

Тема. Формы и фреймы в HTML

1. Создание форм в HTML

Форма (формуляр) в *HTML*-документе определяется в виде блока `<FORM> </FORM>`, внутри которого располагаются теги, задающие те или иные формы, а также играющие роль подсказок текстовые строки и обращения к графическим файлам. Внутри формы могут использоваться обычные *HTML*-теги, например, табличные.

В составе тега `<FORM>` записываются атрибуты, определяющие способ и характеристики обработки заключающейся в форме информации, в частности, способ пересылки ее на сервер и кодирования данных в процессе этой пересылки.

Формы в HTML предназначены для передачи пользовательских данных на сервер. С их помощью можно организовать все что угодно – от переписки пользователей до форумов и полноценных веб-приложений. Обработка полученных данных происходит на сервере с помощью специального программного обеспечения, рассмотрение которого выходит далеко за рамки курса HTML. В этой главе мы лишь начнем рассмотрение возможностей, которые предлагают нам формы для организации пользовательских интерфейсов и формирования запросов к серверу.

Полноценная форма состоит из элемента `<form>`, внутри которого располагается любое содержимое (кроме других форм), включающее одно или несколько полей ввода — кнопки, однострочные и многострочные текстовые поля, выпадающие списки и т. д. Функциональность формы определяется рядом атрибутов элемента `<form>`. Одним из них является `name`, в котором можно указать уникальное имя формы. В документе может быть сколько угодно форм, но имена их не должны совпадать. С помощью этого имени подключенные к документу клиентские скрипты могут получать динамический доступ к полям формы еще до ее отправки на сервер.

```
<form name="search" action="/find.cgi">
```

```
<input type="text" name="t" />
```

`<input type="submit" />`

`</form>`

Перед отправкой формы данные из полей приводятся к MIME-типу, соответствующему значению атрибута `enctype`. Затем они отправляются на сервер по адресу, указанному в `action`, методом, выбранным в атрибуте `method`. После этого серверное программное обеспечение может обработать полученную информацию, сформировать запрошенную страницу и отправить ее обратно браузеру.

Значение <code>enctype</code>	Описание
<code>application/x-www-form-urlencoded</code>	Значение по умолчанию. Подходит для большинства случаев. Пробелы и некоторые другие специальные символы кодируются UTF-кодами (напр. %20).
<code>multipart/form-data</code>	Это значение необходимо указывать при отправке файлов на сервер. В форме с предыдущим значением <code>enctype</code> будет отправлено только имя файла, а не его содержимое.
<code>text/plain</code>	Данные передаются в виде обычного текста.

Значение <code>method</code>	Описание
<code>get</code>	Значение по умолчанию. Данные присоединяются к указанному в <code>actionURL</code> в виде пар «поле=значение», разделенных знаком «&». Между запрашиваемым URL и списком этих пар ставится знак «?». Например, «/find.cgi?key=value». Если значения полей содержат символы, не входящие в ASCII, или превышают 100 символов, то необходимо использовать <code>method="post"</code> .
<code>post</code>	Данные передаются в теле запроса. Это никак не отражается на URL, поэтому вы не сможете передать кому-либо ссылку или создать закладку, идентично повторяющую ваш запрос.
<code>put, delete</code>	Эти методы предназначены для отправки запросов на добавление и удаление информации, находящейся по указанному в <code>action</code> адресу. Ключевым отличием от метода <code>post</code> является их идемпотентность, т. е. при повторной загрузке той же страницы (нажатие F5 в браузере) не произойдет повторного действия, поскольку в <code>action</code> указывается не просто скрипт обработчика формы, а именно адрес добавляемой/удаляемой страницы или файла. Эти методы поддерживаются не всеми браузерами и в большинстве случаев вместо них используют <code>post</code> последующим перенаправлением.

Помимо приведенных выше опций, в атрибуте accept-charset можно указать одну или несколько (через пробел) кодировок, допустимых при заполнении формы. Кроме того, в атрибуте target можно обозначить окно или фрейм, в котором будет загружена запрошенная страница.

Основные теги и атрибуты приведены в таблице 1.

Таблица 1. Основные теги и атрибуты создания форм.

Тег	Атрибут	Функция и значение
<Form>	Name	Создание формы: Определяет имя, под которым содержимое поля будет передано обработчику (обязательный атрибут)
	Action	Указывает URL, по которому следует передать введенную информацию (обязательный атрибут)
	Enctype	Определяет способ кодирования содержимого формы при отправке. По умолчанию используется "APPLICATION/X-WWW-FORM-URLENCODED"
	Method	Определяет способ отправки содержимого формы. METHOD=POST - данные из формы передаются серверу отдельно от URL сценария обработчика формы. METHOD=GET - информация из формы для передачи на сервер записывается в конец URL обработчика
	Target	Определение имени окна для результата TARGET="_SELF" - указание на текущую форму TARGET="_PARENT" TARGET="_TOP" – указание на окно самого высшего уровня TARGET="_BLANK" - окно не будет иметь имени TARGET="ИМЯ" - явно указанное имя окна
<Textarea>	Name	Создание поля для ввода нескольких строк текста: Определяет название, которое будет использоваться при идентификации заполненного поля сервером (обязательный параметр)
	Rows	Определяет количество строк текста, видимых на экране
	Cols	Определяет ширину текстового поля - в печатных символах
	Wrap	Определяет способ переноса слов в данной заполняемой форме. WRAP="OFF" – без переноса слов WRAP="VIRTUAL" – деление слов браузером WRAP="PHYSICAL" – перенос слов во всех точках переноса

Таблица 14. Продолжение.

Тег	Атрибут	Функция и значение
<Option>	Value	<i>Описание отдельных пунктов:</i> Устанавливает значение, которое будет передано обработчику. Обязательный элемент
	Selected	Показывает, что данный элемент выбран
<Select>	Multiple	<i>Создание меню выбора:</i> Дает возможность выбора нескольких пунктов меню (по умолчанию - один пункт меню)
	Name	Определяет имя меню, которое будет использоваться при передаче данных на сервер
	Size	Определяет количество видимых пунктов в меню. SIZE=n Если n>1, то результат будет – раскрывающийся список пунктов
<Input>	Name	<i>Создает поле формы (кнопку, поле ввода, чекбокс и т.п.):</i> Указывает имя, под которым содержимое поля будет передано обработчику. Является обязательным
	Size	Устанавливает размер поля ввода текста или пароля в символах
	Maxlength	Устанавливает максимальное количество символов, вводимых в поле ввода текста или пароля
	Value	Устанавливает текст по умолчанию для поля
	Checked	Включает флажок или переключатель
	Type	Устанавливает нужный тип поля ввода

HTML 5 порадовал еще двумя атрибутами элемента <form>. Атрибут autocomplete со значением onсообщает браузеру о необходимости сохранять вводимые в форму данные для организации функции автозаполнения, т. е. чтобы при последующем наборе отображать их во всплывающих подсказках. По умолчанию эта опция включена. Чтобы ее отключить, необходимо указать атрибут autocomplete="off".

Еще одной новинкой стандарта HTML 5 является проверка полей формы на правильность ввода. По умолчанию эта возможность также включена, и отключается она булевым атрибутом novalidate="novalidate". При включенной опции браузер должен проверять, допустимы ли введенные значения в соответствующих элементах, и при необходимости блокировать отправку данных.

2. Создание фреймы в HTML

С помощью тега `<iframe>` можно вставить в страницу другой документ, который отобразится во фрейме – специальной прямоугольной области с собственными полосами прокрутки (scrollbars). Основным атрибутом этого элемента является `src`, в значении которого указывается абсолютный или относительный URL вставляемого документа. HTML5, однако, предлагает альтернативный вариант – вы можете указать исходный код дочерней страницы непосредственно в атрибуте `srcdoc`, без необходимости хранить ее отдельно. Естественно, этот код должен быть экранирован и соответствовать стандарту HTML.

```
<iframe src="frame.html"></iframe>
```

Если вы заполняете оба атрибута, то браузер должен отдать предпочтение `srcdoc`, проигнорировав `src`. В таком случае целесообразно указывать в атрибуте `src` адрес копии документа, содержащегося в `srcdoc`. Это позволит корректно отобразить страницу даже в браузерах, не поддерживающих HTML 5. Несмотря на то, что вся необходимая информация о вставляемом документе задается в атрибутах `<iframe>`, исторически сложилось так, что этот тег все же парный. Это необходимо для обеспечения совместимости с браузерами, которые вообще не поддерживают этот элемент – они просто отобразят его содержимое. Браузеры же с поддержкой `<iframe>` проигнорируют все, что находится между его открывающим и закрывающим тегами.

Стандартом HTML 5 более не поддерживаются атрибуты, отвечающие за графическое оформление фреймов. Даже необходимость отображения полос прокрутки теперь определяется только инструкциями CSS. Однако все еще можно указать ширину и высоту фрейма, присвоив ему атрибуты `width` и `height` соответственно. Значения их, как и в случае с `<img />`, указываются в пикселях или процентах.

Остался также атрибут `name`, позволяющий озаглавить фрейм. Если указать имя фрейма в атрибуте `target` какой-либо ссылки, то документ, на который она ссылается, при нажатии по ней откроется в указанном фрейме.

```
<a href="second.html" target="myFrame"> Эта ссылка откроется во фрейме</a>
```

```
<iframe src="frame.html" name="myFrame" width="100%">
```

Ваш браузер не поддерживает фреймы.

```
</iframe>
```

HTML 5 предоставляет еще два новых атрибута для управления фреймами. Установленный булев атрибут `seamless="seamless"` означает, что браузер должен обрабатывать содержимое вставленного документа так, как будто оно является частью основного документа. В частности, на него будут действовать инструкции CSS, подключенные к странице-родителю. Кроме того, ссылки дочерней страницы будут открываться в основном (родительском) окне браузера, а не в самом фрейме, как это происходит по умолчанию. Также ширина и высота `<iframe>` будут автоматически установлены браузером так, как будто фрейм является обычным блочным элементом. Помните, что не все браузеры поддерживают HTML 5 и атрибут `seamless` в частности. Поэтому, когда все же нужно его применить, рекомендуется подключать к дочернему документу необходимые стили CSS и указывать в элементе `<base />` атрибут `target="_parent"`, чтобы обеспечить правильное функционирование ссылок.

Вторым из упомянутых атрибутов является `sandbox`, позволяющий управлять функциональностью фрейма. Если он не указан, то в `<iframe>` разрешены все возможности HTML 5. В противном случае в подключенном документе запрещается:

- отправка данных из форм на сервер (см. главу «Формы»);
- применение скриптов;
- размещение ссылок, указывающих на другие окна и фреймы (нельзя использовать атрибут `target`);

- применение плагинов с элементами `<embed />` и `<object>`;
- использование атрибута `seamless` во вложенных фреймах;
- доступ к содержимому родительского документа, включая `cookie` и `localStorage`, путем принудительной унификации источника страницы, даже если это фактически один и тот же документ (применяется при необходимости сохранить возможность скриптования в пределах дочернего документа, ограничив доступ к родительскому).

Значением атрибута `sandbox` может быть набор из перечисленных ниже ключевых слов, которые указываются через пробел. Каждое из них снимает некоторые из вышеуказанных ограничений:

- `allow-forms` – разрешает отправку данных с помощью форм.
- `allow-scripts` – разрешает применение скриптов
- `allow-same-origin` – отключает принудительную блокировку доступа к родительскому документу.
- `allow-top-navigation` – разрешает ссылаться на документ верхнего уровня. Остальные окна и фреймы остаются под защитой атрибута `sandbox`

Помните, что использование *sandbox* не гарантирует безопасности при отображении загруженного на сервер пользовательского кода, поскольку злоумышленник может загрузить вредоносную страницу непосредственно в главном окне браузера, обойдя действие атрибута. Для более надежной защиты необходимо обеспечить вывод пользовательских страниц с указанием *MIME*-типа `text/html-sandboxed` вместо `text/html`.