Формы и элементы форм

Как мы видели, обработчики событий почти всегда представляют собой центральный элемент JavaScript-программы. А чаще других используются обработчики событий, связанные с формой и ее элементами. В этой главе вводятся объект Form и различные JavaScript-объекты, представляющие элементы формы.

Объект Form

JavaScript-объект Form представляет HTML-форму. Как уже говорилось, формы доступны в виде элементов массива forms[], являющегося свойством объекта Document. Формы расположены в этом массиве в том же порядке, что и в документе. Следовательно, элемент document.forms[0] ссылается на первую форму документа. На последнюю форму документа можно сослаться посредством следующего выражения:

document.forms[document.forms.length-1]

Наиболее интересное свойство объекта Form – массив elements[], содержащий Java Script-объекты различных типов, представляющие различные элементы ввода формы. Элементы этого массива также располагаются в том же порядке, в котором они расположены в документе. На третий элемент второй формы документа в текущем окне можно сослаться так:

document.forms[1].elements[2]

Остальные свойства объекта Form менее важны. Свойства action, encoding, method и target соответствуют одноименным атрибутам тега <form>. Все эти свойства и атрибуты позволяют контролировать, как данные формы отправляются на вебсервер и где будут отображаться результаты. До появления JavaScript данные формы передавались при щелчке на специальной кнопке Submit, а для сброса значений элементов формы применялась специальная кнопка Reset. JavaScript-объект Form поддерживает два метода, submit() и reset(), служащие той же цели. Вызов метода submit() формы передает данные формы, а вызов reset() сбрасывает значения элементов формы. В дополнение к методам submit() и reset() объект Form предоставляет обработчик события onsubmit, предназначенный для обнаружения факта отправки данных формы, и обработчик события onreset, предназначенный для обнаружения факта сброса значений полей формы. Обработчик onsubmit вызывается непосредственно перед передачей данных формы; он может отменить передачу, вернув значение false. Это дает возможность JavaScript-программе проверить введенные пользователем данные на наличие ошибок, чтобы предотвратить передачу серверу неполных или неверных данных.

Обратите внимание: обработчик onsubmit вызывается только при щелчке на кнопке Submit. Вызов метода submit() формы не приводит к вызову обработчика onsubmit. Обработчик события onreset работает схожим образом. Он вызывается непосредственно перед очисткой формы и может предотвратить очистку, вернув значение false. Это позволяет JavaScript-программе выдать запрос на подтверждение очистки данных, что может быть полезным в случае большой или подробной формы. Это можно сделать с помощью следующего обработчика события:

<form... onreset="return confirm('Действительно удалить ВСЕ данные и начать сначала?')" >

Как и обработчик onsubmit, onreset вызывается только при щелчке на кнопке Reset. Вызов метода reset() формы не приводит к вызову onreset.

Элементы форм

В таблице ниже перечислены типы элементов формы.

В первом столбце таблицы представлены типы элементов формы, во втором – HTML-теги, определяющие элементы данного типа, в третьем – значения свойства type для элементов каждого типа. И наконец, четвертый столбец таблицы предоставляет наиболее важный или наиболее часто используемый обработчик события для элемента данного типа, а пятый – краткое описание каждого типа

Объект HTML-тег Свойство type Событие Описание

Button <input type="button"> или <button type="button"> "button" onclick Кнопка

Checkbox <input type="checkbox"> "checkbox" onclick Флажок

File <input type="file"> "file" onchange Поле для ввода имени файла, загружаемого на вебсервер

Hidden <input type="hidden"> "hidden" Обработчиков событий нет Данные, сохраняемые вместе с формой, но невидимые пользователю

Option <option> Нет Обработчики событий подключаются к объекту Select, а не к отдельным объектам Option Один элемент объекта Select

Password <input type="password"> "password" onchange Поле для ввода п роля (набранные символы невидимы)

Radio <input type="radio"> "radio" onclick Переключатель – одновременно может быть установлен только один

Reset <input type="reset"> или <button type="reset"> "reset" onclick Кнопка, очищающая значения формы

Select <Select> "select one" onchange Список или выпадающее меню, в котором может быть выбран один эле мент (см. также объект Option)

Select <select multiple> "select multiple" onchange Список, в котором может быть выбрано несколько элементов (см. также объект Option).

Submit <input type="submit"> или <button type="submit"> "submit" onclick Кнопка для передачи данных формы

Text <input type="text"> "text" onchange Однострочное поле ввода

Textarea <textarea> "textarea" onchange Многострочное поле ввода

Именование форм и элементов форм

У каждого элемента формы есть атрибут name, который должен быть установлен в HTML-теге, если форма предназначена для отправки серверной программе. У тега <form> также есть атрибут name, который можно установить. Этот атрибут не имеет никакого отношения к отправке форм. Как уже говорилось, он придуман для удобства JavaScript-программистов. Если в теге <form> определен атрибут name, то объект Form, создаваемый для данной формы, как обычно, сохраняется как элемент массива forms[] объекта Document, а также в собственном персональном свойстве объекта Document.

Имя этого нового свойства и представляет собой значение атрибута name. В частности, в примере мы определили форму с помощью следующего тега:

<form name="everything">

Это позволило нам ссылаться на объект Form так:

document.everything

Часто это удобнее, чем нотация массива:

document.forms[0]

Кроме того, использование имени формы делает код позиционно независимым: он останется работоспособным, даже если документ будет реорганизован так, что формы расположатся в нем в ином порядке. В формах именование идет дальше, т. к. у всех элементов формы тоже есть собственный атрибут name. Когда вы даете имя элементу формы, вы создаете новое свойство объекта Form, ссылающееся на этот элемент. Именем этого свойства становится значение атрибута. Следовательно, вы можете следующим

образом сослаться на элемент с именем «zipcode», находящийся на форме с именем «address»:

document.address.zipcode

Разумно выбрав имена, можно сделать синтаксис более элегантным, чем альтернативный синтаксис, в котором индексы массивов жестко «зашиты» в исходные тексты программы (и зависят от позиции):

document.forms[1].elements[4]

Чтобы в группе переключателей мог быть установлен только один, всем входящим в группу элементам должны быть даны одинаковые имена. Также, общепринятой, хотя и не обязательной, практикой также является назначение одинаковых атрибутов name группе взаимосвязанных флажков. Когда несколько элементов формы имеют одинаковые значения атрибута name, интерпретатор JavaScript просто помещает эти элементы в массив с указанным именем. Элементы массива располагаются в том порядке, в котором они присутствуют в документе. Поэтому на объекты Radio (переключатель) из примера можно ссылаться так:

document.everything.browser[0]

document.everything.browser[1]

document.everything.browser[2]

Общие свойства элементов форм

У всех (или у большинства) элементов форм есть общие свойства, перечисленные далее.

type Доступная только для чтения строка, идентифицирующая тип элемента формы.

form Ссылка на объект Form, в котором содержится этот элемент.

name Доступная только для чтения строка, указанная в HTML-атрибуте name.

value Доступная для чтения и записи строка, задающая «значение», содержащееся в элементе формы или представляемое им.

Для элементов Text (однострочное текстовое поле) и Textarea (многострочное текстовое поле) это свойство содержит введенный пользователем текст.

Для элементов Button это свойство задает отображаемый на кнопке текст, который иногда требуется изменять из сценария.

Свойство value для элементов Radio (переключатель) и Checkbox (флажок) не редактируется и ни как не представляется пользователю.

Обработчики событий элементов форм

Большинство элементов форм поддерживают следующие обработчики событий:

onclick Вызывается при щелчке левой кнопкой мыши на данном элементе. Этот обработчик особенно полезен для кнопок и сходных с ними элементов формы.

onchange Вызывается, когда пользователь изменяет значение, представляемое элементом, например вводит текст или выбирает пункт в списке. Кнопки и сходные с ними элементы обычно не поддерживают этот обработчик события, т. к. у них нет значения, которое можно редактировать. Обратите внимание: этот обработчик не вызывается, например, при каждом нажатии пользователем очередной клавиши в ходе заполнения поля ввода. Он вызывается, только когда пользователь изменяет значение элемента и затем перемещает фокус ввода к какому-либо другому элементу формы. То есть вызов этого обработчика события указывает на завершенное изменение. onfocus Вызывается, когда элемент формы получает фокус ввода. onblur Вызывается, когда элемент формы теряет фокус ввода.

Элементы Button, Submit и Reset

Элемент Button (кнопка) – один из наиболее часто используемых элементов формы, т. к. он предоставляет понятный визуальный способ вызова пользователем какого-либо запрограммированного сценарием действия. Элемент Button не имеет собственного поведения, предлагаемого по умолчанию, и не представляет никакой пользы без обработчика события onclick (или другого события). Свойство value элемента Button управляет текстом, появляющимся на самой кнопке. Это свойство можно установить, изменив текст (только «чистый» текст, а не HTMLтекст), присутствующий на кнопке, и это часто бывает полезно.

Обратите внимание: гиперссылки предоставляют такой же обработчик события onclick, что и кнопки, и любой объектКнопку можно заменить гиперссылкой, выполняющей при щелчке такое же действие.

Элементы Submit и Reset очень похожи на элемент Button, но имеют связанные с ними действия, предлагаемые по умолчанию (передача или очистка формы).

Так как у этих элементов имеются действия по умолчанию, они могут быть полезны даже без обработчика событий onclick. Если обработчик события onclick возвращает false, стандартное действие этих кнопок не выполняется. Обработчик onclick элемента Submit позволяет проверить введенные в форме значения, но обычно это делается в обработчике onsubmit са мой формы. Создавать кнопки Button, Submit и Reset можно при помощи тега <button> вместо традиционного тега <input>. Тег <button> более гибок, т. к. выводит не просто текст, заданный атрибутом value, а любое HTMLсодержимое (форматированный текст и/или изображения), присутствующее между тегами <button> и </button>. Тег <button> не обязательно должен находиться в пределах тега <form> и может размещаться в любом месте HTML-документа. Объект Button, созданный тегом <button>, формально отличается от созданного тегом <input>, но оба имеют одинаковые значения поля type, да и в остальном их поведение довольно схоже. Основное отличие состоит в том, что тег <button> не использует значение свойства value для определения внешнего вида кнопки, т. е. внешний вид кнопки нельзя изменить путем установки свойства value.

Элементы Checkbox и Radio

Элементы Checkbox (флажок) и Radio (переключатель) имеют два визуально различимых состояния: они могут быть либо установлены, либо сброшены. Пользователь может изменить состояние такого элемента, щелкнув на нем. Переключатели объединяются в группы связанных элементов, имеющих одинаковые значения HTML-атрибутов name. При установке одного из переключателей в группе другие переключатели сбрасываются. Флажки тоже часто составляют группы с одним значением атрибута name, и ссылаясь на них по имени, необходимо помнить, что объект, на который вы ссылаетесь, представляет собой массив элементов с одинаковыми именами. В примере18.1 имеется три объекта Checkbox с именами extras, и мы можем ссылаться на массив из трех этих элементов следующим образом:

document.everything.extras

Для ссылки на отдельный флажок мы должны указать его индекс в массиве:

document.everything.extras[0] // Первый элемент формы с именем "extras"

У флажков и переключателей есть свойство checked. Это доступное для чтения и записи логическое значение определяет, установлен ли в данный момент эле мент. Свойство defaultChecked представляет собой доступное только для чтения логическое значение, содержащее значение HTMLатрибута checked; оно определяет, должен ли элемент устанавливаться при первой загрузке

страницы. Флажки и переключатели сами не отображают какой-либо текст и обычно выводятся вместе с прилегающим к ним HTML-текстом (или со связанным тегом <label>).

Элементы Text, Textarea, Password и File

Элемент Text (однострочное текстовое поле) применяется в HTMLформах и JavaScript-программах, пожалуй, чаще других. В однострочное текстовое поле пользователь может ввести короткий текст. Свойство value представляет текст, введенный пользователем. Установив это свойство, можно явно задать выводимый текст. Обработчик события onchange вызывается, когда пользователь вводит новый текст или редактирует существующий, а затем указывает, что он завершил редактирование, убрав фокус ввода из текстового поля. Элемент Textarea (многострочное текстовое поле) очень похож на элемент Text за исключением того, что разрешает пользователю ввести (а JavaScriptпрограмме вывести) многострочный текст. Многострочное текстовое поле создается тегом <textarea>, при этом синтаксис существенно отличается от синтаксиса тега <input>, используемого для создания однострочного текстового поля. (Подробнее об этом см. в разделе с описанием элемента Textarea в четвертой части книги.) Тем не менее эти два типа элементов ведут себя очень похожим образом. Свойство value и обработчик события onchange для многострочного текстового поля мо но использовать точно так же, как для однострочного.

Элемент Password (поле ввода пароля) – это модификация однострочного тексто вого поля, в котором вместо вводимого пользователем текста отображаются символы звездочек. Эта особенность позволяет вводить пароли, не беспокоясь о том, что другие прочитают их через плечо. Обратите внимание: элемент Password защищает введенные пользователем данные от любопытных глаз, но при отправке данных формы эти данные никак не шифруются (если только отправка не выполняется по безопасному HTTPScoединению) и при передаче по сети могут быть перехвачены.

И наконец, элемент File (поле ввода имени файла) предназначен для ввода пользователем имени файла, который должен быть загружен на сервер. По существу, это однострочное текстовое поле, совмещенное со встроенной кнопкой, выводящей диалоговое окно выбора файла. У элемента File, как и у однострочного текстового поля, есть обработчик события onchange. Однако в отличие от текстового поля ввода, свойство value объекта File доступно только для чтения. Это не дает злонамеренным JavaScriptпрограммам обмануть пользователя, загрузив файл, не предназначенный для отправки на сервер.

Элементы Select и Option

Элемент Select представляет собой набор вариантов (представленных элементами Option), которые могут быть выбраны пользователем. Броузеры обычно отображают элементы Select в виде выпадающих меню или списков. Элемент Select может работать двумя сильно различающимися способами, и значение свойства type зависит от того, как он настроен. Если в теге <select> определен атрибут multiple, пользователь может выбрать несколько вариантов, и свойство type объекта Select равно "select multiple". В противном случае, если атрибут multiple отсутствует, может быть выбран только один вариант, и свойство type равно "select one". В некотором отношении элемент с возможностью множественного выбора похож на набор флажков, а элемент без такой возможности – на набор переключателей. Элемент Select отличается от них тем, что единственный элемент Select предоставляет полный набор вариантов выбора, которые задаются в HTML с помощью тега <option> и представлены в JavaScript объектами Option, хранящимися в массиве options[] элемента Select.

Поскольку элемент Select предлагает набор вари антов, у него нет свойства value, как у других элементов формы. Вместо этого, о чем мы вскоре поговорим, свойство value определяется в каждом объекте Option, содержащемся в элементе Select. Когда пользователь выбирает вариант или отменяет выбор, элемент Select вызывает свой обработчик события onchange.

Для элементов Select без возможности множественного выбора значение доступного только для чтения свойства selectedIndex равно номеру выбранного в данный момент варианта. Для элементов Select с возможностью множественного выбора одного свойства selectedIndex недостаточно для представления полного набора выбранных вариантов. В этом случае для определения выбранных вариантов следует в цикле перебрать элементы массива options[] и проверить значения свойства selected каждого объекта Option. Помимо свойства selected у элемента Option есть свойство text, задающее строку текста, которая отображается в элементе Select для данного варианта. Можно установить значение этого свойства так, чтобы изменить предлагаемый пользователю текст. Свойство value также представляет доступную для чтения и записи строку текста, который отсылается на вебсервер при передаче данных формы. Даже если вы пишете чисто клиентскую программу, и передачи данных вашей формы никогда не происходит, свойство value (или соответствующий ей HTML атрибут value) может применяться для хранения данных, которые потребуются при выборе пользователем определенного варианта. Обратите внимание, что элемент Option не определяет связанных с формой обработчиков событий; используйте вместо этого обработчик onchange соответствующего элемента Select. Помимо

установки свойства text объектов Option есть другие способы динамического изменения выводимых в элементе Select вариантов.

Практическое задание

Создать форму, разместить на ней элементы различных типов.

Добавить зависимости между элементами (например - один элемент разблокируется для ввода, когда другой заполнен или отмечен).

Добавить проверку значений формы, выводить сообщение при некорректном заполнении формы.

По нажатию submit-кнопки собирать данные с формы в один объект и выводить в консоль.

Изменять стиль submit-кнопки в зависимости от заполненности формы.