План

1. Возможности HTML5. История развития.
2. Использование HTML5 API Canvas.
3. CSS3. Понятие о селекторах и свойствах.
4. Классы, блоки, виды верстки. Использование шрифтов
5. Анимация, 2D и 3D эффекты
6. Фоновые изображения. Кроссбраузерная верстка
7. Особенности работы с мобильными устройствами
8. Работа с видео и аудио
9. Использование XML и XSLT.

1. Возможности HTML5. История развития.

HTML5 — язык для структурирования и представления содержимого всемирной паутины. Это пятая версия HTML. Цель разработки HTML5 — улучшение уровня поддержки мультимедиа-технологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека и простоты анализа для парсеров.

Во всемирной паутине долгое время использовались стандарты HTML 4.01, XHTML 1.0 и XHTML 1.1. Веб-страницы на практике оказывались свёрстаны с использованием смеси особенностей, представленных различными спецификациями, включая спецификации программных продуктов, например веб-браузеров, а также сложившихся общеупотребительных приёмов. HTML5 был создан как единый язык разметки, который мог бы сочетать синтаксические нормы HTML и XHTML. Он расширяет, улучшает и рационализирует разметку документов, а также добавляет единый API для сложных веб-приложений.

В HTML5 реализовано множество новых синтаксических особенностей. Например, элементы `<video>`, `<audio>` и `<canvas>`, а также возможность использования SVG и математических формул. Эти новшества разработаны для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования сторонних API и плагинов. Другие новые элементы, такие как `<section>`, `<article>`, `<header>` и `<nav>`, разработаны для того, чтобы обогащать семантическое содержимое документа (страницы). Новые атрибуты были введены с той же целью, хотя ряд элементов и атрибутов был удалён. Некоторые элементы, например `<a>`, `<menu>` и `<cite>`, были изменены, переопределены или стандартизированы. API и DOM стали основными частями спецификации HTML5. HTML5 также определяет некоторые особенности обработки ошибок вёрстки, поэтому синтаксические ошибки должны рассматриваться одинаково всеми совместимыми браузерами.

HTML5:

- В отличие от HTML4, у которого 3 валидатора (strict, transitional, frameset), у HTML5 валидатор один: <!DOCTYPE html>.
- HTML5 поддерживает MathML и SVG.
- Новые теги:
  - <section>, <article>, <aside>, <header>, <footer>, <nav>, <main>, <hgroup> (уже считается устаревшим в W3C),
  - <figure>, <figcaption>, <video>, <audio>, <source>, <track>, <picture>, <canvas>, <svg>, <math>, <embed> (для вставки контента с плагином (только)),
  - <datalist>, <keygen>, <output>, <progress>, <meter>, <dialog>,
  - <data>, <time>, <mark>, <ruby>, <rt>, <rp>, <bdi>, <wbr>,
  - <details>, <summary>, <menu>, <menuitem> (нужно использовать вместо тега <command>).
- Новые значения атрибута type для тега <input>:
  - date, datetime, datetime-local, time, month, week,

- color, email, tel, number, range, search, url.
- Новые атрибуты для тегов, например:
  - autocomplete, autofocus, placeholder, required, pattern (для регулярных выражений) и другие для тега <input>,
  - autocomplete, autofocus, placeholder, required и другие для <textarea>,
  - async для тега <script>,
  - srcset для тега <img>,
  - download для тегов <a> и <area>.
- Новые глобальные атрибуты, то есть которые могут использоваться с любым из тегов, например:
  - contenteditable, spellcheck, contextmenu, hidden, draggable/dropzone.
- Некоторые теги отмечены как устаревшие, вместо них рекомендуется использовать CSS:
  - <basefont>, <big>, , <font>, <strike>, <tt>.
- Исчезновение фреймов (<frame>, <frameset>, <noframes> (кроме <iframe>)) из-за проблем с поисковыми системами и некоторых неудобств при навигации по странице.
- Исчезновение некоторых тегов, заменённых в обновлённой спецификации на более актуальные:
  - вместо <acronym> нужно использовать тег <abbr>,
  - вместо <applet> использовать <object>,
  - вместо <dir> использовать <ul>,
  - вместо <bgsound> — тег <audio>,
  - вместо <listing> и <xmp> — теги <pre> или <code>,
  - вместо <strike> — <del> или <s>,
  - вместо <isindex> — комбинацию тега <form> и текстового поля.
- Не поддерживаются некоторые атрибуты у тегов, например:
  - charset и rev у тегов <link> и <a>,
  - coords, shape и name у тега <a>,
  - align у всех тегов,
  - background, bgcolor, text, topmargin, rightmargin, bottommargin, leftmargin, link, alink, vlink у тега <body>.
- Новые API:
  - Рисование 2D-картинок в реальном времени (Canvas),
  - Контроль над проигрыванием медиафайлов,
  - Хранение данных в браузере,
  - Редактирование,
  - Drag-and-drop,
  - Работа с сетью,
  - MIME.
- Новые элементы в DOM.

<!--...--> – тег для комментариев. При просмотре страницы браузером текст внутри этих тегов игнорируется.

!DOCTYPE – Валидатор для документа. В нём описывается какая версия HTML/XHTML используется в файле.

В HTML 4.01 было три валидатора:

- Strict (Строгий, только с рекомендуемыми опциями)
- Transitional (Переходный с не рекомендуемыми опциями)
- Frameset (с поддержкой фреймов)

В HTML5:

<!DOCTYPE html>

История языка HTML началась в 1991 году, когда сэр Тим Бёрнерс Ли представил миру HTML 1.0, в котором тогда описывалось всего лишь 20 тегов. Спецификации для языка HTML 1.0 не существовало.

В 1994 году был создан W3C — World Wide Web Consortium (Консорциум Всемирной паутины), который в 1996 году создаёт HTML 2.0, а также спецификацию для него. W3C — это специальная структура, которая развивает и создаёт различные веб-технологии, в неё может войти любая организация или частное лицо, заинтересованное в развитии интернета.

В январе 1997 года, W3C создает HTML 3.2, затем в декабре того же года HTML 4.0, а в декабре 1999 года у HTML 4.01.

В 2000 году был разработан язык XHTML 1.0, он должен был стать переходным звеном между языками HTML и XML. Задача языка XHTML 1.0 состояла в том чтобы вытеснить из вёрстки все теги и атрибуты форматирования, а также приучить разработчиков сайтов к строгому синтаксису.

В 2001 году, W3C создаёт XHTML 1.1 который полностью становится подобием языка XML, браузер Internet Explorer который занимал 95% рынка его не поддерживал.

В 2002 году, W3C начал разрабатывать XHTML 2.0, язык решили делать с нуля, что привело к проблемам совместимости с предыдущими версиями языков HTML и XHTML.

Многие разработчики и крупные игроки веб-индустрии начали проявлять недовольство политикой и работой W3C, они хотели чтобы язык обладал обратной совместимостью, а также чтобы в новом языке появились стандарты для создания веб-приложений.

История HTML5 начинается в 2004 году, когда большая часть видных деятелей веб-индустрии, а также крупных компаний таких как (Google, Mozilla, Opera, Apple и Microsoft), создают свою собственную рабочую группу под названием WHATWG (возглавил её гениальный программист — Ян Хиксон).

Перед тем как начать работать над HTML5, рабочая группа WHATWG создала две спецификации: Web Forms 2.0 (веб-формы) и Web Apps 1.0 (веб-приложения). Затем эти две спецификации сделали частью спецификации HTML5.

В 2006 году W3C всё еще продолжает работать над XHTML 2.0. В том же году руководитель консорциума сэр Тим Бёрнерс Ли написал в своём блоге, что работа над XHTML 2.0 видимо не будет иметь смысла, поскольку разработчики сайтов не желают создавать свои проекты по XML типу, а желают новых версий

HTML. Поэтому в этом же году W3C начал разработку своей версии HTML 5 (пишется через пробел), в её основе лежали наработки рабочей группы WHATWG.

Сам WHATWG работал над своей версией HTML5 (пишется без пробела), причем эта версия тоже должна была стать одной из спецификаций консорциума W3C.

В 2009 году W3C прекратил развитие XHTML 2.0 и начал разрабатывать HTML5 (решили писать без пробела) уже совместно с WHATWG.

К 2012-му году, практически все современные браузеры в мире, начинают понимать язык HTML5, хотя еще и остаются некоторые теги которые браузеры пока не понимают.

На данный момент, работа над языком HTML5 продолжается, создаются новые теги и технологии, всё это добавляется в спецификацию, сама спецификация HTML5 была опубликовна (28 октября 2014 г.).

Было установлено несколько постулатов для HTML5:

- Новые особенности должны базироваться на HTML, CSS, DOM, и JavaScript
- Уменьшение необходимости внешних программных модулей (таких как Flash)
- Улучшенная обработка ошибок
- Больше разметки - чтобы заменить скрипты
- HTML5 должен быть аппаратно независимым
- Процесс разработки должен быть общедоступен

2. Использование HTML5 API Canvas

Элемент <canvas>, добавленный в HTML5, педназначен для создания графики с помощью JavaScript. Например, его используют для рисования графиков, создания фотокомпозиций, анимаций и даже обработки и рендеринга видео в реальном времени.

«Canvas» в переводе с английского означает «холст».

Приложения от Mozilla поддерживают <canvas> начиная с Gecko 1.8 (т.е. с Firefox 1.5). Этот элемент первоначально был внедрен Apple для OS X Dashboard и Safari. Internet Explorer поддерживает <canvas> начиная с 9 версии и новее; для более ранних версий IE поддержку для <canvas> можно добавить с помощью скрипта из проекта Google's Explorer Canvas. Google Chrome и Opera 9 также поддерживают <canvas>.

Элемент <canvas> также используется технологией WebGL для аппаратного ускорения 3D-графики на вебстраницах.

Основы использования Canvas.

Чтобы создать Canvas-контекст, достаточно просто добавить элемент <canvas> в HTML-документ:

<canvas id="myCanvas" width="300" height="150">

Альтернативное содержимое, которое будет показано, если браузер не поддерживает Canvas.

</canvas>

Нужно добавить идентификатор к элементу canvas, чтобы потом обратиться к нему в JavaScript, также необходимо задать атрибуты width и height для определения ширины и высоты элемента canvas.

Для рисования внутри элемента canvas, нужно использовать JavaScript. Сначала нужно найти созданный тег canvas с помощью функции getElementById, а потом инициализировать нужный контекст. Как только это будет сделано, можно начинать рисование на канве, используя доступные API-команды выбранного контекста.

Canvas 2D API.

С помощью свойств fillStyle и strokeStyle вы можете легко настроить цвета, используемые для заливки и линий объектов. Значения цветов, используемые в этих методах, такие же как и в CSS: шестнадцатеричные коды (#F5E6AB), rgb(), rgba() или даже hsla(), если браузер поддерживает такой способ задания цвета (например, он поддерживается в Opera 10.00 и более новых версиях).

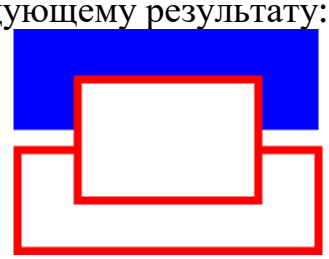
Используя метод fillRect, вы можете нарисовать прямоугольник с заливкой. С помощью метода strokeRect вы можете нарисовать прямоугольник только с границами, без заливки. Если нужно очистить некоторую часть канвы, вы можете использовать метод clearRect. Три этих метода используют одинаковый набор аргументов: x, y, width, height. Первые два аргумента задают координаты (x,y), а следующие два — ширину и высоту прямоугольника.

Для изменения толщины линий можно использовать свойство lineWidth. Пример использования функций fillRect, strokeRect, clearRect:

context.fillStyle = '#00f'; // blue

```
context.strokeStyle = '#f00'; // red
context.lineWidth = 4;
// Draw some rectangles.
context.fillRect (0, 0, 150, 50);
context.strokeRect(0, 60, 150, 50);
context.clearRect (30, 25, 90, 60);
context.strokeRect(30, 25, 90, 60);
```

Этот пример приведет к следующему результату:



Окружность и круг.

Чтобы нарисовать окружность, нужно выполнить такой код:

```
context.beginPath();
context.arc(75, 75, 10, 0, Math.PI*2, true);
context.closePath();
context.fill(); // Если нужен круг, можно залить окружность
```

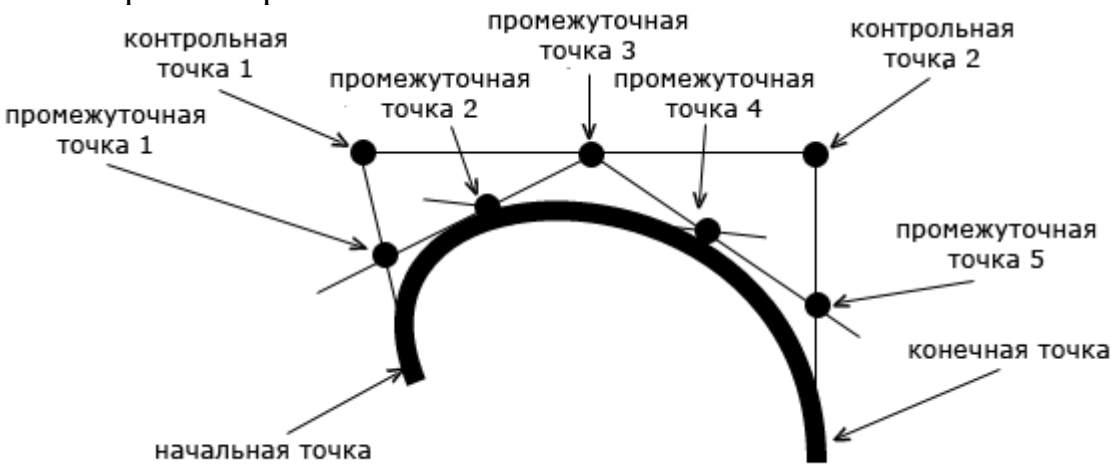
Кривые Безье.

Для создания кривых Безье в HTML5 Canvas можно использовать метод bezierCurveTo(). Кривые Безье задаются с помощью начальной точки, двух контрольных точек и конечной точки. В отличие от квадратичных кривых, кривые Безье в HTML 5 Canvas определяются двумя контрольными точками вместо одной, позволяя создавать кривые с более сложным очертанием.

Метод bezierCurveTo() выглядит следующим образом:

```
context.bezierCurveTo(controlX1, controlY1, controlX2, controlY2, endX, endY);
```

Схема построения кривой Безье:



Контуры.

Контуры Canvas позволяют рисовать фигуры любой формы. Сначала нужно нарисовать "каркас", а потом можно использовать стили линий или заливки, если

это необходимо. Чтобы начать рисование контура, используется метод beginPath(), потом рисуется контур, который можно составить из линий, кривых и других примитивов. Как только рисование фигуры окончено, можно вызвать методы назначения стиля линий и заливки, и только потом вызвать функцию closePath() для завершения рисования фигуры.

Следующий код демонстрирует рисование треугольника:

```
// Задаем свойства заливки и линий.
context.fillStyle = '#00f';
context.strokeStyle = '#f00';
context.lineWidth = 4;
context.beginPath();
// Начинаем рисовать треугольник с верхней левой точки.
context.moveTo(10, 10); // перемещаемся к координатам (x,y)
context.lineTo(100, 10);
context.lineTo(10, 100);
context.lineTo(10, 10);
// Заполняем фигуру заливкой и применяем линии
// Фигура не будет отображена, пока не будет вызван хотя бы один из этих методов.
context.fill();
context.stroke();
context.closePath();
```

Этот пример будет отображен в браузере следующим образом:

Вы также можете посмотреть более сложные примеры контуров с использованием линий, кривых и дуг.

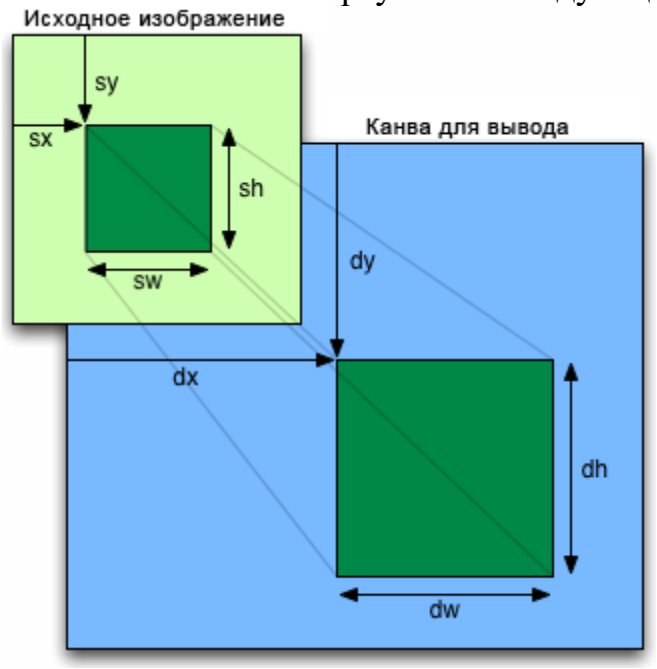
Вставка изображений в Canvas.

Метод drawImage позволяет вставлять другие изображения (img и canvas) на канву. В браузере Opera также существует возможность рисования SVG-изображений внутри элемента canvas. drawImage довольно сложный метод, который может принимать три, пять или девять аргументов:

- Три аргумента: Базовое использование метода drawImage включает один аргумент для указания изображения, которое необходимо вывести на канве, и два аргумента для задания координат.
- Пять аргументов: Используются предыдущие три аргумента и еще два, задающие ширину и высоту вставляемого изображения (в случае если вы хотите изменить размеры изображения при вставке).
- Девять аргументов: Используются предыдущие пять аргументов и еще четыре: два для координат области внутри исходного изображения и два для

ширины и высоты области внутри исходного изображения для обрезки изображения перед вставкой в Canvas.

Спецификация HTML 5 объясняет эти аргументы следующим образом:



Следующий пример показывает использование всех трех вариантов обращения к этому методу:

// Три аргумента: элемент img или canvas, координаты для вывода на канву (x,y).

context.drawImage(img_elem, dx, dy);

// Пять аргументов: элемент img или canvas, координаты для вывода на канву (x,y),

// ширина и высота для изменения размеров перед выводом

context.drawImage(img_elem, dx, dy, dw, dh);

// Девять аргументов: элемент img или canvas, координаты, ширина и высота для обрезки изображения,

// координаты для вывода на канву (x,y), ширина и высота для изменения размеров перед выводом.

context.drawImage(img_elem, sx, sy, sw, sh, dx, dy, dw, dh);

Так будет выглядеть этот пример в браузере:

Манипуляции над пикселями.

2D Context API предоставляет три метода, которые позволяют выполнять попиксельное рисование: createImageData, getImageData и putImageData.

Пиксели хранятся в объектах типа ImageData. Каждый объект имеет три свойства: width, height и data. Свойство data имеет тип CanvasPixelArray и содержит массив элементов размером width*height*4 байт; это означает, что каждый пиксель содержит цвет в формате RGBA. Пиксели упорядочены слева направо, сверху вниз, построчно.

Пока не все современные браузеры реализуют метод createImageData. В таких браузерах необходимо получать объект ImageData, используя метод getImageData.

Благодаря возможностям метода ImageData можно сделать очень многое. Например, можно отфильтровать изображение или создать математическую визуализацию (фракталы и т.п.).

Текст.

В настоящее время Text API доступен только в последних сборках движка WebKit, а также в Firefox 3.1 и выше.

Следующие свойства текста доступны для объекта контекста:

- font: Определяет шрифт текста, так же как свойство font-family в CSS)
- textAlign: Определяет горизонтальное выравнивание текста. Допустимые значения: start, end, left, right, center. Значение по умолчанию: start.
- textBaseline: Определяет вертикальное выравнивание текста. Допустимые значения: top, hanging, middle, alphabetic, ideographic, bottom. Значение по умолчанию: alphabetic.

Существуют два метода для вывода текста: fillText и strokeText. Первый отрисовывает текст, заполняя его заливкой стиля fillStyle, другой рисует обводку текста, используя стиль strokeStyle. Оба метода принимают три аргумента: собственно текст и координаты (x,y), в которых его необходимо вывести. Также существует четвертый необязательный аргумент — максимальная ширина. Этот аргумент необходим для умещения текста в заданную ширину.

Свойства выравнивания текста влияют на позиционирование текста относительно координат его вывода (x,y).

Тени.

Shadow API предоставляет четыре свойства:

- shadowColor: Определяет цвет тени. Значения допустимы в том же формате, что и в CSS.
- shadowBlur: Определяет степень размытия тени в пикселях. Эффект очень похож на гауссово размытие в Photoshop.
- shadowOffsetX и shadowOffsetY: Определяет сдвиг тени в пикселях (x, y).

Градиенты.

Свойства fillStyle и strokeStyle также могут иметь объекты CanvasGradient вместо обычных цветов CSS — это позволяет использовать градиенты для линий и заливок.

Для создания объектов CanvasGradient можно использовать два метода: createLinearGradient и createRadialGradient. Первый метод создает линейный градиент, а второй — радиальный градиент.

Как только создан объект градиента, можно добавлять в него цвета с помощью метода addColorStop.

3. CSS3. Понятие о селекторах и свойствах.

CSS (/siːɛsɛs/ англ. Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

CSS3 (англ. Cascading Style Sheets 3 — каскадные таблицы стилей третьего поколения) — активно разрабатываемая спецификация CSS. Представляет собой формальный язык, реализованный с помощью языка разметки. Самая масштабная редакция по сравнению с CSS1, CSS2 и CSS2.1. Главной особенностью CSS3 является возможность создавать анимированные элементы без использования JS, поддержка линейных и радиальных градиентов, теней, сглаживания и многое другое.

Преимущественно используется как средство описания и оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

В отличие от предыдущих версий спецификация разбита на модули, разработка и развитие которых идёт независимо.

CSS3 основан на CSS2.1, дополняет существующие свойства и значения и добавляет новые.

Нововведения, начиная с малых, вроде закругленных углов блоков, заканчивая трансформацией (анимацией) и, возможно, введением переменных.

Селектор CSS – это тот элемент, к которому мы собираемся применять CSS свойства. Слово «селектор» говорит само за себя, оно обозначает выбор.

p {color: red}
div span {background: green}
ul li {list-style: none}

Основных видов селекторов всего несколько:

- * – любые элементы.
- div – элементы с таким тегом.
- #id – элемент с данным id.
- .class – элементы с таким классом.
- [name="value"] – селекторы на атрибут (см. далее).
- :visited – «псевдоклассы», остальные разные условия на элемент (см. далее).

Селекторы можно комбинировать, записывая последовательно, без пробела:

- .c1.c2 – элементы одновременно с двумя классами c1 и c2.
- a#id.c1.c2:visited – элемент a с данным id, классами c1 и c2, и псевдоклассом visited.

В CSS3 предусмотрено четыре вида отношений между элементами.

Самые известные вы наверняка знаете:

- div p – элементы p, являющиеся потомками div.
- div > p – только непосредственные потомки

Есть и два более редких:

- div ~ p – правые соседи: все p на том же уровне вложенности, которые идут после div.
- div + p – первый правый сосед: p на том же уровне вложенности, который идёт сразу после div(если есть).

При выборе элемента можно указать его место среди соседей.

Список псевдоклассов для этого:

- :first-child – первый потомок своего родителя.
- :last-child – последний потомок своего родителя.
- :only-child – единственный потомок своего родителя, соседних элементов нет.
- :nth-child(a) – потомок номер a своего родителя, например :nth-child(2) – второй потомок. Нумерация начинается с 1.
- :nth-child(an+b) – расширение предыдущего селектора через указание номера потомка формулой, где a,b – константы, a под n подразумевается любое целое число.

Этот псевдокласс будет фильтровать все элементы, которые попадают под формулу при каком-либо n. Например:

- :nth-child(2n) даст элементы номер 2, 4, 6…, то есть чётные.
- :nth-child(2n+1) даст элементы номер 1, 3…, то есть нечётные.
- :nth-child(3n+2) даст элементы номер 2, 5, 8 и так далее.

Есть аналогичные псевдоклассы, которые учитывают не всех соседей, а только с тем же тегом:

- :first-of-type
- :last-of-type
- :only-of-type
- :nth-of-type
- :nth-last-of-type

Они имеют в точности тот же смысл, что и обычные :first-child, :last-child и так далее, но во время подсчёта игнорируют элементы с другими тегами, чем тот, к которому применяется фильтр.

На атрибут целиком:

- [attr] – атрибут установлен,
- [attr="val"] – атрибут равен val.

На начало атрибута:

- [attr^="val"] – атрибут начинается с val, например "value".
- [attr|="val"] – атрибут равен val *или* начинается с val-, например равен "val-1".

На содержание:

- [attr*="val"] – атрибут содержит подстроку val, например равен "myvalue".
- [attr~="val"] – атрибут содержит val как одно из значений через пробел.

Например: [attr~="delete"] верно для "edit delete" и неверно для "undelete" или "no-delete".

На конец атрибута:

- [attr$="val"] – атрибут заканчивается на val, например равен "myval".

Другие псевдоклассы:

- :not(селектор) – все, кроме подходящих под селектор.
- :focus – в фокусе.
- :hover – под мышью.
- :empty – без детей (даже без текстовых).
- :checked, :disabled, :enabled – состояния INPUT.
- :target – этот фильтр сработает для элемента, ID которого совпадает с анкором #... текущего URL.

Например, если на странице есть элемент с id="intro", то правило :target { color: red } подсветит его в том случае, если текущий URL имеет вид http://...#intro.

«Псевдоэлементы» – различные вспомогательные элементы, которые браузер записывает или может записать в документ.

При помощи *псевдоэлементов* ::before и ::after можно добавлять содержимое в начало и конец элемента:

Псевдоэлементы ::before/::after добавили содержимое в начало и конец каждого LI.

Новые свойства CSS3

background-origin – применяется для изменения алгоритма расчета ширины и высоты элемента.

Согласно спецификации CSS ширина блока складывается из ширины контента (width), значений отступов (margin), полей (padding) и границ (border). Аналогично обстоит и с высотой блока. Свойство box-sizing позволяет изменить этот алгоритм, чтобы свойства width и height задавали размеры не контента, а размеры блока.

Синтаксис

box-sizing: content-box | border-box | padding-box | inherit

Значения

- *content-box*

Основывается на стандартах CSS, при этом свойства width и height задают ширину и высоту контента и не включают в себя значения отступов, полей и границ.

- *border-box*

Свойства width и height включают в себя значения полей и границ, но не отступов (margin). Эта модель используется браузером Internet Exporer в режиме несовместимости.

- *padding-box*

Свойства width и height включают в себя значения полей, но не отступов (margin) и границ (border).

box-sizing – применяется для изменения алгоритма расчета ширины и высоты элемента.

Синтаксис

- box-sizing: content-box | border-box | padding-box | inherit

Значения

- *content-box*

Основывается на стандартах CSS, при этом свойства width и height задают ширину и высоту контента и не включают в себя значения отступов, полей и границ.

- *border-box*

Свойства width и height включают в себя значения полей и границ, но не отступов (margin). Эта модель используется браузером Internet Exporer в режиме несовместимости.

- *padding-box*

Свойства width и height включают в себя значения полей, но не отступов (margin) и границ (border).

border-radius

Синтаксис:

- border-radius: <радиус>{1,4} [/ <радиус>{1,4}]

Количество углов	Значение
1	Задает значение для всех 4 углов
2	Первое значение задает радиус верхнего левого и нижнего правого уголка, второе значение — верхнего правого и нижнего левого уголка.
3	Первое значение задает радиус для верхнего левого уголка, второе — одновременно для верхнего правого и нижнего левого, а третье — для нижнего правого уголка.
4	По очереди устанавливает радиус для верхнего левого, верхнего правого, нижнего правого и нижнего левого уголка.

box-shadow – добавляет тень к элементу. Допускается использовать несколько теней, указывая их параметры через запятую, при наложении теней первая тень в списке будет выше, последняя ниже. Если для элемента задается радиус скругления через свойство border-radius, то тень также получится с закругленными уголками. Добавление тени увеличивает ширину элемента, поэтому возможно появление горизонтальной полосы прокрутки в браузере.

Синтаксис:

box-shadow: [горизонтальное смещение] [вертикальное смещение] [радиус размытия] [растяжение] [цвет];

Можно использовать множество теней, а также внутренюю тень при помощи свойства insert.

Columns – универсальное свойство, которое позволяет одновременно задать ширину и количество колонок многоколоночного текста.

Синтаксис:

columns: [column-width] || [column-count]

Значения:

Комбинация свойств column-width и column-count. Порядок значения не имеет.

Дополнительно:

column-gap: значение | normal

Задаёт расстояние между колонками в многоколоночном тексте.

column-rule: border-width | border-style | цвет

В многоколоночном тексте отрисовывает линию между колонками.

Gradient – градиентом называют плавный переход от одного цвета к другому, причём самих цветов и переходов между ними может быть несколько. С помощью градиентов создаются самые причудливые эффекты веб-дизайна, например, псевдотрёхмерность, блики, фон и др. Также с градиентом элементы смотрятся более симпатично, чем однотонные.

- Линейный градиент

  Синтаксис:

  background: linear-gradient(Позиция, начальный , конечный цвета);
- Радикальный градиент

  Описание:

Радиальные по своему принципу похожи на линейные градиенты, но один цвет переходит в другой не вдоль прямой линии, а словно круги по воде вокруг точки.

Синтаксис:

background: radical-gradient(Позиция, начальный , конечный цвета);

Дополнительно

- background-image: url(путь к файлу) | none[, url(путь к файлу) | none]*
- background-attachment: fixed | scroll | local[, fixed | scroll | local]*
- text-overflow: clip | ellipsis
  - *clip* - текст обрезается по размеру области.
  - *ellipsis* - текст обрезается и к концу строки добавляется многоточие.
- word-wrap: normal | break-word | inherit
  - *break-word* - перенос строк добавляется автоматически, чтобы слово поместилось в заданную ширину блока.

4. Классы, блоки, виды верстки. Использование шрифтов

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега. При использовании совместно с тегами синтаксис для классов будет следующий:

Тег.Имя класса { свойство1: значение; свойство2: значение; ... }

Внутри стиля вначале пишется желаемый тег, а затем, через точку пользовательское имя класса.

Правила написания классов:

- в CSS перед названием селектора класса обязательно ставится точка (но при присвоении класса в HTML-документе эта точка не нужна);
- в названии классов можно использовать только буквы латинского алфавита, дефис, символ подчеркивания, цифры;
- название класса всегда должно начинаться с буквы (правильные варианты названий: .intro, .img-border, .nav_menu_01; неправильные: .2color, .-link, ._divider);
- названия классов CSS чувствительны к регистру, поэтому классы вроде .review и .Review будут восприниматься как два отдельных.

Блоки в HTML – это элементы, которые не могут находиться на одной строке с другими блоками и они отделяются абзацами. К блочным элементам относят:

заголовки – <h1>...<h6>

параграфы – <p>

элементы –

Не блочным элементами называют элементы, которые могут находиться на одной строке с другими не блочными элементами и они не отделяются абзацами.

К не блочным элементам относят:

выделить текст жирным –

выделить текст курсивом –

подчеркнуть текст – <u> и т.д.

Сегодня существует много разработчиков, которые верстают сайты с помощью таблиц html. Иные же считают, что способ этот давно устарел, что таблицы не предназначены для вёрстки, что код становится, трудно читаем, страницы грузятся дольше и т.д. Такие веб разработчики верстают свои сайты с использованием блоков, реализуемых тегом .

Любой блочный элемент состоит из набора свойств, подобно капустным листам накладываемых друг на друга. Основой блока выступает его контент (это может быть текст, изображение и др.), ширина которого задается свойством width, а высота через height; вокруг контента идут поля (padding), они создают пустое пространство от контента до внутреннего края границ; затем идут собственно сами границы (border) и завершают блок отступы (margin), невидимое пустое пространство от внешнего края границ. Порядок влияния этих свойств на блок четко определён и не может быть нарушен.

Виды верстки сайтов

- Фиксированная верстка или статическая. Вне зависимости от размеров окна браузера или устройства ширина страниц будет постоянной. Все элементы

занимают строго определенную ширину в пикселях на странице. Если разрешение экрана большое и окно браузера развернуто на весь экран, то, как правило, по бокам остается свободное место. И наоборот, на мобильных устройствах при фиксированной верстке снизу на странице появляется полоса прокрутки.

- Резиновая верстка. Страница занимает всю ширину браузера, каких бы размеров она не была и на каком бы устройстве вы не открыли эту страницу. Ширина элементов страницы задается в процентах от ширины окна, и поэтому занимают всю доступную область. При этой верстке тяжело добиться хорошего удобного дизайна при всех возможных разрешениях экранов, ведь страницы будут выглядеть по-разному.
- Табличная верстка или верстка таблицами. В данном случае сетка страницы строится с помощью таблиц. Представьте, что на странице есть главная таблица, а в ее ячейках при необходимости располагают вложенные таблицы, а в их ячейках могут быть новые таблицы и так до бесконечности. Код получается громоздким, это очень неудобно, да и надобности в этом уже нем. Сегодня так страницы уже давно не верстают. Более того, поисковики не любят такие страницы и плохо их индексируют. Если вам нужно, вы можете разместить на странице таблицу или несколько, но только не делайте с их помощью структуру страницы.
- Блочная верстка, верстка блоками или div-верстка. Это, пожалуй, самая распространенная верстка сегодня. Сетка страниц конструируется из множества блоков и , которые вложены друг в друга. Для доступа к ним используются атрибуты id и class. Откройте исходный код (Ctrl + U) практически любой веб-страницы и вы увидите такую верстку.
- Адаптивная верстка или мобильная верстка. Ее еще иногда называют респонсивная верстка. Ключевая особенность в том, что страницы хорошо адаптируются под любое разрешение экрана пользователя. Не важно, отрыли вы страницу на стационарном ноутбуке, на здоровенном широкоформатном мониторе или на смартфоне — в любом случае страница должна хорошо смотреться и быть удобной для пользователя. Это достигается тем, что используют несколько таблиц стилей под разные разрешения.
- Гибкая верстка или flex верстка. Вначале применяется известная всем блочная верстка, а потом нужные блоки превращают во флекс-контейнеры (флексбоксы). В стилях элемента указывают display: flex; Элементу можно указать направление главной оси и выравнивание. Эти возможности стали доступны с приходом CSS3. Теперь страницы становятся очень гибкими. Этот механизм подробно разбирается на продвинутых курсах в HTML Academy. Рекомендую ознакомиться, за этим настоящее и будущее верстки.
- Семантическая верстка. Она явилась логичным продолжением блочной верстки и стала доступна в HTML5. Новые теги делают станицу более структурированной. Поисковики любят такие страницы. Что это за новые теги в HTML5, можете посмотреть в другой моей статье.
- Валидная верстка или по-другому верста без ошибок. Это верстка, выполненная в соответствии со стандартами W3C. Проверить свою HTML-страницу на корректность вы можете с помощью специального валидатора W3C.

- Кроссбраузерная верстка. Как понятно из названия, при такой верстке страницы выглядят одинаково в разных браузерах. Первое, с чего обычно начинают — подключают к странице специальный CSS файл — сброс стилей.

Использование шрифтов

Тег <font> является устаревшим в HTML 4, и удален из HTML5.

Всемирный Интернет Консорциум (W3C) удалил тег <font> из их рекомендаций.

В HTML 4 каскадные таблицы стилей (cascade style sheets или, сокращенно, CSS) должны использоваться для определения вывода и отображения свойств для большинства HTML элементов.

Существуют тысячи шрифтов, которые предназначены для оформления текстов. При этом следует отметить, что число шрифтов, применяемых для набора текста на сайтах, существенно ниже. Конечно, ничто не мешает выбрать и задать, например, для заголовка, вычурный шрифт, установленный на локальном компьютере. Но если такого шрифта на компьютере пользователя нет, то текст будет отображаться шрифтом, установленным в браузере по умолчанию. Получается, что усилия разработчика пропадут даром. Есть три способа избежать этого.

- Создать надпись в графическом редакторе и вставить её на веб-страницу как изображение. Этот метод подключает всю мощь графических систем по управлению текстом, при этом допустимо включать любые шрифты. Из недостатков самый главный состоит в том, что рисунок так просто не изменишь, и придётся вновь обращаться к редактору. Так что рисунки в качестве текста применяются только для создания небольших и постоянных надписей вроде заголовков.
- Загрузить пользовательский шрифт. Действительно, если можно загружать по сети изображения, почему бы то же самое не проделать и со шрифтами?
- Воспользоваться стандартными шрифтами, встроенными в браузер и операционную систему. Этот способ является пока наиболее распространённым для указания шрифта на веб-странице.

Выбор шрифта для оформления текста осуществляется через стилевое свойство font-family. Оно устанавливает семейство шрифтов, которое будет использоваться для оформления текста. Список шрифтов может включать одно или несколько названий, разделённых запятой. Если в имени шрифта содержатся пробелы, например, Trebuchet MS, оно должно заключаться в двойные или одинарные кавычки.

Когда браузер встречает первый шрифт в списке, он проверяет его наличие на компьютере пользователя. Если такого шрифта нет, берётся следующее имя из списка и также анализируется на присутствие. Поэтому несколько шрифтов увеличивает вероятность того, что хотя бы один из них будет обнаружен на клиентском компьютере. Заканчивают список обычно ключевым словом, которое описывает тип шрифта — serif, sans-serif, cursive, fantasy или monospace. Таким образом, последовательность шрифтов лучше начинать с экзотических типов и заканчивать обобщённым именем, которое задаёт вид начертания (пример 1).

Универсальные семейства шрифтов:

- serif — шрифты с засечками (антиквенные), типа Times;
- sans-serif — рубленые шрифты (шрифты без засечек или гротески), типичный представитель — Arial;
- cursive — курсивные шрифты, обычно подставляется Comic Sans MS;
- fantasy — декоративные шрифты; как правило, в Windows используется шрифт Impact, либо текст отображается шрифтом, установленным по умолчанию;
- monospace — моноширинные шрифты, ширина каждого символа в таком семействе одинакова, например, шрифт Courier.

Если у вас на компьютере уже установлен специфический шрифт, то в стилях достаточно добавить строку.

h1 { font-family: SuperPuperFont; }

Значением свойства font-family выступает название гарнитуры шрифта, она будет применяться ко всем заголовкам <h1>. Но что увидят посетители сайта, у которых наш эффектный и редкий шрифт не установлен? Подобная ситуация наиболее вероятна, так что если браузер не распознаёт заявленный шрифт, он будет использовать шрифт по умолчанию, к примеру в Windows это Times New Roman. Весь наш тщательно продуманный шрифтовой дизайн в одночасье рассыплется и пойдёт прахом, поэтому надо поискать наиболее универсальное решение. Первое что сразу же приходит в голову — это организовать загрузку файла шрифта на компьютер пользователя и отображение текста выбранным шрифтом. По сравнению с другими методами вроде отображения текста через рисунок этот способ самый простой и универсальный.

Какие плюсы в итоге даёт нам загрузка файла шрифта с последующим применением через CSS.

- Текст легко добавлять и править.
- В браузере можно пользоваться поиском и находить желаемые фразы.
- В настройках браузера можно уменьшать или увеличивать размер шрифта добиваясь комфортного просмотра.
- Поисковые системы хорошо индексируют содержимое документа.
- Текст можно выделить и скопировать в буфер, а также перевести на другой язык.
- Параметры текста вроде межстрочного расстояния, цвета, размера и тому подобное легко менять с помощью свойств CSS.
- К тексту опять же через CSS просто добавлять разные эффекты, например тень.

Как видите, преимуществ очень много. Небольшие минусы тоже имеются и для баланса их стоит упомянуть.

- Не все версии браузеров поддерживают загружаемый шрифт и один для всех формат.

- Файл, содержащий гарнитуру шрифта, может занимать большой объём, замедляя тем самым загрузку веб-страницы.

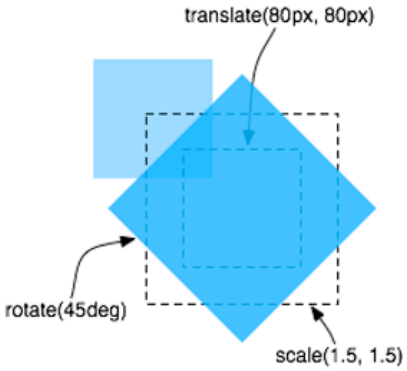
5. Анимация, 2D и 3D эффекты

CSS трансформации делят на 2 типа:

- 2D (двумерные) трансформации позволяют элементам, сгенерированным посредством CSS, трансформироваться в двумерном пространстве.
- 3D (трехмерные) расширяют само понятие трансформации, и позволяют элементам, сгенерированным посредством CSS, трансформироваться в трехмерном пространстве.

2D-трансформации хорошо поддерживаются браузерами (проще говоря, они работают почти во всех современных браузерах).Однако они не будут отображаться в IE8 и ниже.

Ситуация с трехмерными эффектами немного иная. Эти трансформации будут работать в Safari и Chrome (а также в мобильной версии Safari и в системе Android), но вряд ли в остальных браузерах. Их поддерживает IE10, чего нельзя сказать про Firefox и Opera.



Compound Transform

2D-трансформации на CSS

Если подумать о том, что монитор вашего компьютера плоский, то это 2D transforms в CSS3 – это то, с помощью чего, можно проводить различные манипуляции на этой двумерной плоскости.

- параметр transform
- параметр transform-origin
- функции трансформации (значения параметра transform)

Transform

Вы можете не задавать значение параметру transform, либо задать одно, либо несколько. Идея заключается в установке нужной вам трансформации.

```
div{
  transform: scale(1.5);
 }
```

Трансформации применяются как к блочным, так и к строчным элементам. Когда мы используем относительные величины (типа %), они будут вычисляться за счет блока элемента, а не его контейнера.

transform-origin

Данный параметр позволяет вам обозначить стартовую точку трансформации. Теперь мы можем определить исходное положение, у которого будут положения «до» и «после» трансформации.

```
div{
  transform-origin: 0, 0;
}
```

Вышеприведенный код устанавливает исходную точку на 0,0. Эта точка будет фиксирована на протяжении всего процесса трансформации.

Возможные параметры: [percentage | length | left | center | right] [percentage | length | top | center | bottom]

Первое значение отвечает за горизонтальную ось, второе – за вертикальную до тех пор, пока не будет использовано хотя бы одно ключевое слово. Если не указано ни горизонтальное, ни вертикальное значение, то выставляется стандартное значение по центру.

$$\begin{bmatrix} x_{prevCoordSys} \\ y_{prevCoordSys} \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{newCoordSys} \\ y_{newCoordSys} \\ 1 \end{bmatrix}$$

Функции трансформаций определяют то, как будет изменен элемент. За одним лишь исключением, вы должны догадаться о предназначении каждого значения по его названию.

Matrix (матрица)

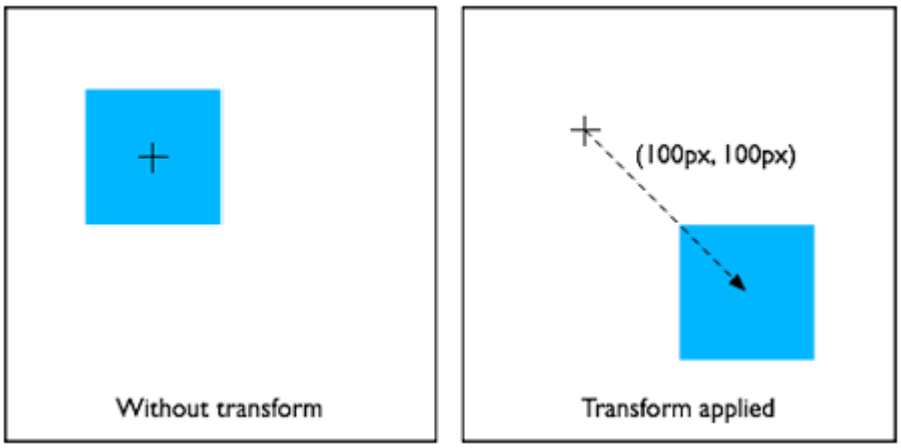
Давайте начнем с этого самого исключения, так как оно очень важно для работы со всем остальным.

Здесь идея заключается в том, что все трансформации могут быть представлены на матрице 3x3. В двумерных трансформациях мы используем лишь 2 оси на матрице с 6 значениями.

Исходная система координат (перед трансформацией) = данной матрице и в некоторых случаях новой системе координат (после трансформации).

В сущности, функция матричных трансформаций позволяет создавать пользовательские трансформации, а остальные функции являются конкретными значениями этой матрицы.

Важно помнить об указании всех 6 значений.



Translation (перемещение)

Существует 3 типа трансформации Translation, которые позволяют вам передвигать элемент из одного места в другое.

- *translate(<translation-value>[, <translation-value>])*
- *translateX(<translation-value>)*
- *translateY(<translation-value>)*

```
div {
  transform: translate(10px, 30%);
}
```

translation-value (значения переходов) могут быть представлены в виде длины, либо в процентном выражении, и каждая из 3 функций, представленных выше, производит именно то, что вы от нее ожидаете.

Все 3 перемещают элемент из одной локации в другую ровно настолько, насколько высокое значение задано в функции. Первое значение как в x, так и в y задает координаты, а последние 2 указывают направление.

Когда применяется трансформация translate, первое значение отвечает за перемещение по оси X, а второе по оси Y.

Scale (масштабирование)

Существуют также 3 функции масштабирования, которые, как вы уже догадались, позволяют вам изменять элементы в размерах.

- *scale(<number>[, <number>])*
- *scaleX(<number>)*
- *scaleY(<number>)*

```
div {
  transform: scale(2.0);}
```

Здесь, для того чтобы уменьшить или увеличить элемент, используются значения меньше, чем 1.0, либо больше, чем 1.0. Здесь вы также можете уменьшать или увеличивать элемент как по оси X, так и по оси Y.

Как и в случае с translate, сначала идет значение по оси X, а за ним – значение по оси Y.

Rotate (вращение)

У rotate есть лишь одна функция, так как определение вращения задается вокруг фиксированной точки (transform-origin), а не по осям x-y.

```
 rotate(<angle>)
div {
  transform: rotate(45deg);
}
```

Положительные углы вращают элемент по часовой стрелке, а отрицательные – против.

Skew (наклон)

Здесь все работает по тому же принципу, что и в эффектах, приведенных выше, и здесь также используются оси x-y.

- *skew(<angle> [, <angle>])*
- *skewX(<angle>)*
- *skewY(<angle>)*

С помощью skew вы можете определить наклон в обоих направлениях.

```
div{
  transform: skew(5deg, 20deg);
}
```

Вы не ограничены лишь одной трансформацией, и можете применять сразу несколько вариаций.

```
div{
  transform: translate(10px 30%) scale(2.0) rotate(45deg) skew(5deg, 20deg);
}
```

Когда вы применяете сразу несколько трансформаций, их нужно задавать по очереди таким же образом, как и по отдельности.

Трехмерные CSS-трансформации

Если вы поняли работу двумерных трансформаций, то у вас не будет проблем с пониманием принципа работы трехмерных трансформацией, так как они всего лишь дополняют некоторые параметры двумерных.

Здесь параметры трансформации также указываются с помощью функции трансформации. Здесь также нужно выставлять значение transform-origin, за исключением лишь того, что здесь также имеется координата Z.

```
div{
  transform-origin: 0, 0, 0;
}
```

- transform-style — определяет способ отображения соединенных элементов в трехмерном пространстве. Здесь есть 2 значения: flat и preserve-3d. Flat означает то, что дочерние элементы генерируются плоскими на двумерной плоскости, а preserve-3d предотвращает использование плоских элементов.
- perspective — задает ту же трансформацию, что и функция perspective transform, за исключением того, что она применяется к дочерним элементам. Значение выставляется на none, либо указываются цифры.
- perspective-origin — определяет исходное положение трансформации, приведенной выше. Существуют значения [percentage | length | left | center | right] [percentage | length | top | center | bottom].
- backface-visibility — определяет, будет ли видна задняя часть трансформируемого элемента. Можно выставить значения visible или hidden.

Функции трехмерных трансформаций основаны на функциях двумерных.

К матрице, например, была добавлена функция matrix3d, которая позволяет определять еще и третью сторону матрицы.

matrix3d(<number>, <number>, <number>, <number>, <number>, <number>, <number>, <number>, <number>;)

Опять же, требуемые функции, являются определенные значениями matrix или matrix3d

Дополнения функции translate:

translate3d(<translation-value>, <translation-value>, <translation-value>)

translateZ(<translation-value>)

Дополнения scale:

scale3d(<number>, <number>, <number>)

scaleZ(<number>)

Дополнения rotate:

rotate3d(<number>, <number>, <number>, <angle>)

rotateX(<angle>)

rotateY(<angle>)

rotateZ(<angle>)

Так как теперь у нас есть еще одно разрешение, мы можем указывать, по какой оси будет вращаться элемент.

В функции Skew не было добавлено ничего нового, но теперь у нас есть еще одна новая функция – perspective.

perspective(<number>)

Perspective задает перспективную матрицы проекции, которая являет собой нечто вроде куба, в котором элемент отображается посетителю.

Число, указанное в функции, определяет глубину и дистанцию плоскости z=0.Чем ниже значение, тем ниже пирамида, если такое объяснение будет более понятным.

Опять же, эти трехмерные трансформации не имеют хорошей поддержки, и сейчас больше рассчитаны на эксперименты, которые можно будет использовать в будущих проектах. Тем не менее, уже сейчас можно найти множество великолепных примеров реализации трехмерных трансформаций. Только нужно будет использовать браузер из семейства WebKit.

6. Фоновые изображения. Кроссбраузерная верстка

CSS фон — свойства, добавляющие фон для любого html-элемента. Каждый элемент имеет фоновый слой, который может быть прозрачным (по умолчанию), цветной заливкой и изображением. В качестве изображения может выступать градиент или картинка. Для одного элемента можно задать несколько фоновых изображений.

Фон не отображается у пустых элементов с нулевой высотой. Свойства фона не наследуются, но фон родительского блока всегда будет виден.

В качестве фона можно использовать любое подходящее для этого изображение в формате GIF, JPEG, PNG и даже SVG. Правильно подобранный фон не отвлекает внимание от текста, хорошо сочетается с цветовой гаммой веб-страницы, при этом файл с фоном желательно должен быть небольшим по объёму, чтобы быстро загружаться.

Фоновое изображение background-image

Свойство устанавливает изображение в качестве фона для элемента. Фоновым изображением может быть картинка или градиент, который задаётся с помощью функций background-image: linear-gradient(), background-image: radial-gradient() или с помощью функций повтора градиента background-image: repeating-linear-gradient(), background-image: repeating-radial-gradient(). Не наследуется.

Значения:

url('URL') – Абсолютный или относительный адрес изображения.

None –Значение по умолчанию, также удаляет изображение у элемента из группы элементов, для которых установлено фоновое изображение.

Initial – Устанавливает значение свойства в значение по умолчанию.

Inherit – Наследует значение свойства от родительского элемента.

div {background-image: url(https://html5book.ru/images/flower.png);}

table {background-image: radial-gradient(farthest-corner at 50% 50%, white, # DCECF8);}

p {background-image: linear-gradient(to top left, white, lightblue);}

Повтор фоновых изображений background-repeat

Свойство определяет, каким образом будет повторяться фоновый рисунок. Не наследуется.

Значения:

Repeat – Весь фон страницы будет заполнен фоновым рисунком. Если при этом задать background-position, то повтор будет осуществляться с указанной позиции. Значение по умолчанию.

no-repeat – Фоновое изображение не будет повторяться.

repeat-x – Фоновый рисунок повторяется от левого до правого края веб-страницы по верхнему краю страницы.

repeat-y – Фоновый рисунок повторяется от верхнего до нижнего края веб-страницы по левому краю страницы.

Initial – Устанавливает значение свойства в значение по умолчанию.

Inherit – Наследует значение свойства от родительского элемента.

div {background-repeat: no-repeat;}

div {background-repeat: repeat-x;}
div {background-repeat: repeat-y;}
div {background-repeat: repeat;}

Позиционирование фоновых изображений background-position

Свойство управляет точным расположением фонового изображения. Можно определить начальную позицию фонового изображения в виде горизонтальной и вертикальной координат посредством ключевых слов, точных абсолютных и процентных значений. Значение по умолчанию background-position: 0% 0%. Не наследуется.

Значения:

left top, left center, left bottom, right top, right center, right bottom, center top, center center, center bottom – позиционирование по горизонтали и вертикали задаётся с помощью пары ключевых слов. Если задано одно ключевое слово, второе примет значение center.

px / em/ % – указывается два значения, первое определяет расстояние между левой стороной изображения и левым краем элемента-контейнера (по горизонтали), второе значение указывает расстояние между верхней стороной изображения и верхним краем элемента-контейнера (по вертикали). Также можно использовать отрицательные значения для обрезки части изображения.

Процентное значение рассчитывается относительно самого изображения и относительно элемента-контейнера, в котором оно расположено. Например, при задании background-position:50% 50%;центр изображения совпадет с центром элемента-контейнера.

Одновременно можно комбинировать значения в px, em и %.

Initial – устанавливает значение свойства в значение по умолчанию.

Inherit –Наследует значение свойства от родительского элемента.

div {background-position: center center;}
div {background-position: 50% 100%;}
div {background-position: left;}
div {background-position: 50px;}

Можно задать фоновую картинку так, чтобы она располагалась только по низу блока:

```
div {
background-color: #FCF8F7;
height: 120px;
background-image: url(https://html5book.ru/images/flower112.png);
background-position: left bottom;
background-repeat: repeat-x;
}
```

Также благодаря свойству позиционирования, для одного блока можно использовать несколько фоновых изображений:

```
div {
width: 660px;
background-color:#E0E4EF;
height: 300px;
```

background-image:url(https://html5book.ru/images/flower112.png),
url(https://html5book.ru/images/leaf112.png),
url(https://html5book.ru/images/flower221.png);
background-repeat: repeat-x;
background-position: 0 250px, 0 150px, 0 98px;
}

Фиксация изображения на месте background-attachment

Свойство позволяет фиксировать фоновое изображение при прокрутке страницы. Не наследуется.

Значения:

Scroll – фоновое изображение прокручивается вместе с текстом и другим содержимым. Значение по умолчанию.

Fixed – предотвращает перемещение, фиксирует фоновое изображение на заднем плане.

Local – фоновое изображение прокручивается вместе с содержимым элемента.

Initial – устанавливает значение свойства в значение по умолчанию.

Inherit – наследует значение свойства от родительского элемента.

div {background-attachment: scroll;}
div {background-attachment: fixed;}
div {background-attachment: local;}

Заполнение фоном отступов и границ элемента background-clip

Свойство определяет, будет ли цвет фона ограничиваться содержимым элемента или будет простираться до внешнего края границы border. Не наследуется.

Значения:

border-box – фон простирается до внешнего края границы элемента. Значение по умолчанию.

padding-box – фона простирается до внешнего края отступов элемента.

content-box – фон окрашивает только содержимое элемента.

Initial – устанавливает значение свойства в значение по умолчанию.

Inherit – наследует значение свойства от родительского элемента.

div {background-clip: border-box;}
div {background-clip: padding-box;}
div {background-clip: content-box;}

Положение фонового изображения относительно его родительского блока background-origin

Свойство определяет, где будет позиционироваться фоновое изображение. Если одновременно задано свойство background-attachment: fixed, эффекта не будет.

Значения:

padding-box – положение элемента вычисляется относительно верхнего левого угла с внутренней стороны границы элемента. Значение по умолчанию.

border-box – положение элемента вычисляется относительно верхнего левого угла с внешней стороны границы элемента.

content-box – положение элемента вычисляется относительно верхнего левого угла содержимого.

Initial – устанавливает значение свойства в значение по умолчанию.

Inherit – наследует значение свойства от родительского элемента.

div {background-origin: padding-box;}
div {background-origin: border-box;}
div {background-origin: content-box;}

Размер изображения background-size

Свойство позволяет масштабировать фоновое изображение по вертикали и горизонтали (background-image). Оно описывает, как изображение будет растягиваться и обрезаться, чтобы полностью закрыть собой фоновую область. С помощью этого свойства изображение также можно уменьшать по ширине и по высоте. Не наследуется.

Значения:

Auto – значение по умолчанию. Высота и ширина изображения равны его оригинальным размерам.

px / em /cm – размер задается парой значений, первое значение устанавливает ширину изображения, второе — высоту. Для того, чтобы фон масштабировался вместе с текстом, размеры изображения нужно задавать в em.

% – задает размер фонового изображения в процентах от ширины или высоты элемента, которое заполняется фоном.

Cover – масштабирует изображение с сохранением пропорций так, чтобы его ширина или высота равнялась ширине или высоте блока.

Contain – масштабирует изображение с сохранением пропорций таким образом, чтобы оно целиком поместилось внутри блока.

Initial – устанавливает значение свойства в значение по умолчанию.

Inherit – наследует значение свойства от родительского элемента.

div {background-size: 300px 150px;}
div {background-size: 50% 30%;}
div {background-size: cover;}
div {background-size: contain;}

Задание фона элемента одним свойством background

Свойство позволяет описать в одном объявлении следующие свойства фона: background-color, background-image, background-position, background-size, background-repeat, background-origin, background-clip и background-attachment. Необязательно указывать все перечисленные свойства, если какое-либо свойство будет пропущено, оно примет значение по-умолчанию.

Если вы указываете в краткой записи фона свойство background-size, то его значения нужно будет записать через слеш /, чтобы отделить его от свойства background-position.

body {
background: #00ff00 url("smiley.gif") no-repeat fixed center;
}

Множественные фоны

CSS3 предоставила еще одну полезную возможность — множественные фоны, которые можно задавать как для страницы целиком, так и для конкретного блока. Например:

```
div {
width: 680px;
height: 630px;
background-image:url(https://html5book.ru/wp-content/uploads/2015/02/flower_rose.png), url(https://html5book.ru/wp-content/uploads/2015/02/love.png), url(https://html5book.ru/wp-content/uploads/2015/02/border_white.png);
background-repeat: no-repeat;
background-position: bottom right, center center, top left;
}
```

Изображения нужно перечислять в порядке наложения, самое верхнее изображение должно быть первым, а самое большое, расположенное на заднем плане — последним.

Кроссбраузерность

Существует много различных браузеров, от известных Opera, Mozilla, Chrome и Internet Explorer и до менее известных их аналогов, вот кроссбраузерность и отвечает за одинаковое отображение верстки во всех этих браузерах. По сути сейчас все современнные браузеры отлично отображают верстку, но этого нельзя сказать о старых версиях IE.

С появлением новых браузеров и выпуском очередных версий старых у верстальщика появляется еще больше работы. Значение кроссбраузерности нельзя недооценивать, так как основная задача верстальщика — это передать замысел дизайнера конечному пользователю, предоставить ему возможность просматривать страницу с любого устройства, с любым разрешением, с различной скоростью подключения к интернету, сделать его работу удобной и приятной.

Сбросьте все отступы и правила, которые браузеры могут добавлять по умолчанию, и вы уже добьетесь определенной кроссбраузерности. Этот подход еще называют как css reset. То есть создается набор стилей, который сбрасывает настройки по умолчанию.

В сети есть много различных вариантов css reset, так как каждый веб-разработчик любит делать по-своему. В самом примитивном варианте сброс стилей можно сделать так:

```
{
padding: 0;
margin: 0;
}
ul{
list-style: none;
}
```

То есть мы просто убрали все внутренние и внешние отступы и всех элементов, а также убрали маркеры у списков, потому что сегодня мало кто

использует их. Цель css reset – позволить вам начать описание стилей с чистого листа.

Под определением кроссбраузерной верстки понимается код html (верстка), который корректно выводится во всех имеющихся браузерах. В данном случае понятие «корректности» означает одинаково точное расположение деталей страницы и бесперебойную работу ее функционала. Простыми словами, вид любого сайта должен быть одинаков вне зависимости от того, в каком браузере он открыт. Для разработчиков веб-страниц эта задача имеет повышенную сложность. Все связанно с тем, что каждый из браузеров способен отображать одинаковые элементы по-разному. Обновленные версии также могут отображаться браузерами с определенными ошибками. Это связанно с тем, что разработчики обновляют настройки веб-обозревателя не одновременно.

Даже при наличии общепринятых мировых стандартов, призванных для построения веб-страниц, ряд разработчиков регулярно внедряет новые технологии и правила в настройки браузера. Эти обновления позволяют достигать, казалось бы, невозможных высот в создании динамичных и многофункциональных сайтов. Разработчики ставят перед собой цели поддержания их браузером новых скриптов и библиотек. Такие тенденции не могут быть одинаково хорошими для простых пользователей. Одни из них рады тому, что обновленный браузер отображает наиболее продвинутые страницы, у других же сайты открываются некорректно. Для того, чтобы избежать данной проблемы, верстальщик должен осуществить запись кода, отображаемый в различных обозревателях одинаково.

Процесс кроссбраузерной верстки должен обеспечивать наиболее корректную демонстрацию сайтов как на обновленных, так и на устаревших веб-обозревателях. В работу верстальщика входит функция по созданию кода, который бы подходил к каждому из них. Для выполнения этой работы требуются специальные знания и навыки.

Выполнение работы начинается с общения с заказчиком, которому необходимо донести мысль о том, что мир веб-разработок не идеален, а значит для получения самого достойного результата, будет необходимо пожертвовать определенными моментами. Дизайнеры и программисты, работающие над созданием конкретного сайта, ставят своей целью:

- максимальную схожесть исходного макета с пожеланиями заказчика;
- достижение наиболее корректного отображения страницы на самых используемых браузерах.

7. Особенности работы с мобильными устройствами

В мобильных приложениях роль CSS особенно велика в связи с обычно малым размером экранов мобильных устройств и особенностями взаимодействия и ожидания. При работе с мобильными устройствами используются разные средства ввода: пальцы, стилусы, кнопки. Необходимость присматриваться к мелкому размеру текста может вызвать у пользователя раздражение. Таблицы CSS являются также основой адаптивного дизайна мобильных приложений, включающего набор методик использования CSS для экранов разных размеров. Учитывая особенности мобильных устройств - размер окна, ориентацию (портретную или ландшафтную), соотношение сторон, поддержку устройством тех или иных стилей CSS и т.д. - может возникнуть необходимость определить какой-то особенный стиль для мобильных устройств определенного класса.

HTML5 на полный экран:

В браузерах android — встроенном до версии 4.3 и других, например Chrome, существует только одно решение данной проблемы. Ширина и высота документа должна соответствовать экрану устройства, тогда нужного эффекта мы можем добиться следующим js кодом:

window.addEventListener("load", function() { window. scrollTo(0, 0); });

Google maps использует этот способ для полноэкранного отображения своего контента. Так же, можно легко избежать появления адресной строки при скроле пальцами:

document.addEventListener("touchmove", function(e) { e.preventDefault() });

В браузерах Google Chrome, Firefox OS, Firefox для android, BlackBerry OS 10 и Amazon Silk (браузер, разработанный для Kindle Fire) мы можем использовать W3C Fullscreen API.

Браузеры iPhone iOS, Android, Chrome под Android имеют разные реализации полноэкранного отображения содержимого. IE11 на Windows Mobile еще и требует разрешения на переход в полноэкранных режим.

Разные браузеры требуют применения специфических префиксов, поэтому приходится создавать универсальный код:

```
var body = document.documentElement;
if (body.requestFullscreen) {
  body.requestFullscreen();
} else if (body.webkitrequestFullscreen) {
  body.webkitrequestFullscreen();
} else if (body.mozrequestFullscreen) {
  body.mozrequestFullscreen();
} else if (body.msrequestFullscreen) {
  body.msrequestFullscreen();
}
```

Главное, что нужно запомнить: активацию полного экрана можно производить только после действия пользователя, например касания пальцем.

На iPhone, начиная с iOS 7.1, Apple использует атрибут minimal-ui мета-тега viewport, который позволяет отобразить минимальные элементы

пользовательского элемента при портретной ориентации и скрыть все элементы пользовательского интерфейса при альбомной ориентации. Функция недоступна на iPad.

```
<meta name="viewport" content="width=devicewidth, minimal-ui">
```

При использовании данной возможности, нужно уделить особое внимание области вдоль краев.

Как определить активацию minimal-ui:

```
@media (device-height: 568px) and (height: 529px),
(device-height: 480px) and (height: 441px) {
/* minimal-ui is active */
}
```

В iOS, на iPhone и iPad, и Chrome на Android, мы можем установить полноэкранный ярлык на домашнем экране. Web-приложение может быть запущено вне браузера. Для использования данной возможности нужно добавить несколько мета-тегов:

```
<!-- for ios 7 style, multi-resolution icon of 152x152 -->
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-barstyle"
content="black-translucent">
<link rel="apple-touch-icon" href="icon-152.png">
<!-- for Chrome on Android, multi-resolution icon of 196x196 -->
<meta name="mobile-web-app-capable" content="yes">
<link rel="shortcut icon" sizes="196x196" href="icon-196.png">
```

В случае с Firefox OS и Firefox на Android, мы можем создать полноэкранное приложение на домашнем столе, создав JSON и используя JavaScript API.

Canvas API — bitmap-ориентированный интерфейс, позволяющий работать с изображениями, загруженными из файла. Например, если создать canvas с шириной 200, мы получим изображение шириной 200 точек и в документе оно будет отображено с шириной 200 css-точек, независимо от разрешения.

Это означает, что на iPhone 5S изображение будет отображено шириной 400 реальных точек экрана, а на Nexus 5 — 600 точек.

Если вы хотите создать изображение высокого разрешения, можно просто увеличить оригинал вдвое:

```
<canvas width="400" height="400" style="width: 200px; height: 200px"></canvas>
<script>
document.queryselector("canvas").getContext("2d").scale(2, 2);
</script>
```

Обратите внимание, что увеличение размера canvas, так же, увеличивает использование памяти и процессора при работе с ними.

Новые типы элементов HTML, такие как type=email, могут способствовать адаптации виртуальной клавиатуры мобильного устройства к активному полю ввода.

Для элемента type=number, Android и Windows Phone отображают полную цифровую клавиатуру, а iOS только цифры.

Если добавить атрибут pattern=»[0-9]*», в iOS мы получим полную цифровую клавиатуру, как на остальных платформах:

<input type="number" pattern="[0-9]">*

Этот же трюк можно использовать и для type=password, для определенных полей:

<input type="password" pattern="[0-9]">*

Если вы сталкиваетесь с адаптивным веб-дизайном, вы используете мета-тег viewport и набор css правил для разных размеров экрана.

Windows 8 и 8.1 позволяет разместить на одном экране браузер и другие полноэкранные приложения, включая классический рабочий стол. В данном случае и когда ширина окна становится меньше 1024 точек, IE автоматически переходит в мобильный режим отображения и пускает по боку все наши правила адаптивной разметки.

Для избежания такого поведения, мы можем использовать css viewport:

```
@media only screen and (max-width: 400px) {
@-ms-viewport { width: 320px; }
}
```

Используя поле <input type=datetime>, мы на большинстве современных браузеров получим диалог выбора даты и времени. Так было и на мобильных устройствах до появления iOS 7 и Chrome 26 на Android. Начиная с этих версий, браузеры обрабатывают его как простое текстовое поле.

Разработчики этих браузеров не потрудились нормально документировать свойство type=datetime-local, которое заставляет выводить диалог выбора даты и времени:

<input type="datetime-local">

Стандарт html5 включает новый атрибут элементов: ContentEditable. Он может применяться к любому html элементу, включая , <table> и <ul>.

При пользовательской активации такого элемента, появляется виртуальная клавиатура и он становится редактируемым. Для сохранения изменений, нужно обрабатывать атрибут innerHTML элемента.

В iOS пользователь имеет возможность выделять текст жирным, курсивом или подчеркиванием.

```
<ul contenteditable>
<li>static item in the hTML
</ul>
```

Начиная с iOS 7, Safari для iPhone и iPod touch поддерживает возможность смены раскраски фона за полу-прозрачным интерфейсом самого Safari в цвет фона вашего сайта.

Есть простой способ обойти эту особенность и заменить цвет фона на любой другой:

```
body {
/* for safari's tint */
background-color: blue;
/* for the document's body background color */
background-image: linear-gradient(to bottom, red 0%, red 100%);
```

}

При добавлении ярлыка сайта на домашний экран мобильного устройства, его наименование берется из тега <title>. Из-за особенностей SEO, зачастую в этом теге может быть совсем не то, что мы хотим показать пользователю.

Есть способ задать название ярлыка на домашнем столе мобильного устройства.

Недокументированный тег Safari на iOS:

<meta name="apple-mobile-web-app-title" content="Myname">

Для Windows Phone и планшетов на Windows 8.x можно указать название ярлыка используя специальный тег application-name:

<meta name="application-name" content="Myname">

Пользователь может закрепить сайт из IE на стартовый экран Windows 8.x или Windows Phone (последние версии). Вы можете создать живой заголовок, обновлять информацию и анимировать значок даже когда сайт не открыт.

Для этого необходимо несколько мета-тегов. Для начала рассмотрим добавление значка:

<meta name="msapplication-TileImage" content="tile-background.png">

<meta name="msapplication-TileColor" content="#FF0000">

Для «оживления», мы можем использовать Badge Polling URI (начиная с IE10) или Notification Polling URI (начиная IE11).

Первый может обновлять значок, количество непрочитанных элементов. Второй может отображать новые сообщения и информацию.

В обоих случаях, мы можем указать частоту обновления информации:

<meta name="msapplication-badge"
content="frequency=60;polling-uri=http://host/ badge.xml">
<meta name="msapplication-notification"
content="frequency=30;polling-uri=http://host/ live.xml">

Для упрощения работы с живыми плитками, вы можете использовать Live Tile creator от Microsoft или специальный плагин для WordPress.

Мобильные браузеры выглядят как их старшие собратья, но ведут себя, зачастую, совсем не так. Пример: iOS Safari и его вкладки. Если открыть три вкладки: хабр, gmail и вконтакте, динамическое содержимое будет обновляться только на одной, активной в данный момент вкладке. Никакие js скрипты не будут выполняться на фоновых вкладках, в отличие от десктопных браузеров.

Есть, по крайней мере, один способ это исправить. Можно проверять обновление данных на сервере и изменять название вкладки или выводить алерт, в случае необходимости.

Данный способ основан на обычном мета-теге refresh, который определяет перезагрузку страницы. Safari обрабатывает этот мета-тег даже для фоновых вкладок. И нам осталось только определить интервал перезагрузки страницы.

<meta http-equiv="refresh" content="2">

<script>

var mr = document.queryselector("meta"); setInterval(function() { mr.content=mr.content; }, 1000);

</script>

8. Работа с видео и аудио

HTML5 аудио предоставляет улучшенные возможности работы с аудио контентом. До недавнего времени единственным способом добавления звуковых файлов на веб-страницы было интегрирование фонового звука с помощью тега <bgsound>, который проигрывался во время просмотра пользователем страницы без возможности выключения.

Благодаря добавлению в спецификацию HTML5 нового элемента <audio>, появилась возможность добавлять аудио содержимое со встроенным программным интерфейсом без привлечения подключаемых модулей.

HTML5-элемент <audio> используется для внедрения звукового контента в веб-страницы. В общем виде html-разметка имеет следующий вид:

```
<audio src="name.ogg" controls></audio>
```

Атрибут controls добавляет отображение браузерами интерфейса управления аудио плеера — кнопки воспроизведения, паузы, громкости.

В настоящий момент не существует аудио формата, который бы работал во всех браузерах, поэтому для обеспечения доступности контента максимально широкой аудитории рекомендуется включать несколько источников звука, представленных с использованием атрибута src элемента <source>. Одновременно можно добавить резервный контент для браузеров, которые не поддерживают элемент <audio>.

```
<audio controls>
  <source src="name.ogg" type="audio/ogg">
  <source src="name.mp3" type="audio/mpeg">
  <a href="sounds/name.mp3">Скачать name.mp3</a>
</audio>
```

Атрибуты тега <audio>:

Autoplay – Автоматическое воспроизведение аудио файла сразу же после загрузки страницы.

Controls – Указывает браузеру, что нужно отобразить базовые элементы управления воспроизведением (начинать и останавливать воспроизведение, переходить в другое место записи, регулировать громкость).

Loop – Циклическое воспроизведение аудио файла.

Muted – Выключает звук при воспроизведении аудио файла.

Preload – Атрибут, отвечающий за предварительную загрузку аудио контента. Не является обязательным, некоторые браузеры игнорируют его. Возможные значения:

- auto — браузер загружает аудио файл полностью, чтобы он был доступен, когда пользователь начнет его воспроизведение.
- metadata — браузер загружает первую небольшую часть аудио файла, чтобы определить его основные характеристики.
- none — отсутствие автоматической загрузки аудио файла.

Src – Содержит абсолютный или относительный URL-адрес аудио файла.

Аудио кодек (декодер) представляет собой программу для преобразования цифровых данных в формат звукового файла или звукового потока. Популярными аудио форматами являются следующие:

MP3 — самый популярный аудио формат, использующий сжатие с потерями и позволяющий уменьшить размер файла в несколько раз. В силу лицензионных отчислений не поддерживается Firefox и Opera.

AAC (Advanced Audio Codec) — закрытый кодек, аналог MP3, но по сравнению с последним, поддерживает более высокое качество звука при сходном размере файла.

Ogg Vorbis — бесплатный формат с открытым кодом, поддерживается в Firefox, Opera и Chrome. Обеспечивает хорошее качество звука, но недостаточно широко поддерживается аппаратными проигрывателями.

Элемент <source> используется для добавления нескольких медиа-ресурсов для <audio> и <video>. Указывает на альтернативные видео/аудио файлы, которые браузер может выбрать из предложенных на основании поддерживаемого им типа носителя или кодека.

Атрибуты тега <source>:

Media – Определяет тип медиа-устройства (т.е. для каких устройств оптимизирован файл).

Src – Содержит абсолютный или относительный URL-адрес медиафайла.

Type –Определяет MIME-тип медиафайла.

Элемент <track> используется в качестве дочернего элемента <audio> и <video>. Добавляет текстовую дорожку для субтитров, заголовков медиафайлов или другой текстовой информации, которая должна быть видна во время воспроизведения аудио или видео файла.

Атрибуты тега <track>:

Default – Указывает, что данная дорожка воспроизводится по умолчанию. Только один элемент <track> может содержать данный атрибут.

Kind –Указывает тип текстовой дорожки, по умолчанию выводятся субтитры (subtitles). Принимаемые значения:

- captions — перевод диалогов и звуковых эффектов, отображаемый в виде текста поверх видео (для глухих пользователей).
- chapters — добавляет названия глав в виде списка для навигации по медиафайлу.
- descriptions — добавляет звуковое описание происходящего в видео (для слепых пользователей).
- metadata — метаданные, используемые скриптами, не отображаются для пользователей.
- subtitles — текстовое дублирование звуковой дорожки видео, отображается в виде субтитров к видео.

Label – Добавляет название дорожки. Если этот атрибут не задан, браузер применит значение по умолчанию.

Src – Содержит абсолютный или относительный URL-адрес аудио- или видео файла.

Srclang – Язык воспроизводимой дорожки.

С помощью css-стилей можно придать аудио плееру необычный вид. Воспроизведение управляется с помощью небольшого скрипта (используется библиотека jQuery), звук появляется при наведении на пластинку.

Для того, чтобы вставить видео на сайт, используется тег video.

<video src="path/to/video/file.mp4"></video>

У данного тега есть атрибут src, в который мы должны прописать путь до нашего видеоролика.

Чтобы мы смогли увидеть элементы управления, такие, как шкала времени, воспроизведение/пауза, управление громкостью и полноэкранный режим, существует атрибут controls.

<video src="video.mp4" controls></video>

Если вам нужно, чтобы видео воспроизводилось сразу при загрузке страницы, используйте атрибут autoplay.

<video src="video.mp4" controls autoplay></video>

Атрибут loop используется для того, чтобы зациклить видео, т.е. для того, чтобы видео сразу же запускалось снова после того, как оно закончилось.

<video src="video.mp4" controls autoplay loop></video>

Есть еще такой интересный атрибут как preload. Как понятно из названия, он отвечает за предзагрузку видео. У него 3 значения:

- none - означает, что никакой предзагрузки не будет. Не будет вообще никакой информации, даже такой, как длительность, уровень громкости и т.д.
- metadata - это значение, наоборот, покажет нам ту информацию, которую не покажет значение none
- auto - подгружает видео сразу после загрузки страницы, чтобы человек мог его сразу запустить и не ждать, пока оно загрузится. Что-то вроде полоски на YouTube, но тут она не отображается

Понятно, что если у вас стоят сразу 2 атрибута - preload и autoplay, то весь смысл атрибута preload пропадает.

<video src="video.mp4" controls preload="auto"></video>

Конечно же, у данного тега есть такие атрибуты как width и height, которые отвечают за ширину и высоту видео.

<video src="video.mp4" controls autoplay widht="500" height="500"></video>

При помощи атрибута poster вы можете выставить свою картинку, которая будет показываться до того, как вы воспроизведете видео.

<video src="video.mp4" controls poster="poster.png"></video>

Для того, чтобы выключить звук у видео, существует атрибут muted.

<video src="video.mp4" controls autoplay muted></video>

Также, в html5 video есть возможность указать промежуток времени, с которого начнется просмотр и когда он закончится. Для этого достаточно указать #t= после имени видеофайла и указать время начала и конца через запятую.

<video src="video.mp4#t=3,5" controls></video>

В примере выше мы указали, что видео должно начаться с 3 секунды и закончиться на 5.

Если вы хотите указать только время, с которого должен начаться просмотр, то можно указать только одно значение

<video src="video.mp4#t=3" controls autoplay></video>

В примере выше видео будет начинаться с 3 секунды и идти до конца.

Если же вы, наоборот, хотите указать, чтобы видео начиналось с самого начала и шло до какого-то определенного значения, то просто оставьте первый параметр пустым, поставьте запятую и укажите второй параметр - время конца видео.

<video src="video.mp4#t=,5" controls></video>

В примере выше видео начнется с самого начала и будет идти до 5-ой секунды.

Возможность встраивать видео в html появилась недавно и понятно, что старые браузеры не поддерживают данной возможности. Чтобы это исправить, мы можем написать им какой-то текст между тегами <video></video>

<video src="video.mp4" controls>
К сожалению, ваш браузер не поддерживает HTML5 Video.
</video>

Или вы можете встроить сюда какой-то другой плеер, например, на flash или javascript, и тогда видео смогут просмотреть даже пользователи старых браузеров.

Каждый современный браузер поддерживает свой формат видео, а старые браузеры имеют поддержку еще хуже. Чтобы поддерживать несколько форматов видео, существует тег source.

<video controls autoplay>
<source src="video.mp4"></source>
<source src="video.ogv"></source>
</video>

При парсинге страницы браузер выберет тот формат, который он поддерживает, и подключит его. Файл подключается только один.

У тега source есть атрибут type, в котором мы указываем MIME тип и кодеки.

<video controls autoplay>
<source src="video.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'></source>
<source src="video.ogv" type='video/ogg; codecs="theora, vorbis"'></source>
</video>

Указывать тип файла не обязательно, но желательно. Все дело в том, что чтобы определить, что это за файл, браузеру придется скачать его начало.

9. Использование XML и XSLT

Предположим, web-узел вашей компании использует основанное на XML программное обеспечение фирмы Commerce One, в котором для безопасной коммуникации через Интернет применяется Java Message Service (JMS). Ваша деятельность была настолько успешной, что вы только что поглотили своего конкурента. К сожалению, для своего узла в Интернете ваш бывший конкурент использует другой основанный на XML продукт, RosettaNet. Как вам теперь преобразовать заказ на покупку xCBL Commerce One, написанный на XML, в заказ на покупку RosettaNet, также написанный на XML, но совершенно на другом диалекте?

Разумеется, применить XSLT. Такого рода XML-XML преобразования становятся все более и более распространенными. Все больше компаний применяют JMS для безопасных коммуникаций через Интернет, и поскольку JMS выполняется в Java, будет разумным связать JMS с основанными на Java процессорами XSLT, такими, как Xalan или Saxon.

Основная задача XSLT состоит не просто в замене одного элемента на другой, но в полной реорганизации содержимого XML-документа. Например, вам может потребоваться реорганизовать planets.xml в терминах плотности планет при помощи XSLT для создания нового XML-документа:

```
<?xml version="1.0" encoding="UTF-8"?>
<DATA>
<DENSITY>
<VALUE>.983</VALUE>
<NAME>Mercury</NAME>
<MASS>.0553</MASS>
<DAY>58.65</DAY>
<RADIUS>1516</RADIUS>
</DENSITY>
<DENSITY>
<VALUE>.943</VALUE>
<NAME>Venus</NAME>
<MASS>.815</MASS>
<DAY>116.75</DAY>
<RADIUS>3716</RADIUS>
</DENSITY>
<DENSITY>
<VALUE>1</VALUE>
<NAME>Earth</NAME>
<MASS>1</MASS>
<DAY>1</DAY>
<RADIUS>2107</RADIUS>
</DENSITY>
</DATA>
```

Мы рассмотрим преобразование, которое полностью меняет содержимое planets. xml, оставляя только небольшой код HTML и код JavaScript для отображения нескольких кнопок в браузере.

До сих пор мы создавали новые элементы только при помощи элементов буквального результата, то есть рассматривая новые элементы как текст и встраивая их в таблицу стилей. Но, как мы увидим в этой главе, не всегда возможно знать имена создаваемых новых элементов. Можно состыковать вместе создаваемые элементы по ходу дела, рассматривая их как сырой текст, но это явная недоработка, поскольку разметка трактуется как текст. В этой главе мы начнем применять элементы XSLT <xsl:element>, <xsl:attribute>, <xsl:processing-instruction> и <xsl:comment> для создания новых элементов, атрибутов, инструкций обработки и комментариев на этапе выполнения. Хорошее знание этих элементов необходимо при реорганизации содержимого XML.

Мы также рассмотрим использование режимов XSLT для осуществления нескольких преобразований с документом и сориентируемся, как применять только один из нескольких подходящих шаблонов.

В большей части этой главы исследуются возможности элемента <xsl:output>, с краткого обзора которого я и начну.

Элемент <xsl:output>

С элементом <xsl:output>, элемент мы впервые познакомились в главе 2 и использовали его, главным образом, для задания типа результирующего документа. Этот тип может задать, например, будет ли процессор XSLT записывать инструкцию обработки XML, <?xml version="1.0"?>, в начале документа, а также задать тип MIME (такой, как "text/xml" или "text/html") документов, отправляемых процессором XSLT из web-сервера браузеру. Кроме того, если вы установите выходной тип в HTML, большинство процессоров XSLT смогут распознать, что не всем элементам HTML необходимы закрывающие или открывающие теги и т. п.

В следующем списке перечислены атрибуты <xsl:output>:

cdata-section-elements (необязательный). Задает имена тех элементов, чье содержимое должно выводиться как разделы CDATA. Принимает значения списка QName, разделенного символами-разделителями;

doctype-public (необязательный). Задает открытый идентификатор, который будет использован в объявлении <!DOCTYPE> в выходных данных. Устанавливается в строковое значение;

doctype-system (необязательный). Задает системный идентификатор, который будет использован в объявлении <!DOCTYPE> в выходных данных. Устанавливается в строковое значение;

encoding (необязательный). Задает кодировку символов. Устанавливается в строковое значение;

indent (необязательный). Определяет, будет ли выходной документ выровнен с отражением структуры вложенности. Устанавливается в yes или no;

media-type (необязательный). Задает тип MIME вывода. Устанавливается в строковое значение;

method (необязательный). Задает формат вывода. Принимает значения "xml", "html", "text" или допустимое имя QName;

omit-xml-declaration (необязательный) Определяет, будет ли включено в вывод объявление XML. Устанавливается в "yes" или "no";

standalone (необязательный). Определяет, будет ли включено в вывод отдельное объявление XML, и если да - устанавливает его значение. Устанавливается в yes или no;

version (необязательный). Задает версию вывода. Принимает значение допустимого NMToken.

Чаще всего используется атрибут method, поскольку именно он определяет требуемый тип выходного дерева. Официально методом выводаметод вывода по умолчанию является HTML, при условии, что выполняются все три следующих условия:

– корневой узел результирующего дерева имеет дочерний элемент;

– в названии элемента документа результирующего дерева присутствует часть "html" (в любой комбинации верхнего и нижнего регистров) и пустой URI пространства имен;

– все текстовые узлы перед первым дочерним элементом корневого узла содержат только символы-разделители.

Если выполняются все три этих условия, то по умолчанию метод вывода устанавливается в HTML. В ином случае методом вывода по умолчанию является XML.

Тем не менее, не стоит полагаться на установки метода вывода по умолчанию, лучше явно присвоить этому атрибуту значение. Три распространенных значения атрибута method - "html", "xml" и "text", и мы познакомимся с ними в следующих разделах.

Метод вывода: HTML

Для метода вывода HTMLметод вывода;HTML процессор XSLT должен предпринять определенные действия. Например, для этого метода атрибут version определяет версию HTML. Значение по умолчанию - 4.0.

Этот метод не должен добавлять завершающий тег для пустых элементов. (Для HTML 4.0 пустыми элементами являются <AREA>, <BASE>, <BASEFONT>,
, <COL>, <FRAME>, <HR>, <IMG>, <INPUT>, <ISINDEX>, <LINK>, <META> и <PARAM>.) Метод вывода HTML должен распознавать названия элементов HTML независимо от регистра.

В соответствии с W3C, метод вывода HTML не должен скрывать содержимое элементов <SCRIPT> или <STYLE>. Например, следующий элемент буквального результата:

<SCRIPT>
if (x < y)
</SCRIPT>

или следующий, использующий раздел CDATA:

<SCRIPT>
<![CDATA[if (x < y)]]>
</SCRIPT>

должен быть преобразован в:

<SCRIPT>

if (x < y)
</SCRIPT>

Метод вывода HTML не должен также подавлять символы <, встречающиеся в значениях атрибутов.

При установке метода вывода в HTML процессор может учесть атрибут выравнивания. Если этот атрибут установлен в yes, процессор XSLT может добавить (или удалить) символы-разделители для выравнивания результирующего документа, поскольку это не влияет на отображение документа в браузере. Для метода вывода HTML значением по умолчанию является "yes".

Как вы могли предположить, метод вывода HTML завершает инструкции обработки при помощи >, а не ?>, а также поддерживает отдельные атрибуты, как и HTML. Например, тег

<TD NOWRAP="NOWRAP">

будет преобразован в:

<TD NOWRAP>

Для этого метода можно установить атрибут media-type, значением по умолчанию для которого является "text/html". Метод HTML не должен убирать символ &, который появляется в значении атрибута, если сразу за ним следует фигурная скобка. Атрибут encoding задает используемую кодировку. Если присутствует элемент <HEAD>, этот метод вывода должен добавить элемент <META> сразу же после тега <HEAD>, определяя кодировку символов:

<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=utf-8">
.
.
.

При помощи атрибутов doctype-public или doctype-system можно вывести объявление типа документа непосредственно перед первым элементом, как мы увидим при преобразовании XML в XHTML.

Таковы правила вывода HTML. Ниже приведен пример преобразования из XML в HTML с небольшими отклонениями. В этом случае таблица стилей будет фактически генерировать код JavaScriptJavaScript;генерация из XML, демонстрируя создания JavaScript при помощи XSLT. В частности, мы прочитаем planets.xml и создадим новый документ HTML, отображающий три кнопки - по одной для каждой из трех планет в planets.xml. При щелчке на кнопке на странице будет выведена масса соответствующей планеты.

Все, что нам понадобится, - это два элемента <xsl:for-each>: один для прохода в цикле по трем планетам и создания для каждой кнопки HTML; и один для прохода по планетам и создания для каждой функции JavaScript. В качестве имен функций JavaScript я воспользуюсь названием планет; при вызове функция выведет массу соответствующей планеты. Заметьте, что для создания нужного кода JavaScript нужно всего лишь применить элемент <xsl:value-of> для получения названий и масс планет.