GeekBrains

Профессиональная вёрстка

Урок 2

Структура HTML5. Знакомство с формами HTML5

Обзор HTML5, отличия от предыдущих версий. Новые семантические элементы HTML5. Новый способ структурирования страниц. Семантика текстового уровня.

Обзор HTML5

Структура HTML5

Документ, созданный с использованием HTML5, должен придерживаться определенной разметки. Сперва указывается тип документа с помощью специального кода описания типа документа, после чего дается название документа, а потом идет его содержимое.

Минимальная структура HTML5 документа представлена ниже.

Общая структура HTML5 документа:

```
<!DOCTYPE html>
<title>Первый документ на HTML5</title>
<p>Минимальная структура HTML5-документа</p>
```

Общая структура HTML5 документа:

```
<!DOCTYPE html>
<html lang="ru">
	<head>
		<meta charset="utf-8">
		<title></title>
		<link href="styles.css" rel="stylesheet">
		<script src="scripts.js"></script>
	</head>
	<body>
		<p>Первый документ с использованием HTML5 документа</p>
	</body>
</html>
```

Сравним данную структуру с предыдущей версией HTML-документа.

Общая структура HTML-документа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
	<title>Hello HTML</title>
	<meta content="text/html; charset=Windows-1251"
http-equiv="content-type"/>
</head>
<body>
...
</body>
</html>
```

Рассмотрим структуру HTML5 подробнее.

<!DOCTYPE html> указывается в первой строке каждого HTML5-документа. Это описание ясно указывает, что далее следует HTML5-содержимое.

<html> – традиционный элемент языка HTML. Его наличие не влияет на обработку браузером остального кода страницы. В HTML5 использование этого элемента оставлено на личное усмотрение разработчика. Атрибутом HTML5 является lang=”ru” – он определяет язык содержимого в создаваемом документе.

<head> и <body> – традиционные элементы разделов. Они облегчают восприятие документа, т.к. четко разделяют информацию о странице (заголовок страницы) и само содержимое (основная часть страницы).

Использование элементов <html>, <head> и <body> является просто вопросом стиля. Страница без этих элементов будет работать на любом браузере.

<meta charset="utf-8"> – тег, определяющий кодировку символов в документе. Новый тег для обозначения символов стал короче.

<title></title> используется для определения заголовка документа.

<link href="styles.css" rel="stylesheet"> указывает требуемую таблицу стилей. Добавление атрибута type="text/css", который был необходим ранее, не требуется.

<script src="scripts.js"></script> – вставка в веб-документ кода JavaScript по ссылке на внешний файл. Атрибут language="JavaScript" не является обязательным: если не указан какой-либо другой язык сценариев, браузеры автоматически предполагают, что используется JavaScript.

Новые семантические элементы HTML5

Мы рассмотрели изменения в синтаксисе HTML5. Более важными являются добавления, удаления и изменения поддерживаемых в HTML элементов.

Добавленные элементы:

Семантические элементы для работы со структурой страниц. Секционные элементы	<article>, <aside>, <figcaption>, <figure>, <footer>, <header>, <nav>, <section>, <details>, <summary>, <main>
Семантические элементы для работы с текстом	<mark>, <time>, <wbr> (поддерживался и ранее, но теперь официально является частью языка)
Элементы для работы с веб-формами и интерактивности	<input> (старый элемент, но со многими новыми подтипами), <datalist>, <keygen>, <meter>,<progress>, <menu>, <output>
Элементы для поддержки аудио, видео и подключаемых модулей	<audio>, <video>, <source>, <track>, <embed> (поддерживался и ранее, но теперь официально является частью языка)
Поддержка холста	<canvas>

Были удалены такие элементы оформления, как <big>, , <font>, <tt> и <strike>.

Новый способ структурирования страниц

Новые семантические элементы HTML5 позволяют улучшить структуру веб-страницы, добавляя смысловое значение заключенному в них содержимому.

Все семантические элементы имеют отличительную особенность: они по существу ничего не делают. Напротив, элемент <video>, например, вставляет в веб-страницу полноценный видеоплеер.

В этом случае возникает вопрос о целесообразности использования новых элементов.

Некоторые причины:

- Более удобное редактирование и сопровождение.
- Оптимизация поисковых движков. В настоящее время поисковые роботы уже проверяют на наличие некоторых семантических элементов HTML5, чтобы собрать всю возможную информацию об индексирующих их веб-страницах.
- Поддержка будущих возможностей. В новых браузерах и инструментах редактирования веб-страниц будет использоваться весь диапазон предоставляемых семантическими элементами возможностей.

Семантические элементы для работы со структурой страниц. Секционные элементы

Элемент <article>

Используется для группировки записей: публикаций, статей, записей блога, комментариев. Представляет собой независимый обособленный блок, предназначенный для многократного использования; как правило, начинается с заголовка. Может дублироваться на других страницах сайта и содержать внутри другие элементы <article>, которые по содержанию имеют близкое отношение к внешней статье. Если на странице присутствует только одна статья с заголовком и текстовым содержимым, она не нуждается в обертке элементом<article>.

```
<article>
      <header>
            <h2>...</h2>
      </header>
      <p>...</p>
      <footer>
      Опубликовано в категории<a href="/?cat=2" > Музыка</a>.
      <a href="/?p=17#respond">0 комментариев</a>
      </footer>"дз"
</article>
```

Элемент <section>

Представляет собой универсальный раздел документа. Группирует тематическое содержимое, не используется многократно и обычно содержит заголовок. Не является блоком-оберткой – для этих целей уместнее использовать элемент . В качестве содержимого может выступать оглавление, разделы научных публикаций, программа мероприятия. Домашняя страница сайта также может быть поделена на секции: блок вводной информации, новости и контакты.

```
<article>
      <h1>...</h1>
            <section>
                  <h2>...</h2>
                  <p>...</p>
            </section>
```

```
            <section>
                  <h2>...</h2>
                  <p>...</p>
            </section>
            <p>...</p>
</article>
```

<article> внутри <section>

Можно создавать родительские элементы <section> с вложенными элементами <article>, в которых есть один или несколько элементов <article>. Не все страницы должны быть устроены именно так, но это допустимый способ вложения элементов. Например, основная область контента страницы содержит два блока со статьями разной тематики. Можно сделать на этом акцент, поместив каждую статью одной тематики внутрь элемента <section>.

```
<section>
      <h1>Заметки о природе</h1>
            <article>
                  <h2>...</h2>
                  <p>...</p>
            </article>
            <article>
                  <h2>...</h2>
                  <p>...</p>
            </article>
</section>
<section>
      <h1>Исторические заметки</h1>
      <article>
            <h2>...</h2>
            <p>...</p>
      </article>
      <article>
            <h2>...</h2>
            <p>...</p>
      </article>
</section>
```

Элемент <aside>

Группирует содержимое, связанное с окружающим его контентом напрямую, но которое можно счесть отдельным (удаление этого блока не повлияет на понимание основного содержимого). Чаще всего элемент позиционируется как боковая колонка (как в книгах) и включает в себя группу элементов: <nav>, цифровые данные, цитаты, рекламные блоки, архивные записи. Не подходит для блоков, просто позиционированных в стороне.

```
<aside>
      <h2>...</h2>
      <p>...</p>
</aside>
```

```
<aside>
      <h2>...</h2>
      <p>...</p>
      <blockquote>
            <p>...<cite>...</cite>...</p>
            <p>...</p>
      </blockquote>
</aside>
```

На рис. 1 показан способ применения элемента <aside>.

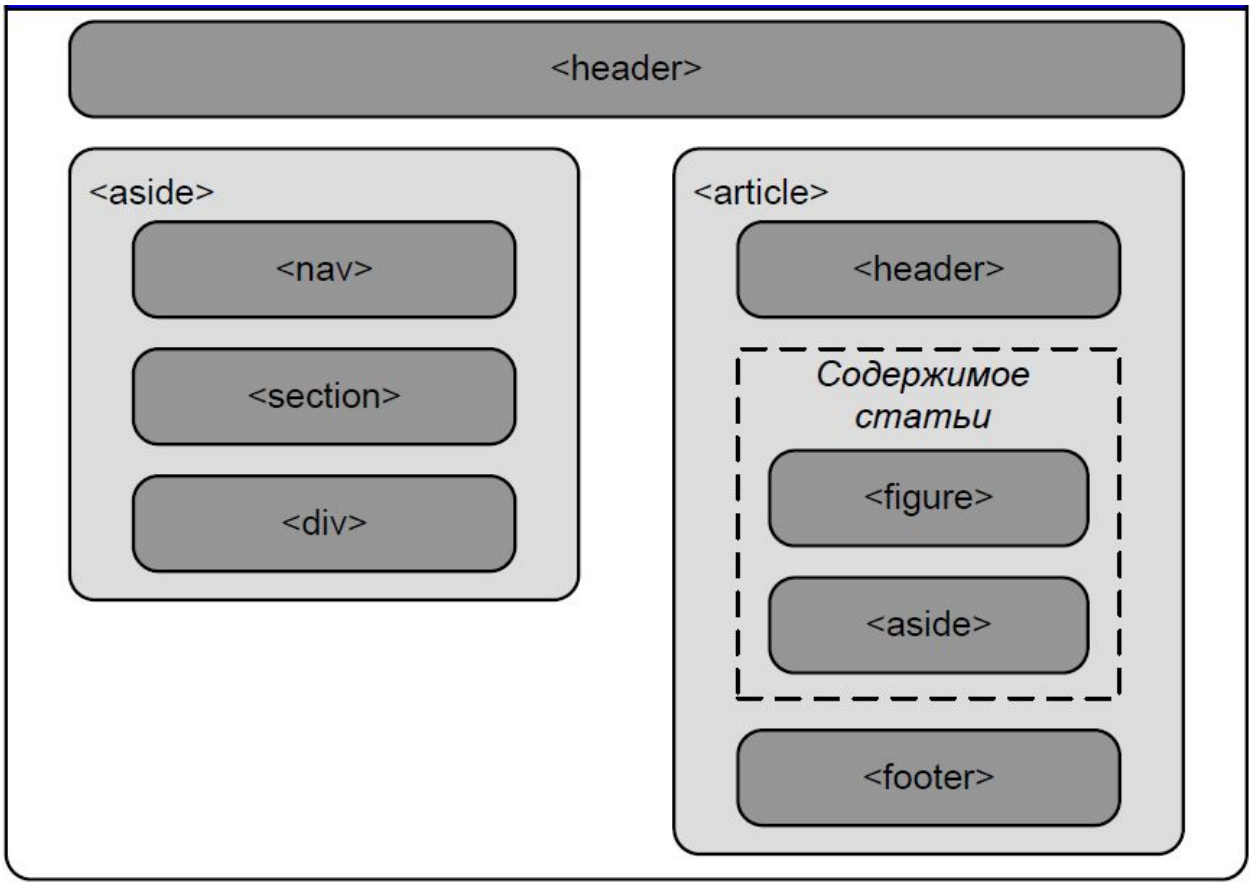

Рис. 1. Использование элемента <aside>

Элемент <footer>

Представляет собой нижний колонтитул содержащей его секции или корневого элемента. Обычно содержит информацию об авторе статьи, данные о копирайте и т.д. Если используется как колонтитул всей страницы, содержимое дополняется сведениями об авторских правах, ссылками на условия использования, контактную информацию, ссылками на связанное содержимое и т.п.

В одном веб-документе может быть несколько элементов <footer>. Каждая страница и каждая статья может иметь свой элемент <footer>. Более того,<footer> можно поместить в элемент <blockquote>, чтобы указать источник цитирования.

```
<footer>
      <address>...</address>
```

```
      <small>@2014...</small>
</footer>
```

Элемент <address>

Используется для определения контактной информации автора/владельца документа или статьи. Для обозначения автора документа тег размещают внутри элемента <body>, для отображения автора статьи – внутри тега <article>. В браузере обычно отображается курсивом.

Элемент <figure>

Многие веб-страницы оформляются изображениями. Но концепт *рисунка* несколько иной, чем изображения. Спецификация HTML5 советует рассматривать рисунки подобно иллюстрациям в книге: изображения, отдельные от текста, но на которые в тексте делаются ссылки. Рисунки размещаются как плавающие объекты, т.е. их вставляют в первое удобное место в тексте, вместо того чтобы закреплять возле конкретного слова или элемента. Часто рисунки снабжены подписями, которые плавают вместе с ними.

Для контейнера рисунка можно использовать элемент <figure>. Текст подписи к рисунку помещается в элемент <figcaption> внутри элемента <figure>:

```
<figure class="FloatFigure">
      <img src="image.jpg" alt="Рисунок в тексте">
      <figcaption>Данный текст отображается под рисунком</figcaption>
</figure>
```

Элемент <header>

Группирует вводные и навигационные элементы, не является обязательным. Может содержать заголовки, оборачивать содержание раздела страницы, форму поиска или логотипы. В HTML-документе может содержаться одновременно несколько элементов <header>, и они могут располагаться в любой части страницы.

```
<header>
      <h1>...</h1>
      <h2>...</h2>
</header>
```

Элемент <header> нельзя помещать внутрь элементов <footer>, <address> или другого <header>.

Элемент <nav>

Предназначен для создания блока навигации веб-страницы или всего веб-сайта, при этом не обязательно должен находиться внутри <header>. На странице может быть несколько элементов <nav>. Не заменяет теги <ul> или <ol>, а просто их обрамляет. Не все группы ссылок на странице должны быть обернуты <nav> – этот элемент предназначен в первую очередь для разделов, которые состоят из главных навигационных блоков.

```
<nav>
      <ul>
            <li><a>...</a></li>
```

```
            <li><a>...</a></li>
            <li><a>...</a></li>
      </ul>
</nav>
```

В качестве элементов панели навигации можно использовать не только элементы списков:

```
<nav>
      <p><a href="">...</a></p>
      <p><a href="">...</a></p>
</nav>
```

Также можно добавлять заголовки внутрь элемента:

```
<nav>
       <h2>...</h2>
            <ul>
                  <li><a>...</a></li>
                  <li><a>...</a></li>
                  <li><a>...</a></li>
            </ul>
</nav>
```

Элемент <main>

Используется для основного содержимого документа. Оно должно быть уникальным и не включать типовые блоки, такие как шапку, подвал, навигацию, боковые панели, формы поиска и т.д.

Семантика для текстового содержимого

Элемент <time>

Определяет время (24 часа) или дату по григорианскому календарю с возможным указанием времени и смещения часового пояса. Текст, заключенный в данный тег, не имеет стилевого оформления браузером. Для тега доступен атрибут datetime, в качестве содержимого указывается то, что будет видеть пользователь на экране своего компьютера:

```
<time datetime="2014-09-25"> 25 Сентября 2014</time>
```

Чтобы дата могла считываться автоматически, она должна быть в формате YYYY-MM-DD. Время, которое также может указываться, задается в формате HH:MM с добавлением разделяющего префикса T (time):

```
<time datetime="2014-09-25T12:00"> 25 Сентября 2014</time>
```

Элемент <mark>

Текст, помещенный внутрь тега <mark>, по умолчанию выделяется желтым цветом (цвет фона и шрифта в выделенном блоке можно изменить, задав определенные CSS-стили). С помощью данного тега можно отмечать важное содержимое, а также ключевые слова.

Элемент <bdi>

Отделяет фрагмент текста, который должен быть изолирован от остального текста, для двунаправленного форматирования текста. Используется для текстов, написанных одновременно на языках, читающихся слева направо и справа налево.

Элемент <wbr>

Одиночный тег, который указывает браузеру место, где можно добавить разрыв длинной строки в случае необходимости.

Итоги

Недостатком элемента является то, что он не предоставляет никакой информации о странице. Встретив в разметке элемент , вы (или браузер, средство разработки, поисковый робот и т.п.) понимаете, что нашли отдельный блок страницы, но не знаете назначение этого блока.

Чтобы исправить такую ситуацию, в HTML5 некоторые элементы можно заменить более описательными семантическими элементами. Они действуют точно так же, как и -элемент: группируют часть разметки в блок, но не выполняют никаких собственных операций над содержимым блока. Они также предоставляют «стилевую зацепку», позволяющую присоединять форматирование. Но кроме всего этого, они добавляют в страницу семантический смысл.

Работа с формами

HTML-формы – простые элементы управления HTML, которые применяются для сбора информации от посетителей веб-сайта. К ним относятся текстовые поля для ввода данных с клавиатуры, списки для выбора предопределенных данных, флажки для установки параметров и т.п.

HTML-формы существовали с самых ранних времен языка HTML и с тех пор практически не изменились.

Стандарт HTML5 совершенствует уже существующую модель HTML-форм. Это означает, что HTML5-формы могут работать и на старых браузерах, только без нововведений. HTML5-формы также позволяют применять новые возможности, которые уже используются разработчиками в настоящее время. Эти возможности более доступны, не требуют написания страниц сценариев JavaScript или применения инструментариев JavaScript сторонних разработчиков.

HTML-форма. Базовые понятия

Веб-форма – это набор текстовых полей, списков, кнопок и других активизируемых щелчком мыши элементов управления, посредством которых посетитель страницы может предоставить ей тот или иной вид информации.

Все основные веб-формы работают одинаково. Пользователь вводит определенную информацию, а потом нажимает кнопку, чтобы отправить введенную информацию на веб-сервер. По прибытии на веб-сервер она обрабатывается каким-либо приложением, которое потом предпринимает соответствующий очередной шаг.

Модернизация стандартной HTML-формы

Элемент <form> удерживает вместе все элементы управления формы, которые также называются полями. Он также указывает браузеру, куда отправить данные после нажатия пользователем кнопки отправки, предоставляя URL в атрибуте action. Но если вся работа будет выполняться на стороне клиента сценариями JavaScript, то для атрибута action можно просто указать значение #.

Форма разделяется на логические фрагменты с помощью элемента <fieldset>. Каждому разделу можно присвоить название, для чего используется элемент <legend>. В листинге 1 приводится разметка формы.

Листинг 1 – HTML-форма

```
<form action="#">
  <p><i>Пожалуйста, заполните форму регистрации пользователей. Обязательные
поля помечены </i><em>*</em></p>
  <fieldset>
    <legend>Контактная информация</legend>
      <label for="name">Фамилия <em>*</em></label>
      <input id="name"><br>
      <label for="name">Имя <em>*</em></label>
      <input id="name"><br>
      <label for="name">Отчество</label>
      <input id="name"><br>
      <label for="telephone">Телефон <em>*</em></label>
      <input id="telephone"><br>
      <label for="email">Email <em>*</em></label>
      <input id="email"><br>
  </fieldset>
  <fieldset>
    <legend>Персональная информация</legend>
      <label for="age">Возраст<em>*</em></label>
      <input id="age"><br>
      <label for="gender">Пол</label>
      <select id="gender">
        <option value="female">Мужской</option>
        <option value="male">Женский</option>
      </select><br>
      <label for="comments">Перечислите дополнительную информацию, которую вы
хотели бы сообщить о себе</label>
      <textarea id="comments"></textarea>
  </fieldset>
  <fieldset>
    <legend>Выберите ваши увлечения</legend>
    <label for="sport"><input id="sport" type="checkbox"> Спорт</label>
    <label for="tourist"><input id="tourist" type="checkbox"> Туризм</label>
    <label for="padi"><input id="padi" type="checkbox"> Дайвинг</label>
    <label for="travels"><input id="travels" type="checkbox">
Путешествия</label>
    <label for="auto"><input id="auto" type="checkbox"> Автомобили</label>
    <label for="it"><input id="it" type="checkbox"> IT</label>
  </fieldset>
  <p><input type="submit" value="Отправить информацию"></p>
</form>
```

Файл forms.css находится в приложении к занятию.

Большая часть работы в примере выполняется универсальным элементом **<input>**, который собирает данные и создает флажки, переключатели и списки. Для ввода одной строки текста применяется элемент <input>, а нескольких – элемент **<textarea>**; элемент **<select>** создает выпадающий список. Краткий обзор этих и других элементов управления форм приведен в табл. 1.

Таблица 1

Элементы управления формы

Элемент управления	**HTML-элемент**	**Описание**
Однострочное текстовое поле	<input type="text"> <input type="password">	Выводит однострочное текстовое поле для ввода текста. Если для атрибута type указано значение password, вводимые пользователем символы отображаются в виде звездочек (*) или маркеров-точек (•)
Многострочное текстовое поле	<textarea>...</textarea>	Текстовое поле, в которое можно ввести несколько строчек текста
Флажок	<input type="checkbox">	Выводит поле флажка, который можно установить или очистить
Переключатель	<input type="radio">	Выводит переключатель – элемент управления в виде небольшого кружка, который можно включить или выключить. Обычно создается группа переключателей с одинаковым значением атрибута name, вследствие чего можно выбрать только один из них
Кнопка	<input type="submit"> <input type="image"> <input type="reset"> <input type="button">	Выводит стандартную кнопку, активизируемую щелчком мыши. Кнопка типа submit всегда собирает информацию с формы и отправляет ее для обработки. Кнопка типа image делает то же самое, но позволяет вместо текста на кнопке вставить изображение. Кнопка типа reset очищает поля формы от введенных пользователем данных. А кнопка типа button сама по себе не делает ничего. Чтобы ее нажатие выполняло какое-либо действие, для нее нужно добавить сценарий JavaScript
Список	<select>...</select>	Выводит список, из которого пользователь может выбирать значения. Для каждого значения списка добавляем элемент <option>

Теперь, когда есть форма, с которой можно работать, настало время улучшить ее с помощью HTML5.

Добавление подсказок

Обычно поля новой формы не содержат никаких данных. Для некоторых пользователей такая форма может быть не совсем понятной, в частности, какую именно информацию нужно вводить в конкретное поле. Поэтому часто поля формы содержат пример данных, которые нужно ввести в них. Этот подстановочный текст также называется "водяным знаком", так он часто отображается шрифтом светло-серого цвета, чтобы отличить его от настоящего, введенного содержимого.

Подстановочный текст для поля создается с помощью атрибута placeholder:

```
  <fieldset>
    <legend>Контактная информация</legend>
      <label for="name">Фамилия <em>*</em></label>
      <input id="name" placeholder="Иванов"><br>
      <label for="name">Имя <em>*</em></label>
      <input id="name" placeholder="Иван"><br>
      <label for="name">Отчество</label>
      <input id="name" placeholder="Сидорович"><br>
  </fieldset>
```

Фокусировка на элементе

Вводить информацию в форму пользователи не смогут до тех пор, пока не щелкнут мышью по необходимому полю или выделят его с помощью клавиши <Tab>, установив таким образом *фокус* на этом поле.

Возможно установить фокус на нужном начальном поле автоматически. На этой идее основан новый HTML5-атрибут autofocus, который можно вставить в элемент <input>:

```
    <label for="name">Фамилия <em>*</em></label>
      <input id="name" placeholder="Иванов" autofocus><br>
```

Проверка ошибок

Поля в форме предназначены для сбора информации от посетителей страницы.

Серьезной веб-странице требуется проверка данных, т.е. какой-либо способ обнаружения ошибок во введенных данных, а еще лучше – не допускающий ошибок вообще. На протяжении многих лет веб-разработчики делали это с помощью процедур JavaScript собственной разработки или профессиональных библиотек JavaScript.

Создатели HTML5 разработали систему проверки на *стороне клиента.*

Допустим, определенное поле нельзя оставлять пустым и посетитель должен ввести в него хоть что-то. В HTML5 это осуществляется с помощью атрибута required в соответствующем поле.

Когда пользователь нажмет кнопку для отправки формы, браузер начнет проверять поля сверху вниз. Встретив первое некорректное значение, он прекращает дальнейшую проверку, отменяет отправку формы и выводит сообщение об ошибке рядом с полем, вызвавшим эту ошибку. (Кроме этого, если

при заполнении формы область с полем ошибки вышла за пределы экрана, браузер прокручивает экран, чтобы это поле находилось вверху страницы.) После того как пользователь исправит данную ошибку и опять нажмет кнопку для отправки формы, браузер остановится на следующей ошибке ввода.

Отключение проверки

В некоторых случаях может потребоваться отключить проверку: например, вы хотите протестировать свой серверный код на правильность обработки поступивших некорректных данных. Чтобы отключить проверку для всей формы, в элемент <form> добавляется атрибут novalidate.

Оформление результатов проверки

Веб-разработчики не могут оформлять сообщения об ошибках проверки, но могут изменять внешний вид полей в зависимости от *результатов* их валидации. Например, можно выделить поле с неправильным значением цветным фоном сразу же, когда браузер обнаружит неправильные данные. Для этого требуется добавить несколько простых CSS3-псевдоклассов. Доступны следующие опции:

- required и optional. Применяют форматирование к полю в зависимости от того, использует оно атрибут required или нет.
- valid и invalid. Применяют форматирование к полю в зависимости от правильности введенного в него значения. Не забывайте, что большинство браузеров не проверяет данные, пока пользователь не попытается отправить форму, поэтому форматирование полей с некорректными значениями не выполняется сразу же при введении такого значения.
- in-range и out-of-range. Форматирование к полям, для которых используется атрибут min или max, чтобы ограничить их значение определенным диапазоном значений.

```
form input:required
{
      background-color: lightyellow;
}
```

Проверка с помощью регулярных выражений

Самым мощным (и самым сложным) поддерживаемым HTML5 типом проверки является проверка на основе регулярных выражений.

Регулярное выражение – это шаблон для сопоставления с образцом, закодированный согласно определенным синтаксическим правилам. Регулярные выражения применяются для поиска в тексте строк, которые отвечают шаблону. Например, с помощью регулярного выражения можно проверить, что почтовый индекс содержит правильное число цифр или в адресе электронной почты присутствует знак @, а его доменное расширение содержит по крайней мере два символа. Возьмем, например, следующее выражение:

[A-Z]{3}-[0-9]{3}

Квадратные скобки в начале строки определяют диапазон допустимых символов. Иными словами, группа [A-Z] разрешает любые прописные буквы от A до Z. Следующая за ней часть в фигурных скобках указывает множитель, т.е. {3} означает, что нужны три прописные буквы. Следующее тире не имеет никакого специального значения и означает само себя, т.е. указывает, что после трех

прописных букв должно быть тире. Наконец, [0-9] означает цифры в диапазоне от 0 до 9, а {3} требует три таких цифры.

Регулярные выражения полезны для поиска в тексте строк, отвечающих условиям, заданным в выражении, и проверки, что определенная строка отвечает заданному регулярным выражением шаблону. В формах HTML5 регулярные выражения применяются для валидации.

Для обозначения начала и конца значения в поле символы ^ и $ не требуются – HTML5 автоматически предполагает их наличие. Это означает, что значение в поле должно полностью совпасть с регулярным выражением, чтобы его можно было считать корректным.

Таким образом, следующие значения будут допустимыми для этого регулярного выражения:

- QDR-001
- WES-205
- LOG-104

А эти нет:

- qdr-001
- TTT-0259
- 5SD-000

Но регулярные выражения очень быстро становятся более сложными, чем рассмотренный нами пример, и создание правильного регулярного выражения может быть довольно трудоемкой задачей. Поэтому большинство разработчиков предпочитает использовать для проверки данных на своих страницах готовые регулярные выражения.

Чтобы применить полученное тем или иным путем регулярное выражение для проверки значения поля <input> или <textarea>, его следует добавить в этот элемент в качестве значения атрибута pattern:

Новые типы элементов

В HTML5 в элемент <input> было добавлено несколько новых типов, и если какой-либо браузер не поддерживает их, он будет обрабатывать их как обычные текстовые поля.

В табл. 2 приведены основные типы.

Таблица 2

Тип данных	Назначение
image	Создает кнопку, позволяя вместо текста на кнопке вставить изображение
email	Используется для полей, предназначенных для ввода адресов электронной почты
URL	Применяется для полей ввода URL-адресов
tel	Применяется с целью обозначения полей для ввода телефонных номеров, которые могут быть представлены в разных форматах. HTML5 не требует от браузеров выполнения проверки телефонных номеров
number	При вводе данных в поле типа number браузер автоматически игнорирует все символы, кроме цифр. Можно использовать атрибуты min, max и step. Атрибут step указывает шаг изменения числа (в большую или меньшую сторону). Например, установив значение step в 0.1, можно вводить такие значения, как 0,1, 0,2 0,3 и т.д. По умолчанию значение шага равно 1
range	Подобно типу number, этот тип может представлять целые и дробные значения. Также поддерживает атрибуты min и max для установки диапазона значений. Разница состоит в том, каким образом поле типа range представляет свою информацию. Вместо счетчика, как для поля типа number, интеллектуальные браузеры отображают ползунок
date	Дата по шаблону ГГГГ-ММ-ДД
time	Время в 24-часовом формате с необязательными секундами, по шаблону чч:мм:сс.сс
datetime-local	Дата и время, разделенные прописной английской буквой T (по шаблону ГГГГ-ММ-ДД*T*чч:мм:сс)
datetime	Дата и время, как и для типа datetimelocal, но со смещением временного пояса. Используется такой же шаблон (ГГГГ-ММ-ДД*T*чч:мм:сс-чч: мм), как и для элемента <time>
month	Год и номер месяца по шаблону ГГГГ-ММ
week	Год и номер недели по шаблону ГГГГ-*И*нн. Обратите внимание, что в зависимости от года может быть 52 или 53 недели
color	Применяется для полей, предназначенных для ввода цвета. Тип данных color – интересная, хотя и редко используемая второстепенная возможность, позволяющая посетителю веб-страницы выбирать цвет из выпадающей палитры, похожей на палитру графического редактора MS Paint

В стандарте также вводится несколько совершенно новых элементов. С их помощью в форму можно добавить список предложений, индикатор выполнения задания, панель инструментов

Подсказки ввода <datalist>

Элемент <datalist> предоставляет способ присоединить выпадающий список возможных вариантов ввода к обыкновенному текстовому полю. Заполняющим форму пользователям он дает возможность выбрать вариант ввода из списка значений либо ввести требуемое значение вручную.

Чтобы использовать элемент datalist, сначала нужно создать обычное текстовое поле. Допустим, мы создали обычный элемент <input>:

```
<fieldset>
     <legend>Выберите любимый фрукт</legend>
     <label for="fruit">Фрукт</label>
     <input id="fruit" list="dl_fruits">
</fieldset>
```

Чтобы добавить к этому полю выпадающий список возможных значений, для него нужно создать элемент <datalist>. Технически этот элемент можно разместить в любом месте разметки, т.к. он не отображается браузером, а просто предоставляет данные для использования в текстовом поле ввода. Но логично поместить этот элемент либо непосредственно перед тем элементом <input>, для которого он предоставляет свои данные, либо сразу же после него. Далее показан пример кода для создания списка <datalist>.

Как и традиционное поле <select>, список <datalist> использует элементы <option>. Каждый элемент <option> представляет собой отдельное возможное значение, которое браузер может предложить заполняющему форму. Значение атрибута label содержит текст, который отображается в текстовом поле, а атрибут value – текст, который будет отправлен на сервер, если пользователь выберет данную опцию.

Сам по себе список <datalist> не отображается в браузере. Чтобы подключить его к текстовому полю, нужно установить значение атрибута list равным значению параметра id соответствующего списка <datalist>:

```
<datalist id="dl_fruits">
        <option label="Яблоко" value="apple">
        <option label="Ананас" value="ananas">
        <option label="Абрикос" value="apricot">
        <option label="Арбуз" value="watermelon">
        <option label="Авокадо" value="avocado">
        <option label="Апельсин" value="Orange">
</datalist>
```

В браузерах, которые поддерживают <datalist>, посетители увидят результат. Другие браузеры будут игнорировать атрибут list и разметку <datalist>, делая все предложения возможного ввода бесполезными.

Индикатор выполнения <progress> и счетчик <meter>

Новые графические элементы <progress> и <meter> внешне похожи друг на друга, но имеют разные назначения.

Элемент <progress> отображает ход выполнения задания посредством зеленой пульсирующей полоски на сером фоне.

Элемент <meter> указывает значение в диапазоне известных значений. Внешне он похож на элемент <progress>, но зеленая полоска имеет другой оттенок и не пульсирует.

Элемент <progress> использует атрибут value, который обозначает ход выполнения задания в виде дробной величины от 0 до 1. Графически это отображается соответствующей шириной полоски индикатора. Например, чтобы показать, что задание выполнено на 25%, атрибуту value присваивается значение 0,25:

```
<progress value="0.25"></progress>
```

Альтернативно можно использовать атрибут max, чтобы установить максимальное значение и изменить масштаб индикатора. Например, при значении max, равном 200, значение value должно быть между 0 и 200. Если сделать значение value равным 50, получим те же самые 25% заполнения индикатора, как и в предыдущем примере:

```
<progress value="50" max="200"></progress>
```

Элемент <meter> имеет подобную модель, но отображает любой вид измерений. Иногда его еще называют шкалой. Часто значение атрибута value этого элемента отображает какую-то действительную величину, например, денежную сумму на счету, количество дней, вес в килограммах и т.п. Отображение этой информации управляется установкой значений атрибутов min и max:

```
Пройденная дистанция: <meter min="5" max="70" value="28"> 28 метров</meter>
```

Практическое задание

1. Приступить к верстке второй страницы нашего интернет-магазина.
2. Добавить необходимые теги HTML5, которые изучили на уроке.
3. Добавить необходимую проверку для полей ввода, используя полученные знания.
4. * Так как у всех учеников разное количество времени, можно браться за следующие страницы интернет-магазина.

Дополнительные материалы

<main> – https://webref.ru/html/main.
<article> – https://webref.ru/html/article.
<figure> – https://webref.ru/html/figure.
<figcaption> – https://webref.ru/html/figcaption.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. http://www.wisdomweb.ru/.
2. http://html5book.ru/.
3. Гоше Х. HTML5. Для профессионалов. – СПб.: Питер, 2013. – 496 с.: ил.
4. Хоган Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения. – СПб.: Питер, 2012. – 272 с.
5. Макфарланд Д. Большая книга CSS3. – СПб.: Питер, 2016. – 608 с.