Новые семантические элементы HTML5

До этого момента 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>, <hgroup>, <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>, <command>, <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>

Удалены были элементы оформления, такие как <big>, , <font>, <tt> и <strike>.

Новый способ структурирования страниц

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

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

В данном случае возникает вопрос о целесообразности использования новых элементов.

Рассмотрим некоторые причины:

1. Более удобное редактирование и сопровождение;
2. 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.
3. Поддержка будущих возможностей. В новых браузерах и инструментах редактирования веб-страниц будет использоваться весь диапазон предоставляемых семантическими элементами возможностей.

Рассмотрим назначение новых элементов более подробно:

Семантические элементы для работы со структурой страниц. Секционные элементы

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 элемента <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>, он 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 <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>. Как каждая страница, так и каждая статья может иметь свой элемент <footer>, более того,<footer> можно поместить в элемент <blockquote>, чтобы указать источник цитирования.

```
<footer>
    Все права защищены
</footer>
```

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 содержанию внешней статьи. Если на странице присутствует только одна статья с заголовком и текстовым содержимым, она не нуждается в обёртке элементом<article>.

```
<article>
        <header>
            <h2>...</h2>
        </header>
        <p>...</p>
        <footer>
            Опубликовано дата и время
        </footer>
</article>
```

Элемент <section>

Элемент представляет собой универсальный раздел документа. Группирует тематическое содержимое, не используется многократно и обычно содержит заголовок. Не является блоком-обёрткой, 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.

```
<h1>Сайт портфолио</h1>
<section>
    <h2>Обо мне</h2>
    <p>...</p>
</section>
<section>
    <h2>Мои навыки</h2>
    <p>...</p>
</section>
<p>...</p>
```

<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 две блока со статьями разной тематики. Можно сделать на этом акцент, поместив каждый статьи одной тематики внутрь элемента <section>

```
<section>
        <h2>Заметки о природе</h2>
                <article>
                        <h3>...</h3>
                        <p>...</p>
                </article>
                <article>
                        <h3>...</h3>
                        <p>...</p>
                </article>
</section>
<section>
        <h2>Исторические заметки</h2>
        <article>
                <h3>...</h3>
                <p>...</p>
```

```
        </article>
        <article>
                <h3>...</h3>
                <p>...</p>
        </article>
</section>
```

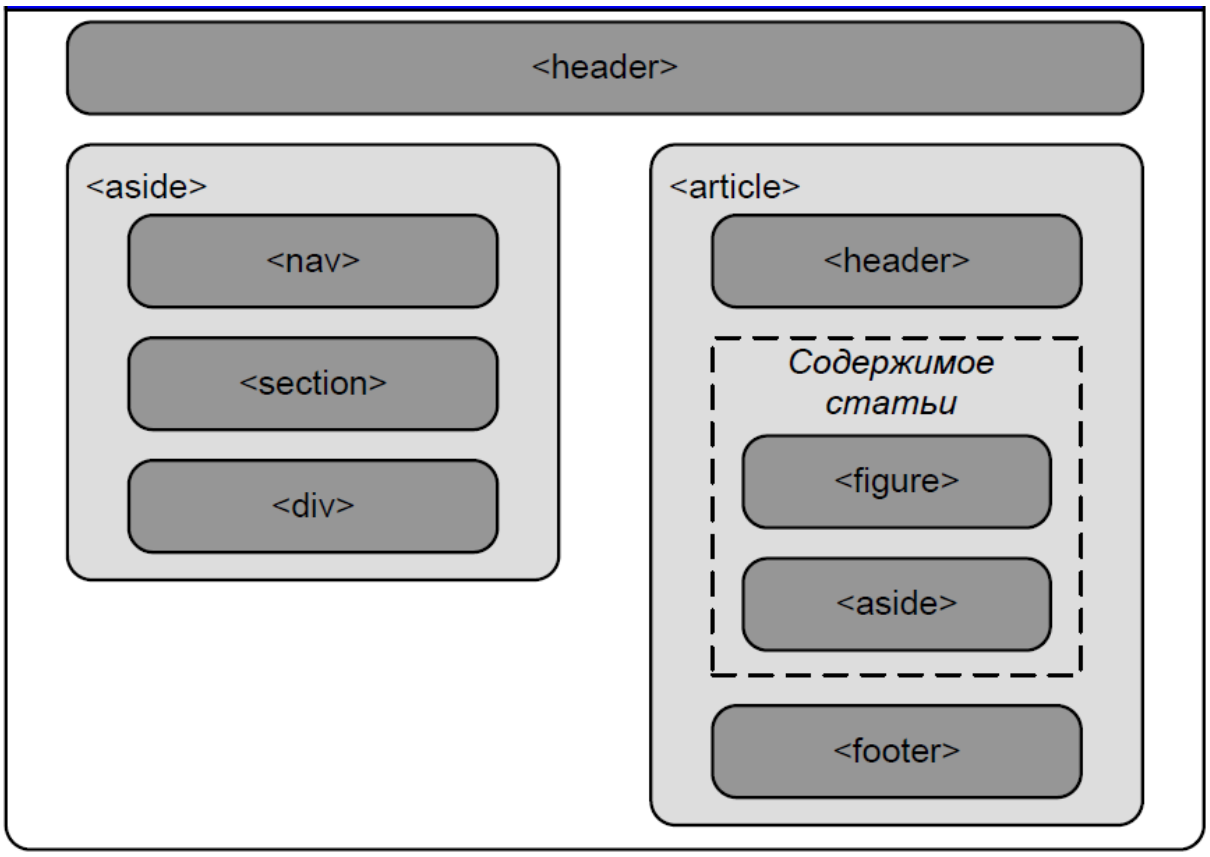
Элемент <aside>

Группирует содержимое, связанное с окружающим его контентом напрямую, но которое можно счесть отдельным (т.е., 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>...</p>
                <p>...</p>
        </blockquote>
</aside>
```

На рисунке показан способ применения элемента <aside>



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 во многом подобно рисункам в книге, иными словами, изображение, отдельное от текста, но на которое в тексте делаются ссылки. В общем, 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.

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

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

Элемент <main>

Используется для основного содержимого документа. Содержимое 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, заключённый в данный тег, не имеет стилевого оформления браузером. Для тега доступен атрибут datetime, в качестве содержимого которого указывается то, что будет видеть пользователь на экране своего компьютера:

```
<time datetime="2030-09-25"> 25 Сентября 2030</time>
```

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

```
<time datetime="2030-09-25T12:00"> 25 Сентября 2030</time>
```

Элемент <mark>

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

Элемент <wbr>

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

Итоги работы с семантикой

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. Эти семантические элементы действуют в точности таким же образом, как и -элемент: они 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 также добавляют в страницу семантический смысл.

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 определённую информацию, а потом нажимает кнопку, чтобы отправить введенную информацию на веб-сервер. По прибытию на веб-сервер эта информация 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. Кроме этого, 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 сценариям JavaScript, то для атрибута action можно просто указать значение #.

```
<form action="#">
  <h2>Пожалуйста, заполните форму регистрации пользователей. Обязательные поля помечены *</h2>
    <h3>Контактная информация</h3>
    <label for="name2">Фамилия <em>*</em></label>
    <input id="name2"><br>
    <label for="name">Имя <em>*</em></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>
    <h3>Персональная информация</h3>
    <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>
     <div><label for="comments">Перечислите дополнительную информацию, которую вы хотели бы сообщить о себе</label></div>
    <textarea id="comments"></textarea>
   <h3>Выберите ваши увлечения</h3>
   <label for="sport"><input id="sport" type="checkbox">Спорт</label>
   <label for="turist"><input id="turist" 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>
   <div><button type="submit" >Отправить информацию</button></div>
</form>
```

Большая часть работы в нашем примере выполняете универсальным элементом **<input>**, который собирает данные и создает флажки, переключатели и списки. Для ввода одной строки текста применяется элемент <input>, а для нескольких — элемент **<textarea>**; элемент <select> создает выпадающий список. Краткое обозрение этих и других элементов управления форм приведено в таблице:

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 её 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

```
<h2>Контактная информация</h2>
<label for="name2">Фамилия <em>*</em></label>
```

```
    <input id="name2" placeholder="Иванов"><br>
    <label for="name">Имя <em>*</em></label>
    <input id="name" placeholder="Иван"><br>
```

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="name2">Фамилия <em>*</em></label>
    <input id="name2" 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 ещё лучше — способ, не допускающий ошибок вообще. На протяжении многих лет веб-разработчики делали это с помощью процедур JavaScript собственной разработки или посредством профессиональных библиотек JavaScript.

Создатели HTML5 разработали систему проверки на *стороне клиента*

Допустим, что 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

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 таблице 1 приведены основные типы:

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 самых 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, который 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-*W*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 — это интересная, хотя 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 даёт возможность либо 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>:

```
<h2>Выберите любимый фрукт</h2>
<label for="fruit">Фрукт</label>
<input id="fruit" list="dl_fruits">
```

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. Для того 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>
```

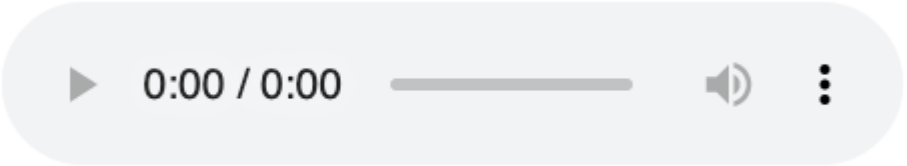
HTML5 Аудио и Видео

Аудио HTML5

В следующем коде приведён пример использования элемента <audio>:

```
<h2>Ознакомительный фрагмент аудио. :</h2>
<audio src="audio.mp3" controls></audio>
```

Атрибут src содержит имя аудиофайла для воспроизведения, а атрибут controls указывает браузеру, что нужно отобразить базовые элементы управления воспроизведением. Своим внешним видом эти элементы управления слегка отличаются от браузера к браузеру, но все они имеют одинаковое назначение: разрешают пользователю начинать и останавливать воспроизведение, переходить в другое место записи и регулировать громкость



Кроме атрибута controls элемент <audio> поддерживает ещё три атрибута: preload, auotoplay и loop. Атрибут preload указывает браузеру способ загрузки аудиофайла. Значение auto этого атрибута указывает браузеру загружать аудио-файл полностью, чтобы он был доступен, когда пользователь нажмет кнопку воспроизведения.

Загрузка осуществляется в фоновом режиме, чтобы посетитель веб-страницы мог перемещаться по странице и просматривать её, не дожидаясь завершения загрузки аудиофайла.

Атрибут preload может принимать два значения. Значение metadata указывает браузеру загрузить первую небольшую часть файла, достаточную, чтобы определить некоторые его основные характеристики (например, общий размер файла). Атрибут none указывает браузеру не загружать аудио-файл автоматически. Эти опции можно использовать для того, чтобы сэкономить пропускную способность подключения.

```
<audio src="rubberduckies.mp3" controls preload="metadata"></audio>
```

Когда атрибуту preload задано значение none или metadata, загрузка аудиофайла начинается после того, как пользователь нажмет кнопку воспроизведения.

Если значение атрибута preload не установлено, то браузеры действуют по своему индивидуальному усмотрению. Большинство браузеров предполагает auto в качестве значения по умолчанию. Кроме этого, важно отметить, что атрибут preload не является обязательным для выполнения правилом, а рекомендацией браузеру желаемого действия, которую он может игнорировать в зависимости от других обстоятельств.

Атрибут autoplay указывает браузеру начать воспроизведение сразу же после завершения загрузки страницы:

```
<audio src="audio.mp3" controls autoplay></audio>
```

Если этот атрибут не используется, пользователь должен нажать кнопку запуска, чтобы начать воспроизведение. Элемент <audio> можно использовать для того, чтобы проигрывать фоновую музыку.

Атрибут loop указывает браузеру повторять воспроизведение:

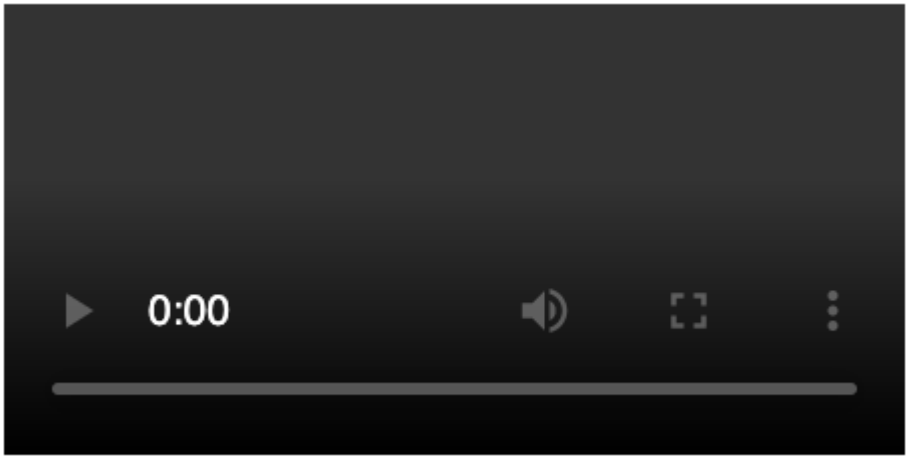
```
<audio src="audio.mp3" controls loop></audio>
```

Воспроизведения видео с помощью элемента <video>

С элементом <audio> хорошо идёт в паре элемент <video>. Он применяет такие же атрибуты src, controls, autoplay и loop. Пример использования этого элемента показан в следующем коде:

```
<h2>Демонстрационное видео!</h2>
<video src="video.mp4" controls></video>
```

Как и в случае с элементом <audio>, атрибут controls указывает браузеру создать набор элементов управления воспроизведением (рисунок 2). В большинстве браузеров эти элементы скрываются при щелчке где-нибудь в области страницы и отображаются опять при наведении курсора мыши на область видеоплеера.



Кроме общих с элементом <audio> атрибутов, элемент <video> имеет три своих собственных атрибута: height, width и poster.

Атрибуты height и width устанавливают высоту и ширину окна воспроизведения в пикселах, соответственно. Следующий код показывает пример создания области воспроизведения размером 400 300 пикселей:

```
<video src="video.mp4" controls width= "400" height="300"></video>
```

Размеры окна воспроизведения должны совпадать с размером видео, но лучше явно указать их, чтобы оформление страницы не искажалось до того, как видео-файл загрузится (или если видеофайл вовсе не загружается).

Атрибут poster позволяет указать изображение, которое можно использовать вместо видео:

```
<video src="video.mp4" controls poster="pict.jpg"></video>
```

Браузеры показывают это изображение в трёх ситуациях: когда первый кадр видео ещё не загрузился, атрибуту preload присвоено значение none или указанный видеофайл отсутствует.

Элемент <source>

Для совместимости с различными браузерами у нас есть не один видеофайл, а три. Один это OGV-файл, второй MP4- файл, третий WebM-файл. HTML5 обеспечивает способ сделать ссылки на все три файла с помощью элемента <source>. Каждый элемент <video> может содержать более одного тега <source>. Ваш браузер пройдётся по списку источников видео по порядку и выберет первым то, что он в состоянии воспроизвести.

Браузер проводит проверку, какое видео он сможет воспроизвести. В худшем случае он загружает каждое видео и пытается его сыграть, однако это большая трата пропускной способности. Вы сэкономите много трафика, если сообщите браузеру информацию о каждом видео. Это можно сделать атрибутом type тега <source>.

```
<video width="320" height="240" controls>
<source src="video.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src="video.webm" type='video/webm; codecs="vp8, vorbis"'>
<source src="video.ogv" type='video/ogg; codecs="theora, vorbis"'>
</video>
```

Элемент <video> определяет ширину и высоту видео, но не ссылку на видеофайл. Внутри <video> три элемента <source>. Каждый элемент <source> ссылается на отдельный видеофайл (с атрибутом src), а также дает информацию о видеоформате (в атрибуте type).

Тип атрибута - комбинация из трех блоков информации: формат файла, видеокодек и аудиокодек. Для видеофайла .ogv формат контейнера это Ogg, представленный здесь как video/ogg, строго говоря, это MIME-тип для видео-файлов Ogg. Видео-кодек Theora и аудио-кодек Vorbis. Само значение должно быть заключено в кавычки, поэтому вы должны использовать различные виды кавычек, чтобы окружить значение целиком.

```
<source src="pr6.ogv" type='video/ogg; codecs="theora, vorbis"'>
```

С WebM почти так же, но с другим MIME-типом (video/webm вместо video/ogg) и другим видео-кодеком (vp8 вместо theora) написанным в параметре codecs.

```
<source src="pr6.webm" type='video/webm; codecs="vp8, vorbis"'>
```

Видео H.264 является более сложным.

```
<source src="pr6.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
```

Преимуществом является то, что браузер проверяет атрибут type первым и смотрит, может ли он воспроизвести видеофайл. Если браузер решает, что он не может этого сделать, то не будет скачивать файл даже частично.