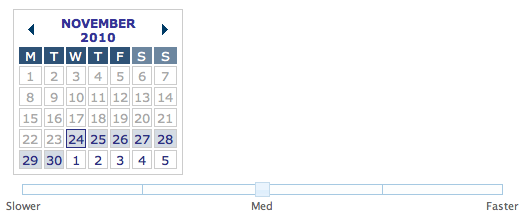
**Введение**

Язык HTML5 содержит много новых свойств, которые должны существенно облегчить создание форм в Web, и сделать их значительно более мощными и совместимыми в Web. Эта статья делает краткий обзор некоторых новых элементов управления форм и функций, которые были введены.

**Плохая форма?**

Посмотрим правде в лицо – формы HTML всегда были проблемой. Они не слишком интересны для разметки, и могут требовать кропотливой работы для оформления – особенно, если вы хотите, чтобы они были совместимы с разными браузерами и соответствовали общему виду вашего сайта. И они могут создавать трудности при заполнении для конечных пользователей, даже если они были созданы с заботой и вниманием, чтобы сделать их как можно удобнее и доступнее. Создание хорошей формы состоит в большей степени в ограничении ущерба, чем в обеспечении приятного взаимодействия пользователя.

Ранее, когда HTML 4.01 стал стабильной рекомендацией, сеть web была по большей части стабильным местом. Да, существовала странная форма для комментариев или сценарий для гостевой книги, но обычно web-сайты предназначались посетителям для простого чтения. С тех пор сеть web существенно изменилась. Для многих людей браузер уже стал окном в мир сложных приложений на базе web, которые стараются обеспечить взаимодействие почти на уровне настольного компьютера.



Несколько сложных элементов управления формой, созданных с помощью JavaScript

Чтобы удовлетворить потребность в более развитых элементах управления, которые нужны для таких приложений, разработчики использовали библиотеки JavaScript и рабочие среды (такие как jQueryUI или YUI). Эти решения, конечно, существенно развились за прошедшие годы, добавляя богатую функциональность на страницы Web, начиная включать даже поддержку считывателей экрана с помощью таких технологий как WAI-ARIA. Но, по сути, они являются обходными путями, чтобы компенсировать ограниченные возможности элементов управления форм, которые имеются в HTML. Они "имитируют" более сложные виджеты (такие как выбор даты и ползунки), создавая дополнительный слой разметки (не всегда семантической), и дополнительный код JavaScript поверх страниц.

HTML5 нацелен на стандартизацию некоторых из наиболее распространенных элементов управления форм, делая их отображение в браузерах простым, исключая необходимость решений, которые существенно используют сценарии.

**Знакомство с примером**

Чтобы проиллюстрировать некоторые новые свойства, статья содержит элементарный пример формы HTML5 (<http://dev.opera.com/articles/view/new-form-features-in-html5/html5-forms-example.html>). Он не предназначен для представления "реальной" формы (так как трудно было бы, пожалуй, найти ситуацию, где могут потребоваться все новые свойства в одной форме), но она должна показать, как выглядят и ведут себя различные новые возможности.

**Примечание**: Внешний вид новых элементов управлении форм и контрольные сообщения варьируются в различных браузерах, поэтому оформление форм для согласованного представления в различных браузерах будет по-прежнему требовать тщательного рассмотрения.

**Новые элементы управления форм**

Так как формы являются основным инструментом ввода данных в приложениях Web, а требуемые для сбора данные стали более сложными, возникла необходимость создать элементы ввода с дополнительными возможностями, чтобы собирать эти данные с дополнительной семантикой и лучшим определением, и предоставить более простое, более эффективное управление ошибками и проверкой.

**<input type="number">**

Первый новый тип элемента ввода, который мы рассмотрим, будет тип ввода number (числа):

<input type="number" … >

Он создает специальный вид поля ввода для числа – в большинстве поддерживающих браузеров оно появляется как текстовое поле ввода со счетчиком, который позволяет увеличивать и уменьшать его значение.

Тип элемента ввода number

Тип элемента ввода number

**<input type="range">**

Создание элемента управления ползунка, позволяющего выбирать в диапазоне значений, было обычно сложным, семантически неясным предложением, но в HTML5 все стало легко – вы используете просто тип ввода range (диапазон):

<input type="range" … >

Тип элемента ввода range

Тип элемента ввода range

Отметим, что, по умолчанию, этот элемент ввода обычно не показывает выбранное в данный момент значение, или даже диапазон значений, который он охватывает. Автору нужно будет предоставить их с помощью других средств – например, для отображения текущего значения можно было бы использовать элемент вывода вместе с некоторым кодом JavaScript для обновления вывода, когда пользователь взаимодействует с формой:

<output onforminput="value=weight.value"></output>

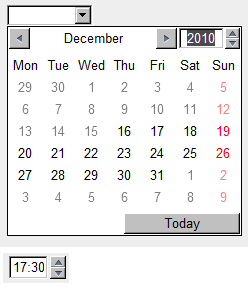
**<input type="date"> и другие элементы управления датой/временем**

HTML5 имеет ряд различных типов элементов ввода для создания сложных средств выбора даты/времени, например, показанное ниже средство выбора даты имеется почти на каждом существующем сайте бронирования самолета/поезда. Такие средства обычно создавались с помощью несемантического путаного кода, поэтому здорово, что теперь существует стандартизованный простой способ сделать это. Например:

<input type="date" … >

<input type="time" … >

Соответственно эти коды создают полностью функциональное средство выбора даты, и элемент ввода текста, содержащий разделитель для часов, минут и секунд (в зависимости от указанного атрибута step ), который позволяет вводить только значение времени.



Типы элементов ввода date и time

Но на этом все не кончается – доступен ряд других связанных типов элементов ввода:

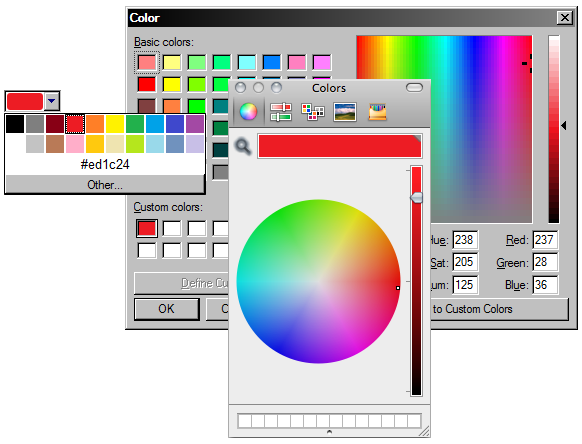
datetime: объединяет функции обоих рассмотренных выше элементов, позволяя выбрать дату и время.

month: позволяет выбрать месяц, сохраняя его внутренне как число между 1-12, хотя различные браузеры могут предоставлять более развитый механизм выбора, например, как прокручиваемый список названий месяцев.

week: позволяет выбрать неделю, сохраняемую внутренне в формате 2010-W37 (37 неделя 2010 года), и выбирать с помощью элемента выбора даты, аналогичного тому, что мы уже видели.

**<input type="color">**

Этот тип элемента ввода предоставляет элемент выбора цвета. Реализация в Opera позволяет пользователю выбрать из набора цветов, ввести шестнадцатеричные значения прямо в текстовое поле, или вызвать собственную процедуру выбора цвета операционной системы.



Элемент ввода color, и собственные инструменты выбора цвета в Windows и OS X

**<input type="search">**

Тип элемента ввода search является, возможно, ничем иным, чем иначе оформленным текстовым элементом ввода. Предполагается, что браузеры будут оформлять эти элементы ввода таким же образом, как оформляются любые функции поиска в конкретной операционной системе. Тем не менее, помимо этих чисто эстетических соображений, важно отметить, что явная разметка полей поиска открывает для браузеров, вспомогательных технологий, и автоматизированных сканеров возможность сделать в будущем что-то полезное с помощью этих элементов ввода – например, браузер мог бы, возможно, предложить пользователю возможность автоматически создать индивидуальный поиск для определенного сайта.

Элемент ввода search, как он выглядит в браузере Opera в OS X

Элемент ввода search, как он выглядит в браузере Opera в OS X

### Элемент <datalist> и атрибут list

До сих пор мы привыкли использовать элементы <select> и <option> для создания раскрывающихся списков вариантов выбора для пользователей. Но как быть, если мы захотим создать список, который позволял бы пользователям выбирать из списка предложенных вариантов, а также позволял вводить свои собственные? Это требует кропотливого создания сценария – но теперь можно просто использовать атрибут list, чтобы соединить обыкновенный элемент ввода со списком вариантов, определенным в элементе <datalist>.

<input type="text" list="mydata" … >

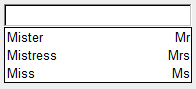
<datalist id="mydata">

<option label="Mr" value="Mister">

<option label="Mrs" value="Mistress">

<option label="Ms" value="Miss">

</datalist>



Создание элемента ввода с предложениями с помощью datalist

#### <input type="tel">, <input type="e-mail"> и <input type="url">

Как предполагают названия, эти новые типы элементов ввода имеют отношение к телефонным номерам, адресам e-mail, или URL. Браузеры будут отображать их как обычные текстовые поля ввода, но явное указание, какой вид текста ожидается в этих полях, играет важную роль в проверке формы на стороне клиента. Кроме того, на некоторых мобильных устройствах браузер будет переключаться из своей обычной экранной клавиатуры для ввода обычного текста в более контекстно-подходящие варианты. Здесь также возможно, что в будущем браузеры будут использовать дополнительные преимущества этих явно размеченных элементов ввода, чтобы предложить новые возможности, такие как автозавершение адресов e-mail и телефонных номеров на основе списка контактов пользователя или адресной книги.

### Новые атрибуты

Кроме явно новых типов элементов ввода HTML5 определяет ряд новых атрибутов для элементов управления форм, которые помогают упростить некоторые общие задачи и дополнительно уточнить ожидаемые значения для определенных полей ввода.

#### placeholder

Обычный удобный прием в формах Web состоит в размещении замещаемого контента в текстовых полях ввода – например, чтобы предоставить дополнительную информацию об ожидаемом типе информации, ввод которой должен выполнить пользователь – который исчезает, когда элемент управления формы получает фокус. Хотя это обычно требует использования JavaScript (удаление контента поля формы в фокусе и восстановление текста по умолчанию, если пользователь покидает поле, не вводя ничего), мы можем теперь использовать просто атрибут placeholder:

<input type="text"… placeholder="John Doe">

Текстовое поле ввода с замещаемым текстом

Текстовое поле ввода с замещаемым текстом

#### autofocus

Другим обычным свойством, которое ранее требовало использования сценария, является автоматическое помещение поля формы в фокус при загрузке страницы. Теперь это можно сделать с помощью атрибута autofocus:

<input type="text" autofocus … >

Помните, что вы не должны иметь на странице более одного элемента управления формы с автофокусом. Необходимо также использовать функциональность такого вида с осторожностью в ситуациях, когда форма представляет основную область интереса на странице. Страница поиска является хорошим примером – при условии, что нет большого количества контента и пояснительного текста, имеет смысл задавать фокус автоматически на текстовом поле ввода формы поиска.

#### min и max

Как предполагают имена, эта пара атрибутов позволяет задать нижнюю и верхнюю границы для значений, которые могут вводиться в числовое поле формы, например, типы элементов ввода number, range, time или date (можно даже использовать их для задания верхней и нижней границы дат – например, в форме бронирования путешествия можно ограничить элемент выбора даты, позволяя выбирать только будущие даты). Для элементов ввода range, min и max необходимо, на самом деле, определять, какие значения возвращаются при отправке формы. Код вполне простой и не требует пояснений:

<input type="number" … min="1" max="10">

#### step

Атрибут step можно использовать с числовым значением ввода, чтобы определить точность возможных вводимых значений. Например, может потребоваться, чтобы пользователи вводили определенное время, но только с шагом 30 минут. В этом случае можно использовать атрибут step, помня о том, что для элементов ввода time значение атрибута задается в секундах:

<input type="time" … step="1800">

### Новые механизмы вывода

Кроме новых элементов управления форм, с которыми может взаимодействовать пользователь, HTML5 определяет ряд новых элементов, предназначенных специально для вывода и визуализации информации для пользователя.

#### <output>

Мы уже упоминали об элементе вывода output, когда говорили об элементе ввода range – этот элемент служит средством вывода "результата вычислений", или в более общем смысле для предоставления явно определенного вывода сценария (вместо простого выталкивания некоторого текста в случайный span или div с помощью innerHTML, например). Чтобы еще более явно указать, каким конкретно элементам управления формы соответствует output, можно – аналогично тому как для label – передать список ID в дополнительном атрибуте for элемента.

<input type="range" id="rangeexample" … >

<output onforminput="value=rangeexample.value" for="rangeexample"></output>

#### <progress> и <meter>

Два этих новых элемента очень похожи, оба в результате создают индикатор/панель, выводимую пользователю. Отличие их состоит в их назначении. Как предполагает название, progress предназначен для представления индикатора выполнения, чтобы указать процент завершения определенной задачи, в то время как meter является более общим индикатором скалярного измерения или дробного значения.

Панель индикатора выполнения

Панель индикатора выполнения

### Проверка

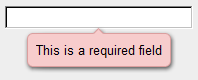
Проверка формы является очень важной, как на стороне клиента, так и на стороне сервера, чтобы помочь законным пользователям избежать и исправить ошибки, и чтобы остановить злоумышленных пользователей от отправки данных, которые могут нанести вред приложению. Так как браузеры могут теперь иметь представление о том, какой тип значений ожидается из различных элементов управления форм (по их собственному типу, или каким-либо верхним/нижним границам, заданным для числовых значений, дат и времени), они могут также предложить собственную проверку формы – еще одна утомительная задача, которая, до сих пор, требовала от авторов написания большого объема кода JavaScript или использования каких-либо готовых сценариев/библиотек проверки.

**Примечание**: Чтобы можно было проверять элементы управления формы, они должны иметь атрибут name, так как без него они в любом случае не могут быть отправлены как часть формы.

#### required

Одним из наиболее общих аспектов проверки формы является принудительное заполнение требуемых полей – недопущение отправки формы, пока определенные элементы информации не будут введены. Это можно теперь легко реализовать, добавляя атрибут required к элементам input, select или textarea.

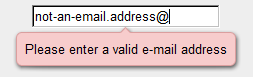
<input type="text" … required>



Клиентская проверка в браузере Opera в действии, показывающая ошибку для требуемого поля, которое было оставлено пустым

#### type и pattern

Как мы видели, авторы могут теперь определять тип записи, который они ожидают из полей формы – вместо простого определения текстового элемента ввода, авторы могут явно создавать элементы ввода для таких вещей как числа, адреса e-mail и URL. Как часть клиентской проверки, браузеры могут теперь проверять, что ввод пользователя в эти более специфические поля соответствует ожидаемой структуре – по существу, браузеры оценивают значения ввода в соответствии со встроенным шаблоном, который определяет, как выглядят допустимые значения в полях ввода этого типа, и будут предупреждать пользователя, когда его ввод не соответствует критериям.



Сообщение об ошибке браузера Opera для неверного адреса e-mail в поле ввода

Для других текстовых полей ввода, которые, тем не менее, должны следовать определенной структуре (например, формы регистрации, где имя пользователя может содержать только определенную последовательность букв нижнего регистра и цифр), авторы могут использовать атрибут pattern для определения своего собственного регулярного выражения.

<input type="text" … pattern="[a-z]{3}[0-9]{3}">

### Стилевое оформление с помощью CSS3

Соответственно встроенной проверке форм HTML5, Модуль Базового Интерфейса Пользователя CSS3 (<http://www.w3.org/TR/css3-ui/>) определяет последовательность новых псевдо-классов, которые могут использоваться для различения требуемых полей и для динамического изменения внешнего вида элементов управления формы на основе того, были они заполнены правильно или нет:

* :required и :optional позволяют явно оформить элементы управления на основе того, имеют они или нет требуемый атрибут
* :valid и :invalid будут применять стили к элементам управления формы в зависимости от клиентской проверки
* :in-range и :out-of-range работает специально с элементами управления, которые имеют атрибут min и/или max.

В нашем примере формы мы применяем специальное оформление проверки и диапазона на элементах ввода текста, e-mail, url и чисел, только когда они имеют в текущий момент фокус, поэтому пользователи будут получать некоторую немедленную визуальную реакцию во время ввода информации в эти поля, не влияющую на общий вид формы в целом.

input[type=text]:focus:valid,

input[type=e-mail]:focus:valid,

input[type=number]:focus:in-range { outline: 2px #0f0 solid; }

input[type=text]:focus:invalid,

input[type=e-mail]:focus:invalid,

input[type=number]:focus:out-of-range { outline: 2px #f00 solid; }



Стили :valid и :invalid, применяемые динамически во время ввода адреса e-mail

### Как обстоят дела для различных браузеров?

На настольном компьютере браузер Opera в настоящее время имеет наиболее полную реализацию новых типов элементов ввода и собственную клиентскую проверку, но уровень поддержки растет также во всех основных браузерах, поэтому пройдет не так много времени, и мы сможем использовать новые мощные средства в собственных проектах. Но как обстоит дело в старых браузерах?

Предполагается, что браузеры, которые не понимают новые типы элементов ввода (такие как date или number ) будут просто интерпретировать их как стандартные текстовые элементы ввода – не так как их более развитые в отношении HTML5 аналоги, но, по крайней мере, они позволяют заполнить формы. Однако, как и в отношении большей части новых функций HTML5, можно предложить лучшие резервные варианты для пользователей браузеров, которые недостаточно развиты для работы с формами HTML5, выполнять базовую проверку свойств на основе JavaScript и, только в случае необходимости, загружать внешние библиотеки JavaScript, чтобы "имитировать" поддержку более сложных виджетов элементов управления форм и проверку. Таким образом можно уже программировать будущее, не отвергая существующих пользователей.