится мы узнаем после того, как рассмотрим исходники диалогового окна.

popup.htm

view plaincopy to clipboardprint?

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Диалог</title>

<script>

function transferData() {

var url = location.href;

var query = url.split("?")[1];

if(query) {

var params = query.split("&");

for(var i = 0; i < params.length; i++) {

var keyval = params[i].split("=");

if(keyval[0] == "loc") {

var loc = keyval[1];

}

if(keyval[0] == "id") {

var txt\_id = keyval[1];

}

}

}

if(loc && txt\_id) {

loclocation.href = loc + "?id=" + txt\_id +

"&data=" + document.getElementById("inp").value;

}

}

</script>

</head>

<body>

<input type="text" id="inp" name="inp"/>

<input type="button" value="OK" onclick="transferData()"/>

</body>

</html>

Здесь при клике по кнопке выполнится функция transferData(), которая сначала извлечет из адресной строки параметры loc и id, те самые, которые мы передали окну при его порождении. И если эти параметры есть – присвоим location.href адрес страницы loc(главной страницы), а в GET-параметры ей передадим id и data(айдишник текстбокса и данные, которые мы в него запишем). При этом в текущее окно (диалоговое) загрузится содержимое главнйо страницы и выполнится та самая функция getDataFromUrl() которая на этот раз извлечет все необходимые параметры из GET-строки и выполнит метод getElementById() объекта opener.document.

Вот и весь секрет кросдоменной передачи данных между окнами.

Вот здесь (Передача данных между окнами в разных доменах) можно протестировать работу вышеприведенного алгоритма.

Замечу, что у всех приведенных здесь решений есть существенный недостаток – корректно передаются только символы английского алфавита, цифры и знаки препинания. Решение этой проблемы оставлю на одну из следующих статей.

В тему:

DOM: вытаскиваем текст вне тегов

Букмарклеты или как скачать аудиофайлы с сайта vkontakte?

AJAX. 1 – Что это такое?

И еще раз про анимированные ссылки

5комментариев

Спасибо за полезную статью, подписался на ваш блог

Adam, July 14, 2010 1:37 am Reply

Огромное спасибо за статью!

Geo, October 5, 2010 2:32 pm Reply

Спасибо за алгоритрм. Для этого нужно вникать в это дело, а у меня терпения нехватает.

collaps, March 17, 2012 1:23 am Reply

а что делать, если нужно передать два и более данных из popup окна?

владимир, September 2, 2013 9:03 am Reply

к элементу data еще добавить GET-параметров, и каждый так же обрабатывать, как и data.

Вячеслав Гринин, September 2, 2013 10:49 am Reply

Оставьте свой комментарий:

Ваше имя

Ваш email\*

Ваш сайт

Текст вашего комментария:

Комментировать

Поиск по блогу:

Найти

Подписаться:

Подписаться

Подписаться письмом

Популярные:

Реализация чата на ASP.NET с использованием Long Polling

Парсинг XML в JavaScript на примере XML-погоды от gismeteo.ru

Определение географического положения по IP-адресу

Что такое PEAR и как его установить?

Используем сервис Gravatar

Облако тегов:

3d ActionScript AJAX api ASP.NET crossdomain CSS DOM dotnet easyAPI Entity Framework Flash framework GeoIP gismeteo haskell http JavaScript jQuery JSON JSONP ORM pda PHP SQL template wap web web-services wordpress xHTML XML XMLHttpRequest XSLT Yesod В помощь фотошоперу верстка дизайн креатиффчег кроссбраузерность парсер позиционирование посещаемость раскрутка социальные закладки

Разное:

Задавайте вопросы

Ищем соавторов

Карта сайта

О блоге

Счетчик:

Передача переменных между страницами.

обсудить на форуме

В этой статье я расскажу о том, как передавать переменные и их значения между страницами. Когда-то для меня это было не такой простой и тривиальной задачей, какой я вижу ее сейчас.

Итак, можно выделить три основных способа передачи переменных между PHP скриптами:

Через массив $\_GET

Через массив $\_POST

Через массив $\_SESSION

Каждый из этих вариантов удобен в конкретных случаях, и обычно, определить каким из массивов пользоваться не сложно. Зачем же передавать данные от одной страницы другой? Примеров масса, здесь и отправка html-формы на электронную почту, и авторизация пользователей, и работа корзины интернет магазина, а также различные фильтры и много много другого. На самом деле владеть методами работы с передачей данных между страницами необходимо любому, кто работает с PHP, без этого ну никуда. Итак, рассмотрим каждый метод передачи данных по очереди:

Передача данных в суперглобальном массиве $\_GET. Как она происходит. Например, у вас есть страница, на которой динамически выводятся какие-либо данные. Чтобы было понятнее, возьмем интернет магазин (я думаю, практически каждый хоть раз сталкивался с написанием интернет магазинов, а если не сталкивался, то обязательно столкнется). Есть страница со списком товаров, и есть страница описания конкретного товара. Вывод описания товара осуществляется по его уникальному идентификатору в базе данных (естественно, я не рассматриваю случай, когда для каждого товара есть своя отдельная html страничка, на которую ведет ссылка из списка товаров, и я надеюсь что вы тоже уже переросли такие сайты). Задача заключается в следующем:

Есть две страницы: список товаров и описание товара. Как определить, описание какого товара показать пользователю, пришедшему со страницы списка товаров?

Решение:

http://домен.ру/product\_list.php - список товаров

http://домен.ру/product\_details.php - описание конкретного товара. Чтобы передать идентификатор на страницу product\_details.php нужно обратиться к ней по ссылке:

http://домен.ру/product\_details.php?product\_id=идентификатор\_продукта

и тогда значение идентификатора товара будет доступно как переменная $\_GET['product\_id']. Ну и дальше можно его использовать в любом месте скрипта product\_details.php. Обратите внимание, что все переменные передаются в адресной строке после знака вопроса (?), чтобы предать несколько переменных их разделяют знаком &. Например:

http://домен.ру/product\_list.php?category\_id=1&order\_by=price&order=DESC

В этом примере мы передали файлу product\_list.php переменные $\_GET['category\_id'], $\_GET['order\_by'] и $\_GET['order']. Зачем это нужно? Например при сортировке списка товаров запрос к базе данных выглядит следующим образом:

SELECT \* FROM products ORDER BY price DESC

Передав переменные order\_by, order и category\_id Можно переписать наш запрос так:

SELECT \* FROM products WHERE category\_id=$\_GET['category\_id'] ORDER BY $\_GET['order\_by'] $\_GET['order'];

Таким образом можно писать фильтры для упорядоченного вывода данных.

Передача переменных в массиве $\_POST. Здесь уже не все так очевидно, как в случае $\_GET, однако сложного тоже ничего нет. Каждый из нас ежедневно сталкивается с html-формами (авторизация, регистрация, поиск и т.д. и т.д.). А как и куда передаются данные, введенные вами в форме? Рассмотрим самую простую форму:

<form action=”on\_submit.php” method=”post”>

<input type=”text” name=”some\_field” value=”” />

<input type=”submit” value=”Отправить” />

</form>

Рассмотрим первую строку. Значение атрибута action определяет файл обработчик формы, т.е. тот файл, которому будут доступны переменные формы. Атрибут action может быть и пустым, тогда обрабатывать форму можно в той же странице, в которой она выводится. Атрибут method определяет, в какой массив будут помещены значения полей формы. Если method=”get”, переменные будут доступны в массиве $\_GET, еслиmethod=”post” - $\_POST. Теперь перейдем непосредственно к полям формы. Мы создали всего одно текстовое поле с атрибутом name=”some\_field”, который и определяет имя переменной, которое будет соответствовать значение поля. Т.е., после нажатия кнопки отправить, доступ к значению поля some\_field можно получить как $\_POST['some\_field']. На самом деле, есть еще один массив $\_REQUEST, который может содержать как массив $\_GET, так и массив $\_POST, но его я встречал только в backend-файлах AJAX приложений, поэтому ничего по поводу того, как его использовать сказать не могу.

Обратимся к случаю, когда обработчик формы и сама форма находятся в одном файле. Вот пример кода:

<?php

if (empty($\_POST)) {

?>

<form action=”on\_submit.php” method=”post”>

<input type=”text” name=”some\_field” value=”” />

<input type=”submit” value=”Отправить” />

</form>

<?php

}

else {

print\_r($\_POST);

}

Здесь мы проверяем, если массив $\_POST пустой, то выводим форму, если нет – распечатываем его содержимое. Обратите внимание, что если вместо функции empty мы будем использовать функцию isset, пример работать не будет, поскольку массив $\_POST существует всегда, а вот пустой он или нет определяем мы с вами.

Передача данных в массиве $\_SESSION. Ну и наконец последний из массивов, который я хотел рассмотреть – массив $\_SESSION. Чтобы получить к нему доступ, нужно в начале каждого файла, который будет использовать этот массив запустить сессию функцией session\_start(). Ну а чтобы добавить переменную в массив нужно всего-лишь записать $\_SESSION['some\_var']='значение\_переменной'. Удобство массива $\_SESSION в том, что он существует, пока вы не закроете браузер. Например, вернемся опять к корзине товаров. Товары в ней описываются массивом $\_SESSION['products'], и к этому массиву можно получить доступ на каждой странице. Массив $\_SESSION удобно использовать при авторизации пользователей. Например, если пользователь авторизован, мы создаем переменную $\_SESSION['login'] равную TRUE или FALSE, и если значение TRUE, то мы разрешаем пользователю просмотр некоторых страниц, если FALSE – то запрещаем.

Сразу оговорюсь, что я не рассматривал использование массива $\_COOKIES. Почему? Может быть кому-то это покажется странным, но я до сих пор им ни разу не пользовался, поэтому не могу сходу описать, как с ним работать.

Подводя итог, можно сказать, что каждый метод удобен в том или ином случае. Переменные массива $\_GET удобны для написания фильтров и условий; $\_POST – для передачи больших массивов, например текстовое описание и т.д. (передача данных методом $\_POST используется в панели управления сайтом для администрирования и добавления данных); $\_SESSION удобно использовать в тех случаях, когда доступ к переменным нам может понадобится на любой из страниц сайта. На этом, я думаю можно закончить. Если возникли вопросы – пишите на форуме.