Создание статичных веб-страниц
с помощью языков XHTML и CSS

# Урок №2
# Списки. Кодировки. Символы


Введение в языки разметки.

Спецификации языка HTML.

Структура HTML-документа.

Классификация элементов.

# Содержание

# Маркированные и нумерованные списки

## Маркированный список

Маркированный, или еще неупорядоченный список применяется для форматирования перечислений, последовательность элементов которых не имеет значения. К примеру, на странице требуется расположить перечень независимых друг от друга услуг, оказываемых некоторой организацией.

Неупорядоченный список создается элементами двух типов:

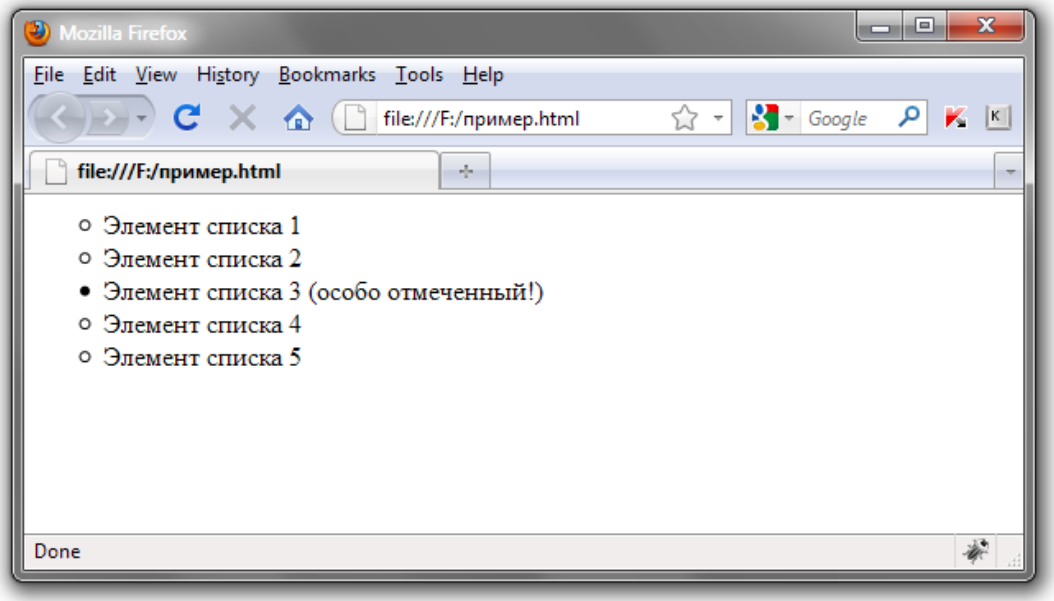
- `<ul>` — Unordered List, неупорядоченный список. Данный элемент создает, собственно, сам список, указывает его начало и конец.
- `<li>` — List Item, элемент списка.

По умолчанию список отмечен круглыми чёрными маркерами. Но это положение вещей можно легко изменить. Для указания типа маркера всего списка или его отдельного элемента, ранее использовали атрибут `type`.

Атрибут `type` может принимать следующие значения:

- `"disc"` — маркер в виде диска. Используется по умолчанию.
- `"circle"` — маркер в виде окружности (внутри пустой)
- `"square"` — маркер в виде заполненного квадрата

Маркированный список



Пример использования:

```
<ul type="circle">
    <li>Элемент списка 1</li>
    <li>Элемент списка 2</li>
    <li type="disc">Элемент списка 3 (особо
        отмеченный!)</li>
    <li>Элемент списка 4</li>
    <li>Элемент списка 5</li>
</ul>
```

На данный момент атрибут `type` считается устаревшим и должен быть заменен свойствами CSS.

## Нумерованный список

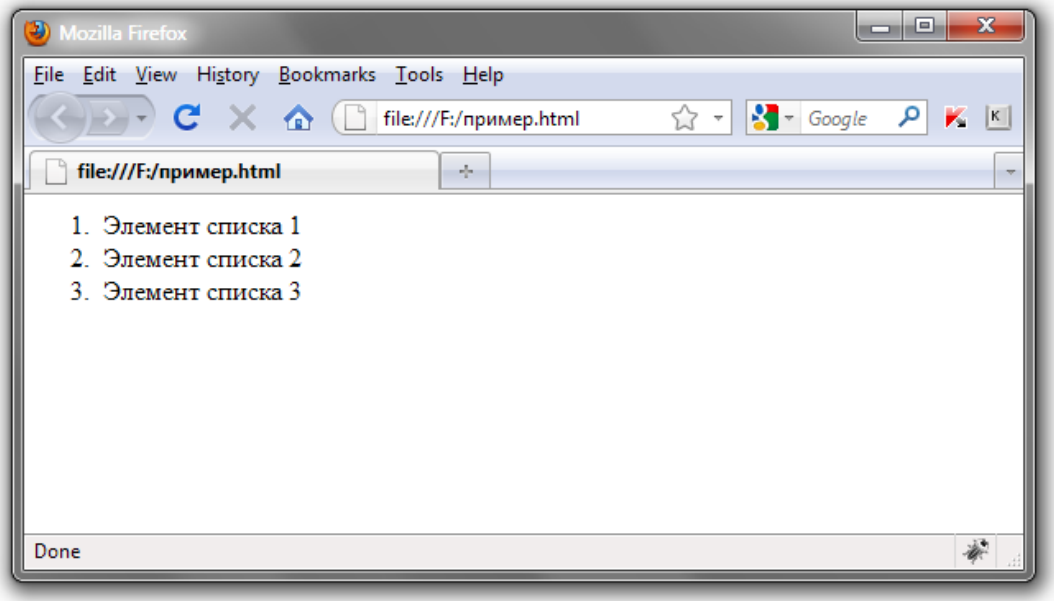
Данный тип списка используется для форматирования перечня действий или элементов, последовательность которых имеет значение. Например, пошаговое описание выполнения некоторого алгоритма, содержание статьи или книги (пронумерованный перечень глав).

Нумерованный список создается точно так же, как и маркированный, только вместо элемента `<ul>` используется элемент `<ol>` — Ordered List, упорядоченный список:

```
<ol>
    <li>Элемент списка 1</li>
    <li>Элемент списка 2</li>
    <li>Элемент списка 3</li>
</ol>
```

Тип нумерации по умолчанию — арабские цифры (1,2,3...). Изменить способ нумерации можно, используя устаревший атрибут type, который может принимать следующие значения:

Нумерованный список



- "1" — нумерация арабская: 1, 2, 3, ... 10, 11...
- "A" — нумерация латинским алфавитом (верхний регистр): A, B, C, ... AA, AB, ...
- "a" — нумерация латинским алфавитом (нижний регистр): a, b, c, ... aa, ab, ...
- "I" — нумерация римскими цифрами (верхний регистр): I, II, III, IV, ...
- "i" — нумерация римскими цифрами (нижний регистр): i, ii, iii, iv, ...

Список можно начать не с «1» или «a», а, например с «3» или «c». Для этого используется отмененный атрибут `start`. В качестве его значения используется число, задающее смещение номера первого элемента:

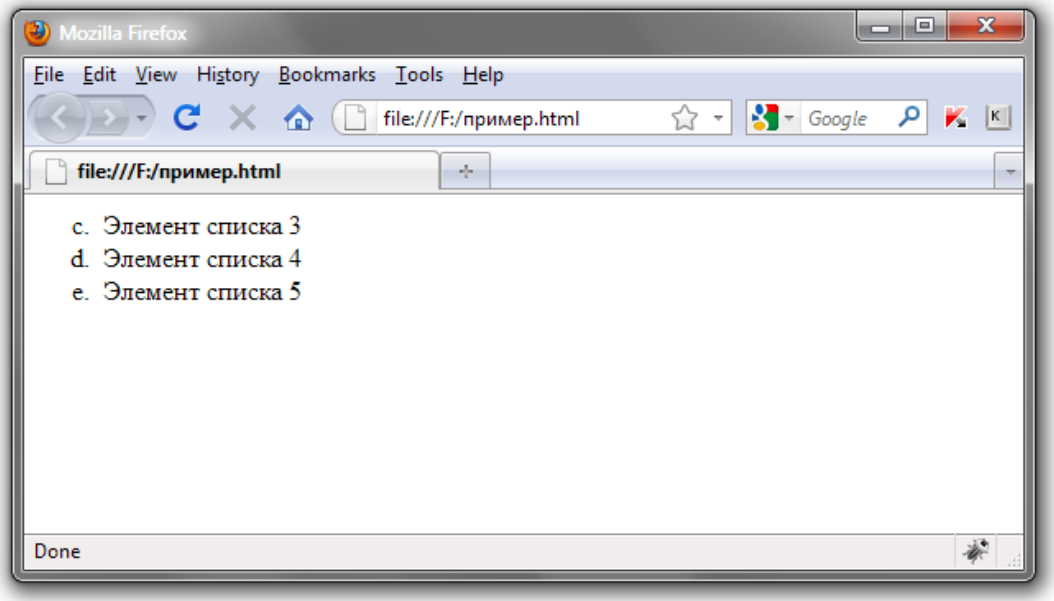
```
<ol start="3" type="a">
    <li>Элемент списка 3</li>
    <li>Элемент списка 4</li>
    <li>Элемент списка 5</li>
</ol>
```

В результате получим список, имеющий нумерацию в виде малых латинских букв, начинающийся с номера «c» и продолжающийся «d», «e» и т.д.

Кроме того, нумерацию списка можно разорвать, а затем продолжить с любого номера. В этом случае используется отмененный атрибут `value` элемента списка `li`. Атрибут `value` записывается в том элементе, с которого требуется продолжить нумерацию:

```
<ol type="I">
    <li>Элемент списка 1</li>
    <li>Элемент списка 2</li>
    <li>Элемент списка 3</li>
    <br /> ...
    <li value="10">Элемент списка 4</li>
    <li>Элемент списка 5</li>
</ol>
```

Нумерованный список с измененными типом нумерации и начальным номером



Это может быть полезным, если необходимо разместить на странице общеизвестный список, такой как список дней недели или месяцев. Такой список не имеет смысла приводить полностью, достаточно указать несколько первых и несколько последних элементов. Для смещения нумерации последних элементов и используют атрибут `value`.

В принципе, необходимость использования атрибутов `start` и `value` не так уж велика. Очевидно именно по этой причине их дальнейшая поддержка спецификацией прекращена, а соответствующей замены в CSS не существует.

## Стили для форматирования списков

Помимо стилевых правил для форматирования шрифта и цвета символов, рассмотренных на прошлом занятии, к списку и его отдельным элементам можно применить ряд специальных правил. Эти правила отвечают за назначение маркера или нумерации элементов списка, а, также, могут управлять положением маркера по отношению к элементу.

**Свойство list-style-type**

Это свойство является заменой устаревшего атрибута type, о котором упоминалось в предыдущих разделах. Свойство `list-style-type` может принимать следующие значения:

- `disc` — маркировка в виде диска (замена `type="disc"`);
- `circle` — маркировка в виде окружности (замена `type="circle"`);
- `square` — маркировка в виде квадрата (замена `type="square"`)
- `decimal` — арабская (десятичная) нумерация (замена `type="1"`)
- `upper-roman` — римская нумерация в верхнем регистре (замена `type="I"`)
- `lower-roman` — римская нумерация в нижнем регистре (замена `type="i"`)
- `upper-alpha` — алфавитная нумерация в верхнем регистре (замена `type="A"`)
- `lower-alpha` — алфавитная нумерация в нижнем регистре (замена `type="a"`)
- `none` — маркер или нумерация отсутствует.

Свойство стиля `list-style-type` можно применять как для форматирования маркированного, так и для нумерованного списка. В принципе, можно маркированному списку `<ul>` дать нумерацию элементов, и наоборот — нумерованному `<ol>` присвоить маркер в виде диска, к примеру. (Вопрос только в том, зачем? Сделав это вы грубо нарушите логику разметки.)

Помимо дублирования значений отмененного атрибута `type`, с помощью стилей можно вообще запретить всяческую нумерацию или маркировку. Для этого используется значение none свойства `list-style-type`.

Ниже показан пример применения стиля для оформления нумерации списка в виде малых букв латинского алфавита:

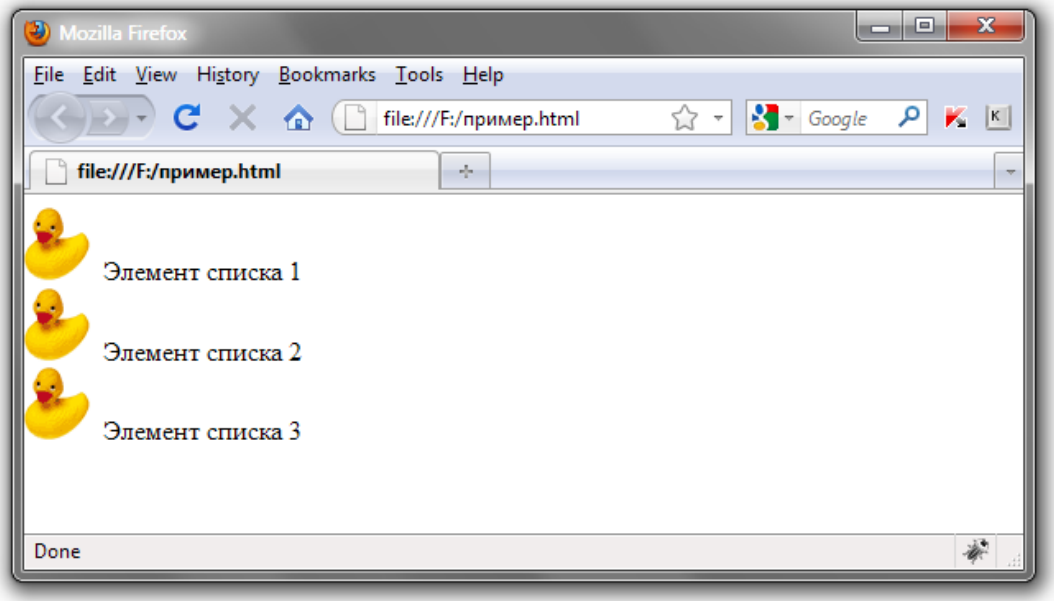
```
<ol style="list-style-type: lower-alpha;">
    <li>Элемент списка 1</li>
    <li>Элемент списка 2</li>
    <li>Элемент списка 3</li>
    <li>Элемент списка 4</li>
    <li>Элемент списка 5</li>
</ol>
```

**Свойство list-style-image**

Еще одним эффектным способом визуального оформления списков является возможность использования графического маркера элемента списка. То есть, вместо использования нескольких скучных стандартных маркеров, вы сможете использовать собственноручно созданное изображение.

Изображение должно быть создано в формате JPG, GIF или PNG. При использовании формата GIF или явным преимуществом является поддержка прозрачности — маркер будет более универсальным и может быть применен на web-страницах, имеющих совершенно разные фоновые цвета или изображения.

Список с графическими маркерами



Предположим, что вы хотите использовать в качестве маркера изображение *marker.gif*, расположенное в той же папке, что и XHTML страница. Код для достижения поставленной цели будет иметь следующий вид:

```
<ul style="list-style-image:
        url(marker.gif);">
    <li>Элемент списка 1</li>
    <li>Элемент списка 2</li>
    <li>Элемент списка 3</li>
</ul>
```

Если в стиле для списка одновременно указать и стандартный тип маркера и графический маркер, то сработает только графический, так как он имеет больший приоритет.

**Совет:** При назначении графического маркера, установите так же и стандартный маркер. Тогда, в случае отсутствия изображения маркера, форматирование списка будет производиться стандартным маркером — своеобразная «подстраховка» на случай, если изображение не сможет быть отображено.

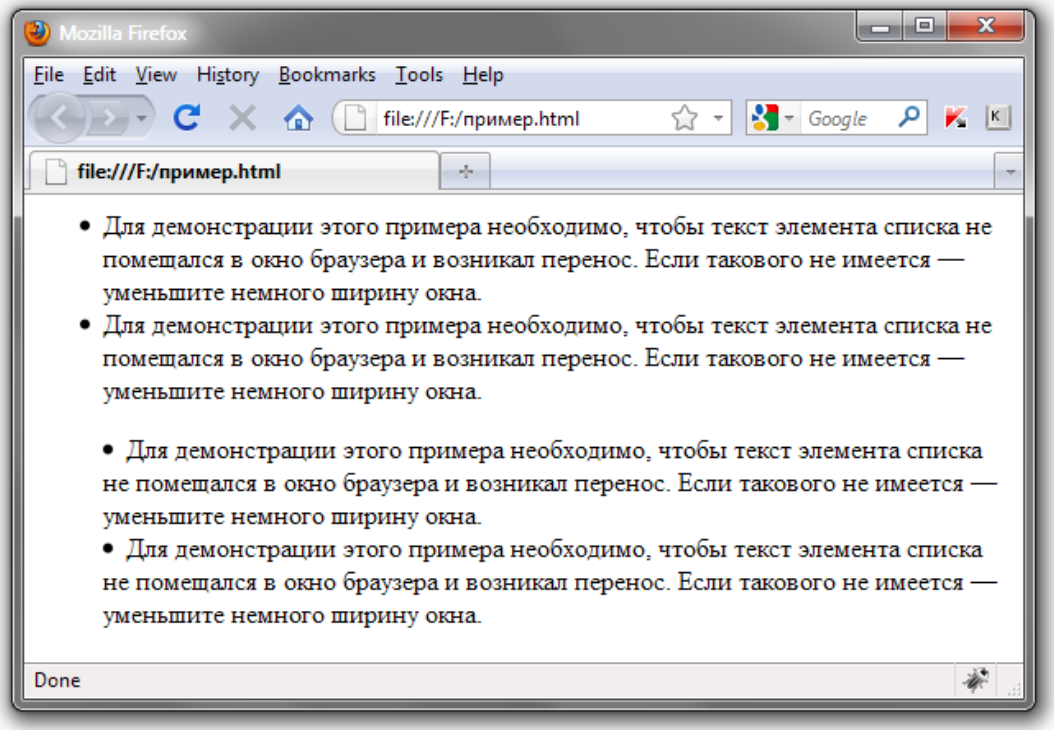
Для отмены графического маркера используйте свойство `list-style-image` со значением none:

```
...
<li style="list-style-image: none;">
...
```

## Свойство list-style-position

Свойство стиля `list-style-position` позволяет управлять положением маркера или номера

Маркированные списки с разными значениями свойства list-style-position



элемента по отношению к тексту элемента списка. Возможные значения:

- `outside` — маркеры расположенны снаружи по отношению к тексту (по умолчанию);
- `inside` — маркеры расположенны «в тексте».

Пример использования свойств `list-style-position` показано на иллюстрации, и дополнительно приведён пример кода, который вы можете скопировать в документ HTML и самостоятельно просмотреть в браузере.

**Резюме**

Использование стилей, хоть и выглядит на первых порах более громоздким, позволяет значительно расширить наши возможности по оформлению списков.

```
<ul style="list-style-position: outside;">
    <li>Для демонстрации этого примера необходимо, чтобы текст
        элемента списка не помещался в окно браузера и возникал
        перенос. Если такового не имеется — уменьшите немного
        ширину окна.</li>
    <li>Для демонстрации этого примера необходимо, чтобы текст
        элемента списка не помещался в окно браузера и возникал
        перенос. Если такового не имеется — уменьшите немного
        ширину окна.</li>
</ul>

<ul style="list-style-position: inside;">
    <li>Для демонстрации этого примера необходимо, чтобы текст
        элемента списка не помещался в окно браузера и возникал
        перенос. Если такового не имеется — уменьшите немного
        ширину окна.</li>
    <li>Для демонстрации этого примера необходимо, чтобы текст
        элемента списка не помещался в окно браузера и возникал
        перенос. Если такового не имеется — уменьшите немного
        ширину окна.</li>
</ul>
```

Пример списка, использующего свойство list-style-position

# Списки определений

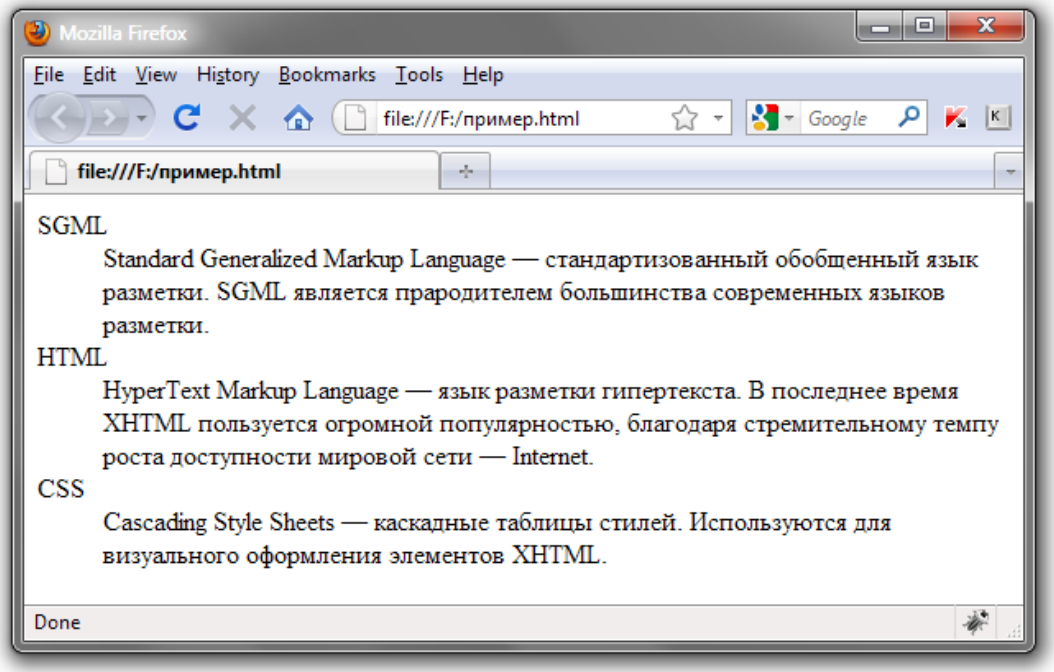
Списки определений представляет собой совершенно иной тип списков, нежели ранее изученные. У них нет маркеров, зато они включают два типа пунктов. Как правило, с их помощью оформляются словарные статьи, толкование терминов и т.д.

Списки определений создаются элементами трех типов:

- `<dl>` — Definition List, собственно сам список
- `<dt>` — Definition Term, определение, термин
- `<dd>` — Definition Description, пояснение, комментарий к термину

Никаких специальных атрибутов элементы <dl>, <dt> и <dd> не поддерживают.

```
<dl>
    <dt>SGML</dt>
    <dd>Standard Generalized Markup Language — стандартизованный
        обобщенный язык разметки. SGML является прародителем
        большинства современных языков разметки.</dd>
    <dt>HTML</dt>
    <dd>HyperText Markup Language — язык разметки гипертекста. В
        последнее время XHTML пользуется огромной популярностью,
        благодаря стремительному темпу роста доступности мировой
        сети — Internet.</dd>
    <dt>CSS</dt>
    <dd>Cascading Style Sheets — каскадные таблицы стилей.
        Используются для визуального оформления элементов
        XHTML.</dd>
</dl>
```



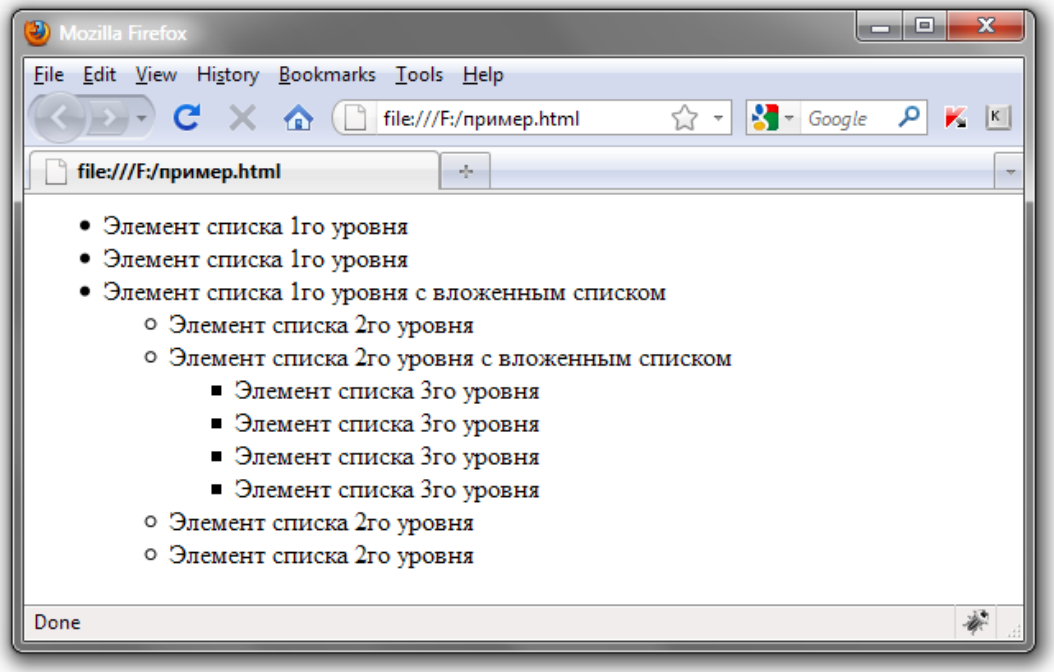
# Многоуровневые списки

Многоуровневые списки используют в тех случаях, когда требуется отразить иерархическую структуру, подчинение одних элементов другим. Наиболее ярким примером многоуровневого списка может служить оглавление сложного документа — например, оглавление этого же урока, которое вы можете увидеть на стр. 2. Заголовки разных уровней выделены отступами, и благодаря этому хорошо видна иерархия заголовков, и их взаимное подчинение.

Для создания многоуровневого списка просто вложите в любой из элементов списка еще один список. Это и будет список второго уровня. Для создания третьего уровня, вложите список в элемент второго уровня, и так далее.

Внимание! Вкладывать элемент `<ul>` или `<ol>` необходимо только в элемент `<li>`. Нельзя вкладывать список в список непосредственно!

```
<ul> <li>Элемент списка 1го уровня</li>
     <li>Элемент списка 1го уровня</li>
     <li>Элемент списка 1го уровня с вложенным списком
         <ul> <li>Элемент списка 2го уровня</li>
              <li>Элемент списка 2го уровня с вложенным списком
                  <ul> <li>Элемент списка 3го уровня</li>
                       <li>Элемент списка 3го уровня</li>
                       <li>Элемент списка 3го уровня</li>
                       <li>Элемент списка 3го уровня</li>
                  </ul></li>
              <li>Элемент списка 2го уровня</li>
              <li>Элемент списка 2го уровня</li>
          </ul></li>
</ul>
```

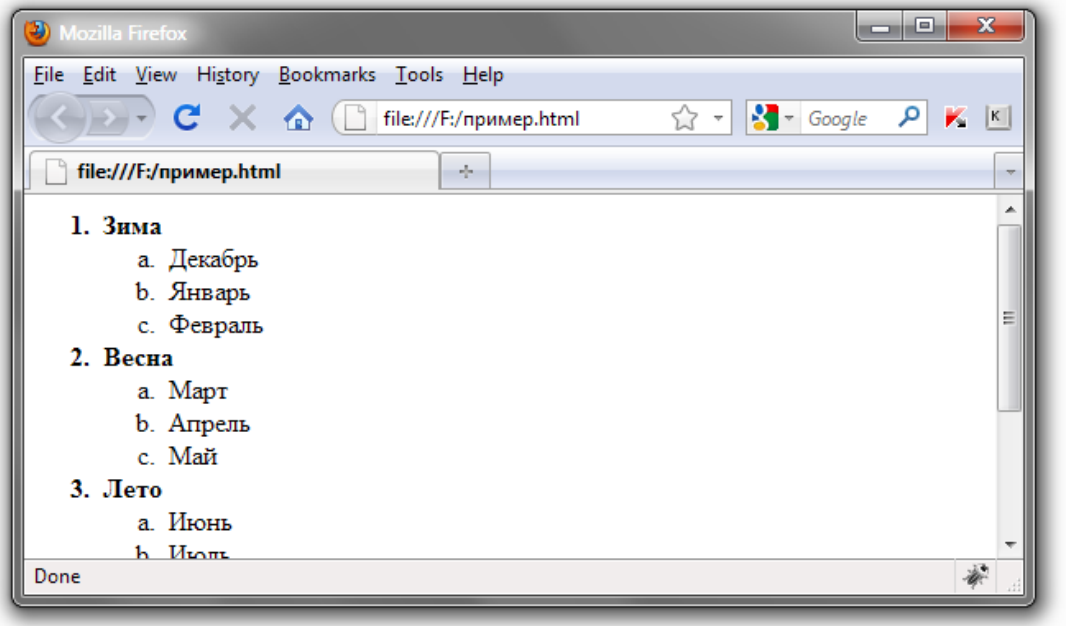


На иллюстрации, показывающей пример отображения браузером многоуровнего списка, можно видеть, что типы маркеров для каждого уровня списка изменяются. Первый уровень маркируется как "disc", второй — "circle", третий и далее — "square". Проделав тот же опыт с упорядоченными списками, вы убедитесь в том, что каждый следующий уровень списка не меняет тип нумерации. Эту задачу придется решать самостоятельно.

Одним из вариантов ее решения является использование стилей CSS для вложенных списков.

Для еще большего визуального отличия элементов первого и второго уровня, внешнему списку назначен полужирный шрифт (font-weight: bold;). В каждом вложенном списке жирность шрифта возвращается в норму (font-weight: normal;).

```
<ol style="font-weight: bold;">
    <li>Зима
        <ol style="list-style-type: lower-alpha;
                   font-weight: normal;">
            <li>Декабрь</li>
            <li>Январь</li>
            <li>Февраль</li>
        </ol></li>
    <li>Весна
        <ol style="list-style-type: lower-alpha;
                   font-weight: normal;">
            <li>Март</li>
            <li>Апрель</li>
            <li>Май</li>
        </ol></li>
        ...
```



## Кодировка документа

В общем случае кодировка (encoding), или кодовая таблица, — это однозначное соответствие между подмножеством целых чисел (как правило, идущих подряд) и некоторым набором символов. Ключевым здесь является понятие символа. Символ может быть буквой (а может и не быть), может соответствовать звуку речи (а может и не соответствовать) и может быть представлен графическим знаком (но может обходиться и без какого бы то ни было видимого образа). Символ — это атом смысла, мельчайшая неделимая частица информации.

Так, латинское «A» и кириллическое «А» — это разные символы, потому что они употребляются в разных контекстах и несут в себе разную информацию.

Определяющим для любой кодировки является количество охватываемых ею кодов и, соответственно, символов. Поскольку тексты в компьютере хранятся в виде последовательности байтов, большинство кодировок естественным образом распадаются на однобайтовые, или восьмибитные, способные закодировать не больше 256 символов, и двухбайтовые, или шестнадцатибитные, чья емкость может достигать 65636 знакомест.

## ASCII

Прежде чем переходить к восьмибитным кодировкам, нужно сказать несколько слов о кодировке под названием ASCII (American Standard Code for Information Interchange) — кодировке также восьмибитной, но охватывающей только 128 символов и потому довольствующейся семью значимыми битами (старший, восьмой бит при этом всегда равен нулю). Важность этой кодировки, включающей латинский алфавит, цифры и основные знаки пунктуации, необычайно велика: почти все остальные (большие по размеру) кодировки совместимы с ней, т. е. размещают на своих первых 128 знакоместах те же самые символы в том же порядке.

Первые 32 позиции в кодировке ASCII заняты так называемыми управляющими символами (control characters), предназначенными не для передачи собственно текстовой информации, а для управления устройством, читающий (или получающим по линии связи) текстовый файл. Лишь немногие из этих символов — возврат каретки, перевод строки, табуляция — до сих пор используются в более-менее общепринятых значениях; остальные, давно уже вышедшие из употребления, в былые времена выполняли для «голого» ASCII-текста те же функции, которые сейчас возложены на разнообразные форматы данных и протоколы связи.

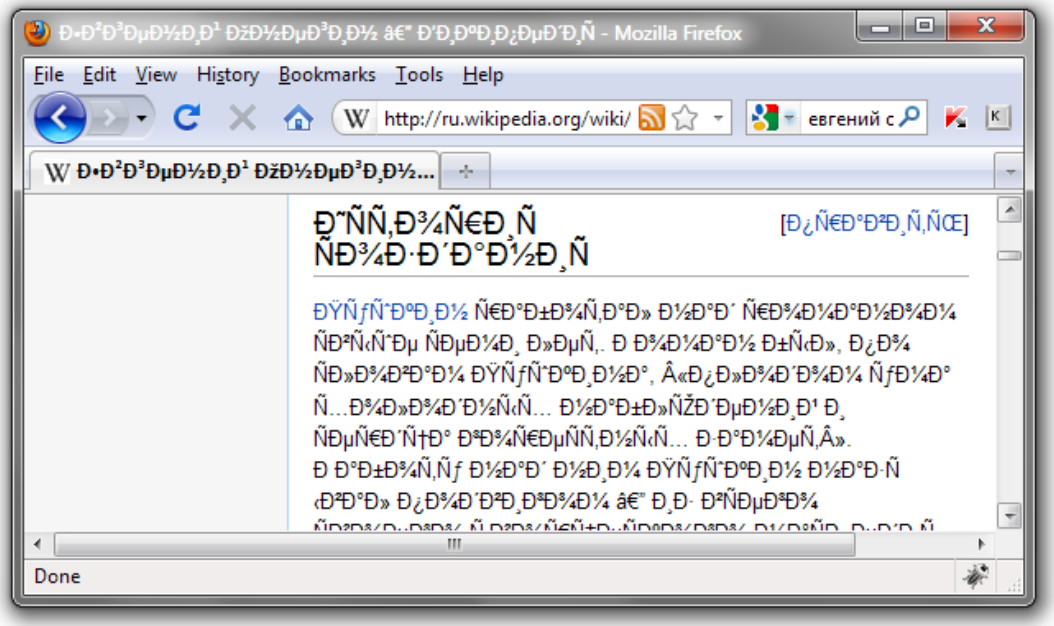
Задействовав в кодировке ASCII старший бит, мы получаем дополнительные 128 знакомест, которых должно хватить для кодирования, например, кириллического алфавита или набора каких-нибудь специальных символов. К сожалению, восьмибитных кодировок на свете существует гораздо больше, чем наборов символов, которые они кодируют. Очень характерна в этой связи ситуация с русским языком — анархия компьютеризации в нашей стране, наложившаяся на всемирную анархию конкурирующих компьютерных платформ и операционных систем, привела к тому, что для кириллицы существует сразу несколько однобайтовых кодовых таблиц.

## Однобайтовые кодировки

### КОИ8

Хронологически одним из первых стандартов кодирования русских букв на компьютерах был КОИ8 («Код обмена информацией, 8-битный»). Эта кодировка применялась еще

Так выглядит страница, написанная по-русски и отображённая в неправильной кодировке



в доисторические советские времена на компьютерах ЕС ЭВМ, и когда в середине 80-х появились первые русифицированные версии операционной системы UNIX, они унаследовали эту кодировку у своих «предков». Это одна из самых распространённых кириллических кодировок в Интернете.

**Windows-1251**

Вторая по значению в русском Интернете (и, безусловно, первая по употребимости на персональных компьютерах) кодировка — это стандартная кириллическая кодировка Microsoft Windows, обозначаемая аббревиатурой CP1251 («CP» расшифровывается как «Code Page», «кодовая страница»).

Все Windows-приложения, работающие с русским языком, обязаны понимать эту кодировку без перевода. Благодаря распространенности Windows кодировка CP1251, также является одной из самых распространённых в Интернете.

**Семейство 8859. Latin-1**

Похожая ситуация с конкурирующими платформами и операционными системами и, как следствие, с конкурирующими несовместимыми кодировками наблюдается и в других языках, пользующихся своим собственным алфавитом или даже латинским алфавитом с расширениями. Международная организация по стандартизации (International Standards Organization, ISO) попыталась навести порядок в восьмибитных кодировках, создав серию кодировок ISO 8859, расширяющих таблицу ASCII для латинских букв с диакритикой и лигатур (кодировка ISO 8859-1), кириллицы (ISO 8859-5), арабского (ISO 8859-6), греческого (ISO 8859-7), и других алфавитов.

Если кодировка ISO 8859-5 для кириллицы так и не прижилась, первая из этой серии — кодировка ISO 8859-1, известная также под именем Latin-1, — сумела стать общепринятым стандартом для кодирования «расширенной» латиницы. В эту кодировку включены почти все символы, употребляющиеся в письменностях западноевропейских языков — французского, немецкого, испанского и т.д.

## Двухбайтовые кодировки

Языки с иероглифической письменностью (японский, китайский, корейский) пользуются смешанными кодировками, в которых иероглифы (а их в сотни раз больше, чем букв в алфавите) представлены двухбайтовыми кодами, а вставки на латинице кодируются по однобайтовой таблице (обычно совпадающей с Latin-1). Переключение между двухбайтовым и однобайтовым режимами производится специально зарезервированными управляющими символами.

В 1991 году была предпринята попытка создать единую универсальную двухбайтовую кодировку, охватывающую все алфавиты и иероглифические системы мира. Результатом стал стандарт под названием Unicode, покрывающий не только системы письменности всех живых и большинства мертвых языков мира, но и множество музыкальных, математических, химических и прочих символов. Хотя массовое применение Unicode в документах и программах остается делом будущего, для web-дизайнера эта кодировка имеет особое значение, так как именно она объявлена «стандартной кодировкой документа» в HTML начиная с версии 4.

**ISO 10646 и UTF-8**

Предвидя неизбежное рано или поздно исчерпание и двухбайтового кодового пространства (пока еще до этого далеко, так как около 30% кодов в Unicode до сих пор не заняты), ISO уже застолбила стандарт четырехбайтовой, совместимой с Unicode кодировки под названием ISO 10646. Пока что вместо этого обозначения, которое то и дело попадается в стандартах, вы можете с чистой совестью подставлять «Unicode», так как никаких новых символов, выходящих за границы совпадающих с Unicode первых 65536 знакомест, в ISO 10646 еще не определено.

По-видимому, в ближайшее время все более важную роль будет играть особый формат Unicode (и ISO 10646) под названием UTF-8. Эта «производная» кодировка пользуется для записи символов цепочками байтов различной длины (от одного до шести), которые с помощью несложного алгоритма преобразуются в Unicode-коды, причем более употребительным символам соответствуют более короткие це-

почки. Главное достоинство этого формата — совместимость с ASCII не только по значениям кодов, но и по количеству бит на символ, так как для кодирования любого из первых 128 символов в UTF-8 достаточно одного байта (хотя, например, для букв кириллицы нужно уже по два байта).

## Указание кодировки документа

Для указания кодировки символов web-страницы используются следующие обозначения кодовых таблиц:

- `windows-1251` — кириллица Windows;
- `KOI8-R` — кодировка КОИ8 для русского языка;
- `KOI8-U` — кодировка КОИ8 для украинского языка;
- `ISO 8859-1` — кодировка Latin-1;
- `ISO 8859-5` — кодировка семейства ISO 8859 для символов кирилицы;
- `UTF-8` — кодировка UTF-8;
- `Unicode` — кодировка Unicode.

На web-странице указать кодировку документа можно двумя способами:

XML декларация:

```
<?xml version="1.0"
        encoding="windows-1251"?>
```

Элемент <meta>:

```
<meta http-equiv="content-type"
        content="text/html;
        charset=windows-1251" />
```

Элемент `<meta>` является дочерним по отношению к разделу заголовка документа (`<head>`) и служит для указания типа и кодировки содержимого страницы. Типом содержимого является структурированный текст в формате html (`text/html`), используемая кодировка кириллица windows (`charset=windows-1251`).

Обычно используют оба способа одновременно. Например, для указания кодировки КОИ8 для украинского языка на web-странице, используют следующую структуру документа:

```
<?xml version="1.0" encoding="KOI8-U"?>
<!DOCTYPE html PUBLIC ... >
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Моя перша сторінка</title>
        <meta http-equiv="Content-Type"
              content="text/html;
        charset=KOI8-U" />
    </head>
    ...
</html>
```

Будьте внимательны! При сохранении текста выбирайте ту же кодировку, что указали на web-странице. В противном случае ваша страница не будет отображаться правильно в большинстве браузеров.

К слову, браузер можно "вручную" переключить на использование требуемой кодировки. В браузере Mozilla Firfox для этого следует воспользоваться меню Вид > Кодировка символов (View > Charater Enconing) и выберите требуемую кодировку.

Поэкспериментируйте с различными кодировками, и вы убедитесь, что символы литинского алфавита, цифры и знаки пунктуации передаются без изменений в подавляющем большинстве из них.

## Символьные подстановки

Как бы ни был широк выбор кодировок и разнообразен набор символов, существует следующее ограничение: не все символы можно набрать на клавиатуре! Например, символ зарегистрированного торгового знака — ®, или торговой марки — ™.

Несомненно, можно воспользоваться утилитой вставки символа, стандартной для Windows. Однако не все редакторы в состоянии такой символ отобразить. Кроме того, при попытке вставить символы из других языков, вы можете выйти за «рамки», определённые выбранной кодировкой (если, конечно, вы не используете UTF-8).

Как поступать в таких случаях? Воспользоваться символьными подстановками.

Символьные подстановки представляют собой специальную последовательность, преобразуемую браузерами в заданный символ. Подстановку можно осуществить несколькими способами:

- `&мнемокод;` — вставка символа по его "мнемокоду" (имени).
- `&#КОД10;` — вставка символа по его десятичному коду.
- `&#xКОД16;` — вставка символа по его шестнадцатиричному коду.

Все три способа равнозначны «в глазах» браузера, поэтому символ ® можно отобразить тремя способами:

- ®
- ®
- ®

Символьные подстановки применяются не только для вставки символов, которых нет на клавиатуре, но, так же для вставки символов, имеющих специальное назначение в XHTML. Существуют некоторые символы, которые

---

Тип подстановки, использующий буквенные аббревиатуры вместо номеров символов, называется *мнемоническим*, поскольку в большинстве случаев его проще запомнить, чем номер символа.

*нельзя* напрямую использовать в коде XHTML, и необходимо заменять их символьными подстановками.

К примеру, вы никогда не задумывались, как вывести на страницу текст, содержащий XHTML тэги? Например такую фразу:

```
<p>Для вывода текста полужирным
        начертанием, выделите его тэгами
        <strong> и </strong>.</p>
```

На приведённой иллюстрации видно, какой неприятный казус приключился при отображении такой страницы: надписи  и  исчезли, а союз «и» оказался выделенным полужирным начертанием.

Символы < и > играют особую роль в XHTML — начинают и завершают тэги. Поэтому непостедственная вставка их в код страницы приведет к интерпретации `<strong>` и `</strong>` как тэгов и не даст желаемого результата. Итак, вывод: необходимо использовать символьные подстановки.

Ещё один символ, который в некоторых случаях нельзя использовать в коде — это символ &, так как он тоже является управляющим (он начинает символьную подстановку). Если на странице необходимо привести код подстановки, необходимо заменить символ & подстановкой `&amp;`.

Наиболее употребимые символьные подстановки

| Символ | Мнемонический код | Десятичный код | Шестнадцатиричный код | Пояснение |
|---|---|---|---|---|
| < | &lt; | &#0060; | &#x003c; | |
| > | &gt; | &#0062; | &#x003e; | |
| & | &amp; | &#0038; | &#x0026; | |
| | | | | |
| |   |   | &#00a0; | Символ «неразрывного пробела». Слова, разделяемые им, не переносятся на разные строки отдельно друг от друга. |
| | &shy; | &#0173; | &#00ad; | Символ «мягкого переноса». Части слова, соединяемые им, переносятся, при необходимости. На месте переноса отображается дефис. |
| | | | | |
| " | &quot; | &#0034; | &#x0022; | Прямая кавычка |
| « | &laquo; | &#0171; | &#x00ab; | Открывающая и закрывающая кавычки-«елочки» |
| » | &raquo; | &#0187; | &#x00bb; | |
| “ | &ldquo; | &#0147; | &#x0093; | Открывающая и закрывающая кавычки-«лапки» |
| ” | &rdquo; | &#0148; | &#x0094; | |
| | | | | |
| – | &ndash; | &#0150; | &#x0096; | Среднее тире |
| — | &mdash; | &#0151; | &#x0097; | Длинное тире |
| | | | | |
| © | &copy; | &#169; | &#A9; | Знак охраны авторских прав |

## Неразрывный пробел

Символ ` ` создает *неразрывный пробел*, и в отличие от обычного пробела, несколько символов ` `, записанных подряд, не сокращаются браузером. Кроме того, если несколько слов соединяются неразрывным пробелом, то при достижении границы окна браузера, все эти слова вместе будут перенесены на следующую строку.

Такой подход используется, например, при записи фамилии и инициалов, или наименования организации:

```
...
<p>Символ "неразрывный пробел" используется
        в тех случаях, когда необходимо
        предотвратить нежелательный
        перенос слов при записи названий
        организаций или инициалов.
        Например:
        <b>WWW Consortium</b>
        или
        <b>A. H. Артемов</b>.
        Чтобы увидеть результат, изменяйте
        ширину окна и следите за поведением
        выделенных фрагментов.</p>
...
```

## Мягкий перенос

В противоположность неразрывному пробелу, символ мягкого переноса —  служит для создания переноса в том месте, где это может потребоваться. При этом, на месте переноса появляется символ -. Если перенос не требуется, символ `&shy;` не визуализируется.

Применяется мягкий перенос преимущественно в длинных словах, для более рамномерного распределения текста на странице:

```
...
<p>В 1991 году была предпри&shy;нята
        попытка создать единую
        универ&shy;сальную
        двухбай&shy;товую коди&shy;ровку,
        охваты&shy;вающую все алфавиты и
        иерогли&shy;фичес&shy;кие системы
        мира. Результа&shy;том стал
        стандарт ...
</p>
```

Пример содержит 2 абзаца. Один из них использует мягкий перенос, другой — нет. Изменяйте размер окна и следите за переносом слов обоих абзацев.

# Домашнее задание

В качестве домашнего задания предлагаю разметить фрагмент текста, взятый из спецификации XHTML 1.1.

Исходный текст, который необходимо разметить, прилагается к уроку и носит название *homework.txt*

# XHTML™ 1.1 - Модульный язык XHTML

## Рекомендация W3C от 31 мая 2001 г.

**Настоящая версия:**
http://www.w3.org/TR/2001/WD-xhtml11-20010531
(В одном файле, Postscript-версия, версия в формате PDF, архив ZIP или архив TAR Gzip)
**Последняя версия:**
http://www.w3.org/TR/xhtml11
**Предыдущая версия:**
http://www.w3.org/TR/2001/PR-xhtml11-20010406
**Редакторы:**
Мюррей Альтхайм (Murray Altheim), Sun Microsystems
Шейн МакКаррон (Shane McCarron), Applied Testing and Technology



## Краткое содержание



## Подробное содержание


СОЗДАНИЕ СТАТИЧНЫХ ВЕБ-СТРАНИЦ
С ПОМОЩЬЮ ЯЗЫКОВ XHTML И CSS.

# СПИСКИ. КОДИРОВКИ. СИМВОЛЫ

Верстка: Е. Радченко