

1.Теги HTML

1.1.Общий синтаксис написания тегов

Чтобы браузер при отображении документа понимал, что имеет дело не с простым текстом, а с элементом форматирования и применяются теги. Общий синтаксис написания тегов следующий

```
<тег    атрибут1="значение"    атрибут2="значение">    <тег    атрибут1="значение"
атрибут2="значение">...</тег>
```

Как видно из данного примера, теги бывают двух типов — одиночные и парные (контейнеры). Одиночный тег используется самостоятельно, а парный может включать внутри себя другие теги или текст. У тегов допустимы различные атрибуты, которые разделяются между собой пробелом. Впрочем, есть теги без всяких дополнительных атрибутов. Условно атрибуты можно подразделить на обязательные, они непременно должны присутствовать, и необязательные, их добавление зависит от цели применения тега.

В примере 1.1. показан типичный HTML-документ с тегами и текстом.

Пример 1.1. Теги в документе

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">  <title>Lorem
ipsum</title>
</head>
<body>
  <p>Lorem ipsum dolor sit amet consectetur cursus pede pellentesque vitae    pretium.
Tristique mus at elit lobortis libero Sed vestibulum ut eleifend habitasse.    Quis Nam Mauris
adipiscing Integer ligula dictum sed at enim urna. Et scelerisque    id et nibh dui tincidunt
Curabitur faucibus elit massa. Tincidunt et gravida    Phasellus eget parturient faucibus tellus at
justo sollicitudin. Mi nulla ut    adipiscing.</p>
</body>
</html>
```

В данном примере используется одиночный тег **<meta>**, а парных тегов сразу несколько: **<html>**, **<head>**, **<title>**, **<body>** и **<p>**.

1.2.Парные теги

Парные теги, называемые по-другому контейнеры, состоят из двух частей — открывающий и закрывающий тег. Открывающий тег обозначается как и одиночный — **<тег>**, а в закрывающем используется слэш — **</тег>**. Допускается вкладывать в контейнер другие теги, однако следует соблюдать их порядок. Так, на рис. 1.1 демонстрируется, как можно и нельзя добавлять один контейнер внутри другого.

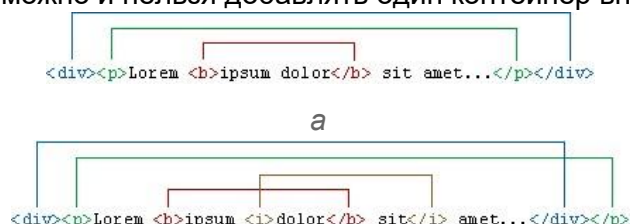


Рис. 1.1. Вложение тегов, а — правильное, б — неверное

Если связать открывающий и закрывающий тег между собой скобкой, как показано на рис. 1.1, то несколько скобок обозначающих разные контейнеры, не должны пересекаться между собой (рис. 1.1а). Любое пересечение условных скобок (рис. 1.1б) говорит о том, что правильная последовательность тегов нарушена.



Не все контейнеры требуют обязательно закрывающего тега, иногда его можно и опустить. Тем не менее, закрывайте все требуемые теги, так вы приучитесь сводить к нулю возможные ошибки.

1.3. Правила применения тегов

Для тегов любого типа действуют определенные правила их использования. Причем, некоторые правила обязательны для выполнения, а другие являются рекомендациями, т.е. их можно выполнять, а можно и нет.

Атрибуты тегов и кавычки

Согласно спецификации HTML все значения атрибутов тегов следует указывать в двойных ("пример") или одинарных кавычках ('пример'). Отсутствие кавычек не приведет к ошибкам, браузеры во многих случаях достаточно корректно обрабатывают код и без кавычек, за исключением текста, содержащего пробелы (пример 1.2).

Пример 1.2. Использование кавычек в атрибутах тегов

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8"> <title>Кавычки в
атрибуте alt</title>
</head>
<body>
  <p></p>
  <p></p> </body>
</html>
```

В данном примере строка 8 написана правильно, со всеми кавычками, а в строке 9 у атрибута **alt** кавычки отсутствуют. Из-за этого браузер в качестве значения **alt** возьмет только первое слово («Вид»), а слово «заголовка» будет воспринято как ошибочное значение. Поэтому всегда приучайтесь указывать атрибуты тегов в кавычках.

Теги можно писать как прописными, так и строчными символами

Любые теги, а также их атрибуты нечувствительны к регистру, поэтому форму записи вы вольны выбирать сами, как писать — **
**, **
** или **
**. В любом случае рекомендуется придерживаться выбранной формы записи на протяжении всех страниц сайта. Заметим также, что текст, полностью набранный прописными символами, читается хуже, чем текст со строчными символами или смешанный.

Переносы строк

Внутри тега между его атрибутами допустимо ставить перенос строк. В примере 1.3 показана одна и та же строка, но оформленная разными способами.

Пример 1.3. Переносы строк в коде тега

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"> <title>Кавычки в
атрибуте alt</title>
</head>
<body>
<p></p>
<p></p>
</body> </html>
```

В данном примере первый тег `` набран в одну строку, включая все его атрибуты, а второй тег `` разбит на несколько строк.



Хотя ошибки при переносе текста в подобном случае и не возникнет, рекомендуем писать теги в одну строку, иначе ухудшается восприятие кода и его становится сложнее править.

Неизвестные теги и атрибуты

Если какой-либо тег или его атрибут был написан неверно, то браузер проигнорирует подобный тег и будет отображать текст так, словно тега и не было. Опять же, следует избегать неизвестных тегов, поскольку код HTML не пройдет валидацию.

Порядок тегов

Существует определенная иерархия вложенности тегов. Например, тег `<title>` должен находиться внутри контейнера `<head>` и нигде иначе. Чтобы не возникло ошибки, следите за тем, чтобы теги располагались в коде правильно.

Если теги между собой равноценны в иерархии связи, то их последовательность не имеет значения. Так, можно поменять местами теги `<title>` и `<meta>`, на конечном результате это никак не скажется.

Закрывайте все теги

Существует три состояния закрывающего тега: обязателен, не требуется или не обязателен. Обязательный закрывающий тег должен присутствовать всегда, иначе это приведет к ошибке при отображении документа. Для некоторых тегов вроде `
` закрывающего тега нет в принципе. Необязательный закрывающий тег говорит о том, что разработчик может его как добавить, так и опустить, к ошибке это не приведет. Однако рекомендуем закрывать все подобные теги, включая необязательные, это дисциплинирует, создает более стройный и строгий код, который легко модифицировать.

Атрибуты тегов

Чтобы расширить возможности отдельных тегов и более гибко управлять содержимым контейнеров и применяются атрибуты тегов.

Для атрибутов тегов используются значения по умолчанию

Когда для тега не добавлен какой-либо допустимый атрибут, это означает, что браузер в этом случае будет подставлять значение, заданное по умолчанию. Если вы ожидали получить иной результат на веб-странице, проверьте, возможно, следует явно указать значения некоторых атрибутов.

Атрибуты без значений

Допустимо использовать некоторые атрибуты у тегов, не присваивая им никакого значения, как показано в примере 1.4.

Пример 1.4. Атрибуты без значений

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <title>Добавление
формы</title>
</head> <body>
  <form action="self.php">
    <p><input type="text"></p>
    <p><input type="submit" disabled></p>
  </form>
</body>
</html>
```

В данном примере используется атрибут **disabled**, у которого явно не задано значение. Подобная запись называется «сокращенный атрибут тега».

Порядок атрибутов в тегах

Порядок атрибутов в любом теге не имеет значения и на результат отображения элемента не влияет. Поэтому теги вида **** и **** по своему действию равны.

Формат атрибутов

Каждый атрибут тега относится к определенному типу (например: текст, число, путь к файлу и др.), который обязательно должен учитываться при написании атрибута. Так, в примере 1.4 упоминается тег ****, он добавляет на веб-страницу рисунок, а его атрибут **width** задает ширину изображения в пикселах. Если поставить не число, а нечто другое, то значение будет проигнорировано и возникнет ошибка при валидации документа.

Структура HTML-кода

Если открыть любую веб-страницу, то она будет содержать в себе типичные элементы, которые не меняются от вида и направленности сайта. В примере 4.1 показан код простого документа, содержащего основные теги.

Пример 1.5. Исходный код веб-страницы

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <title>Пример веб-
страницы</title>
</head>
<body>
  <h1>Заголовок</h1>
  <!-- Комментарий -->
```

```
<p>Первый абзац.</p> <p>Второй абзац.</p>
</body>
</html>
```

Скопируйте содержимое данного примера и сохраните его в папке c:\www\ под именем example41.html. После этого запустите браузер и откройте файл через пункт меню **Файл > Открыть файл (Ctrl+O)**. В диалоговом окне выбора документа укажите файл example41.html. В браузере откроется веб-страница, показанная на рис. 1.2 .



Рис. 1.2. Результат выполнения примера

Далее разберем отдельные строки нашего кода.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

Элемент **<!DOCTYPE>** предназначен для указания типа текущего документа — DTD (document type definition, описание типа документа). Это необходимо, чтобы браузер понимал, как следует интерпретировать текущую веб-страницу, ведь HTML существует в нескольких версиях, кроме того, имеется XHTML (EXtensible HyperText Markup Language, расширенный язык разметки гипертекста), похожий на HTML, но различающийся с ним по синтаксису. Чтобы браузер «не путался» и понимал, согласно какому стандарту отображать веб-страницу и необходимо в первой строке кода задавать **<!DOCTYPE>**.

Существует несколько видов **<!DOCTYPE>**, они различаются в зависимости от версии HTML, на которую ориентированы. В табл. 1.1. приведены основные типы документов с их описанием.

Табл. 1.1. Допустимые DTD

DOCTYPE	Описание
HTML 4.01	
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">	Строгий синтаксис HTML.
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">	Переходный синтаксис HTML.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">	В HTML-документе применяются фреймы.
HTML 5	
<!DOCTYPE html>	В этой версии HTML только один доктайп.
XHTML 1.0	
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1strict.dtd">	Строгий синтаксис XHTML.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1transitional.dtd">	Переходный синтаксис XHTML.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1frameset.dtd">	Документ написан на XHTML и содержит фреймы.
XHTML 1.1	
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">	Разработчики XHTML 1.1 предполагают, что он постепенно вытеснит HTML. Как видите, никакого деления на виды это определение не имеет, поскольку синтаксис один и подчиняется четким правилам.

Разница между строгим и переходным описанием документа состоит в различном подходе к написанию кода документа.

Строгий HTML требует жесткого соблюдения спецификации HTML и не прощает ошибок. Переходный HTML более «спокойно» относится к некоторым огрехам кода, поэтому этот тип в определенных случаях использовать предпочтительнее.

Например, в строгом HTML и XHTML непременно требуется наличие тега **<title>**, а в переходном HTML его можно опустить и не указывать. При этом помним, что браузер в любом случае покажет документ, независимо от того, соответствует он синтаксису или нет. Подобная проверка осуществляется при помощи валидатора и предназначена в первую очередь для разработчиков, чтобы отслеживать ошибки в документе.

В дальнейшем будем применять преимущественно строгий **<!DOCTYPE>**, кроме случаев, когда это оговаривается особо. Это позволит нам избегать типичных ошибок и приучит к написанию синтаксически правильного кода.

Часто можно встретить код HTML вообще без использования **<!DOCTYPE>**, веб-страница в подобном случае все равно будет показана. Тем не менее, может получиться, что один и тот же документ отображается в браузере поразному при использовании **<!DOCTYPE>** и без него. Кроме того, браузеры могут по-своему показывать такие документы, в итоге страница «рассыплется», т.е. будет отображаться совсем не так, как это требуется разработчику. Чтобы не произошло подобных ситуаций, всегда добавляйте **<!DOCTYPE>** в начало документа.

```
<html>
```

Тег **<html>** определяет начало HTML-файла, внутри него хранится заголовок (**<head>**) и тело документа (**<body>**).

```
<head>
```

Заголовок документа, как еще называют блок **<head>**, может содержать текст и теги, но содержимое этого раздела не показывается напрямую на странице, за исключением контейнера **<title>**.

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Тег **<meta>** является универсальным и добавляет целый класс возможностей, в частности, с помощью метатегов, как обобщенно называют этот тег, можно изменять кодировку страницы, добавлять ключевые слова, описание документа и многое другое. Чтобы браузер понимал, что имеет дело с кодировкой UTF-8 (Unicode transformation format, формат преобразования Юникод) и добавляется данная строка.

```
<title>Пример веб-страницы</title>
```

Тег **<title>** определяет заголовок веб-страницы, это один из важных элементов предназначенный для решения множества задач. В операционной системе Windows текст заголовка отображается в левом верхнем углу окна браузера (рис. 1.3).

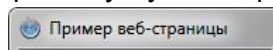


Рис. 1.3. Вид заголовка в браузере

⚠ Тег **<title>** является обязательным и должен непременно присутствовать в коде документа.

```
</head>
```

Обязательно следует добавлять закрывающий тег **</head>**, чтобы показать, что блок заголовка документа завершен.

```
<body>
```

Тело документа **<body>** предназначено для размещения тегов и содержательной части веб-страницы.

```
<h1>Заголовок</h1>
```

HTML предлагает шесть текстовых заголовков разного уровня, которые показывают относительную важность секции, расположенной после заголовка. Так, тег **<h1>** представляет собой наиболее важный заголовок первого уровня, а тег **<h6>** служит для обозначения заголовка шестого уровня и является наименее значительным. По умолчанию, заголовок первого уровня отображается самым крупным шрифтом жирного начертания, заголовки последующего уровня по размеру меньше. Теги **<h1>...<h6>** относятся к блочным элементам, они всегда начинаются с новой строки, а после них другие элементы отображаются на следующей строке. Кроме того, перед заголовком и после него добавляется пустое пространство.

```
<!-- Комментарий -->
```

Некоторый текст можно спрятать от показа в браузере, сделав его комментарием. Хотя такой текст пользователь не увидит, он все равно будет передаваться в документе, так что, посмотрев исходный код, можно обнаружить скрытые заметки.

Комментарии нужны для внесения в код своих записей, не влияющих на вид страницы. Начинаются они тегом `<!--` и заканчиваются тегом `-->`. Все, что находится между этими тегами отображаться на веб-странице не будет.

```
<p>Первый абзац.</p>
```

Тег `<p>` определяет абзац (параграф) текста. Если закрывающего тега нет, считается, что конец абзаца совпадает с началом следующего блочного элемента.

```
<p>Второй абзац.</p>
```

Тег `<p>` является блочным элементом, поэтому текст всегда начинается с новой строки, абзацы идущие друг за другом разделяются между собой отбивкой (так называется пустое пространство между ними).

```
</body>
```

Следует добавить закрывающий тег `</body>`, чтобы показать, что тело документа завершено.

```
</html>
```

Последним элементом в коде всегда идет закрывающий тег `</html>`.

Типы тегов

Каждый тег HTML принадлежит к определенной группе тегов, например, табличные теги направлены на формирование таблиц и не могут применяться для других целей.

Условно теги делятся на следующие типы:

- теги верхнего уровня;
- заголовка документа;
- блочные элементы;
- встроенные элементы;
- универсальные элементы;
- списки;
- таблицы.

Следует учитывать, что один и тот же тег может одновременно принадлежать разным группам, например, теги `` и `` относятся к категории списков, но также являются и блочными элементами.

Далее рассмотрим только те теги, которые потребуются нам в дальнейшей работе.

Теги верхнего уровня

Эти теги предназначены для формирования структуры веб-страницы и определяют раздел заголовка и тела документа.

`<html>`

Тег `<html>` является контейнером, который заключает в себе всё содержимое веб-страницы, включая теги `<head>` и `<body>`. Открывающий и закрывающий теги `<html>` в документе необязательны, но хороший стиль диктует непременно их использование.

`<head>`

Тег `<head>` предназначен для хранения других элементов, цель которых — помочь браузеру в работе с данными. Также внутри контейнера `<head>` находятся метатеги, которые используются для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных.

`<body>`

Тег `<body>` предназначен для хранения содержания веб-страницы, отображаемого в окне браузера. Информацию, которую следует выводить в документе, следует располагать именно внутри контейнера `<body>`. К такой информации относится текст, изображения, таблицы, списки и др.

Набор тегов верхнего уровня и порядок их вложения показан в примере 1.6.

Пример 1.6. Теги верхнего уровня

```
<html>
<head>  ...
</head>
<body>  ...
</body> </html>
```

В данном примере показано, что контейнер `<html>` определяет «каркас» всей веб-страницы, внутри него вначале задается тег `<head>`, затем идет контейнер `<body>`, в нем хранится содержательная часть документа, которая и отображается в браузере. Теги `<html>` и `<body>` хотя и не относятся к обязательным тегам (т. е. их можно не размещать в коде), все же стоит добавлять всегда. Это позволяет получить четкую и понятную структуру документа.

Заметьте, что в примере не упоминается `<!DOCTYPE>`, поскольку этот обязательный элемент кода веб-страницы не является тегом, а предназначен для браузеров, чтобы сообщить им, как интерпретировать текущий документ.

Теги заголовка документа

К этим тегам относятся элементы, которые располагаются в контейнере `<head>`. Все эти теги напрямую не отображаются в окне браузера, за исключением тега `<title>`, который определяет название веб-страницы.

`<title>`

Используется для отображения строки текста в левом верхнем углу окна браузера. Такая строка сообщает пользователю название сайта и другую информацию, которую добавляет разработчик.

`<meta>`

Метатеги используются для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных. Хотя тег `<meta>` всего один, он имеет несколько атрибутов, поэтому к нему и применяется множественное число.

Так, для краткого описания содержимого веб-страницы используется значение `description` атрибута `name`, как показано в примере 1.7.

Пример 1.7. Использование description

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html>
<head>
<title>HTML</title>
<meta name="description" content="Сайт об HTML и создании сайтов">
<meta http-equiv="content-type" content="text/html; charset=utf-8"> </head>
<body>
<p>...</p>
</body>
</html>
```

Описание сайта, заданное с помощью тега **<meta>** и значения **description**, обычно отображается в поисковых системах или каталогах при выводе результатов поиска. Значение **keywords** также предназначено в первую очередь для повышения рейтинга сайта в поисковых системах, в нем перечисляются ключевые слова, встречаемые на веб-странице (пример 1.8).

Пример 1.8. Использование keywords

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html>
<head>
<title>HTML</title>
<meta name="keywords" content="HTML, META, метатег, тег, поисковая система">
<meta http-equiv="content-type" content="text/html; charset=utf-8"> </head>
<body>
<p>...</p>
</body>
</html>
```

Ключевые слова можно перечислять через пробел или запятую. Поисковые системы сами приведут запись к виду, который они используют.

Практическая работа №1.

ЧАСТЬ 1

Основы HTML.

Задание № 1. Создание простейшего файла HTML

1. Создайте личную папку, куда вы будете сохранять все файлы своего сайта.
2. Запустите программу Блокнот (Notepad).
3. Наберите в окне программы простейший файл HTML.

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY>
    Расписание занятий на вторник
  </BODY>
</HTML>
```

4. Сохраните файл под именем RASP.HTML (обязательно укажите тип файла HTML при сохранении) в личной папке.
5. Для просмотра Web-страницы используйте любую программу браузера (Internet Explorer, Opera, Mozilla Firefox или другую). Для этого, не покидая программу Блокнот (сверните окно на панель задач), откройте личную папку и двойным кликом по файлу RASP.HTML откройте окно браузера.

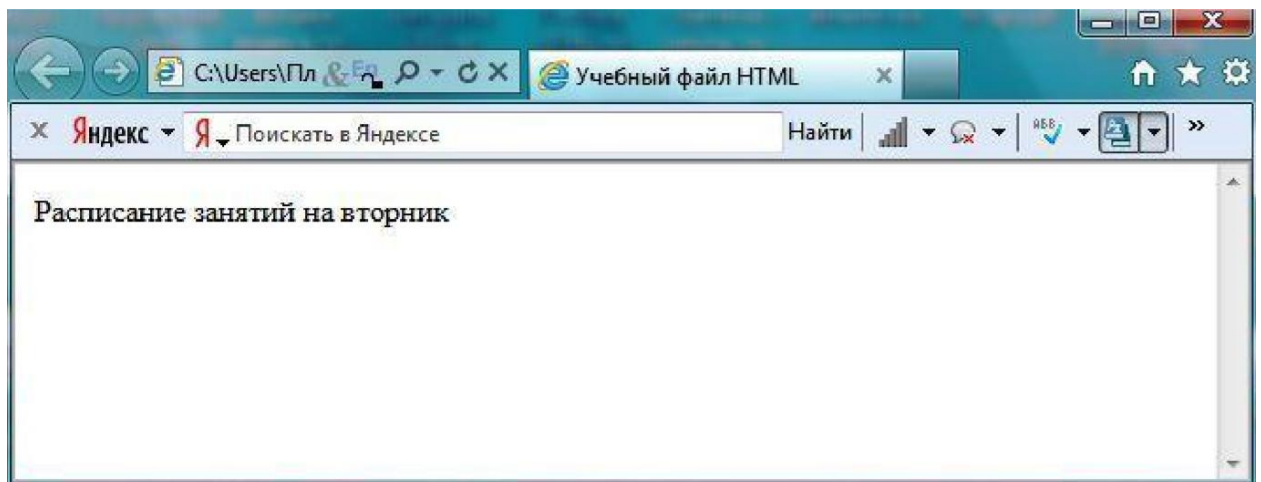


Рисунок 1

На экране вы увидите результат работы, изображенный на рисунке 1.

Задание № 2. Управление расположением текста на экране

1. При необходимости откройте текст Web-страницы в Блокноте (1 щелчок правой клавишей мыши по файлу RASP.HTML, в контекстном меню выбрать команду Открыть с помощью... и выбрать программу Блокнот). При необходимости открыть файл в браузере – двойной клик по значку файла левой клавишей мыши.

2. Внести изменения в файл RASP.HTML, расположив слова Расписание, занятий, на вторник на разных строках.

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY>
    Расписание занятий на вторник
  </BODY>
</HTML>
```

3. Сохраните текст с внесенными изменениями в файле RASP.HTML (меню Файл | Сохранить). Если у вас уже отображается Web-страница, то вам достаточно переключиться на панели задач на программу браузера и обновить эту страницу (кнопка). Изменилось ли отображение текста на экране? Не удивляйтесь тому, что внешний вид вашей Web-страницы не изменился.

Не забывайте каждый раз сохранять текст Web-страницы при ее корректировке в программе Блокнот и обновлять страницу при ее просмотре в программе браузера.

Задание № 3. Некоторые специальные команды форматирования текста

Существуют специальные команды, выполняющие перевод строки и задающие начало нового абзаца. Кроме того существует команда, запрещающая программе браузера изменять каким-либо образом форматирование текста и позволяет точно воспроизвести на экране заданный фрагмент текстового файла. Тег перевода строки
 отделяет строку от последующего текста или графики. Тег абзаца <P> тоже отделяет строку, но еще добавляет пустую строку, которая зрительно выделяет абзац. Оба тега являются одноэлементными, тег <P> – двойной, т.е. требуется закрывающий тег.

1. Внесите изменения в файл RASP.HTML

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY>
    <P>Расписание</P>
    <BR>занятий<BR>
    на вторник
  </BODY>
</HTML>
```

2. Сохраните внесенные изменения, переключитесь на панели задач на программу браузера, обновите Web-страницу. Как изменилось отображение текста на экране? Выглядеть ваша Web-страница будет примерно так, как показано на рисунке 2.

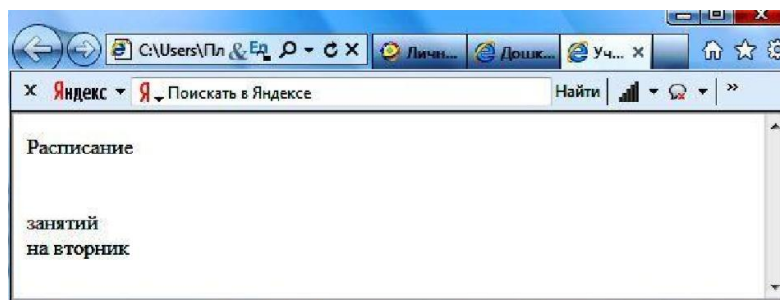


Рисунок 2

Задание № 4. Выделение фрагментов текста

1. Внести изменения в текст файла RASP.HTML

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY>
    <B>Расписание</B>
    <I> занятий</I>
    <U> на вторник</U>
  </BODY>
</HTML>
```

2. Посмотрите полученную Web-страницу.

Возможно использование комбинированных выделений текста.

```
<I><B>Расписание</B></I> <I><U> занятий</U></I> <U> на вторник</U>
```

Но при этом необходимо помнить следующее правило использования комбинированных тегов:

```
<Тег_1><Тег_2> ... </Тег_2></Тег_1> – правильная запись.
```

```
<Тег_1><Тег_2> ... </Тег_1></Тег_2> – ошибочная запись.
```

Обратите внимание на «вложенность» тегов, она напоминает «вложенность» скобок.

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY>
    <P><H1>Расписание</H1></P>
    <I> занятий</I><U>на вторник</U>
  </BODY>
</HTML>
```

Задание № 5. Задание размеров символов Web-страницы

Существует два способа управления размером текста, отображаемого браузером:

- использование стилей заголовка,
- задание размера шрифта основного документа или размера текущего шрифта.

Используется шесть тегов заголовков: от <H1> до <H6> (тег двойной, т.е. требует закрытия). Каждому тегу соответствует конкретный стиль, заданный параметрами настройки браузера.

1. Внесите изменения в файл RASP.HTML
2. Просмотрите свою Web-страницу. На экране вы увидите то, что отображено на рисунке 3

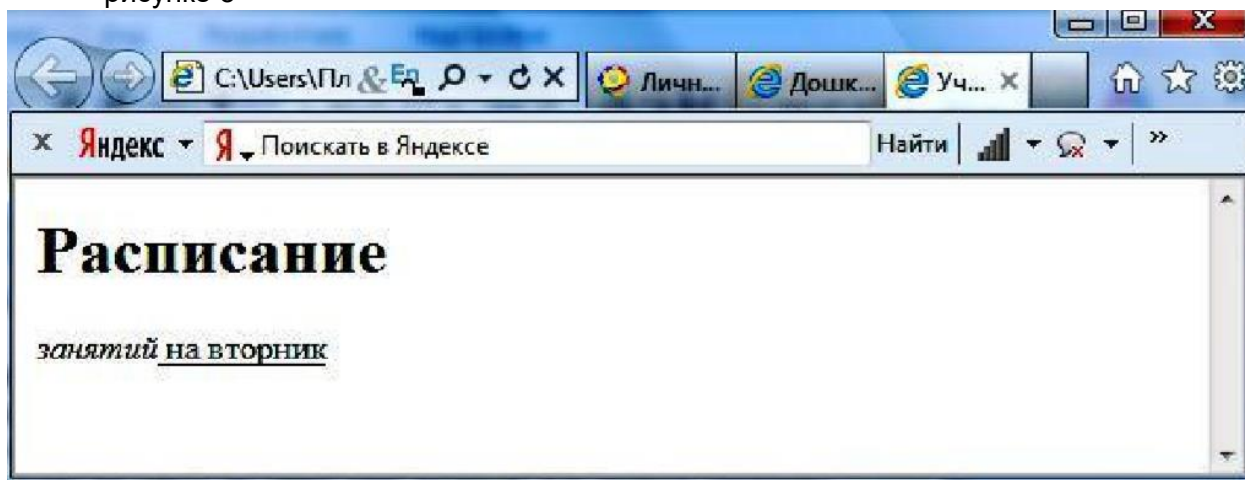


Рисунок 3

Задание № 6. Установка размера текущего шрифта

Тег шрифта позволяет задавать размер текущего шрифта в отдельных местах текста в диапазоне от 1 до 7.

1. Внесите изменения в текст RASP.HTML

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY>
    <FONT SIZE="7">Расписание</FONT>
    занятий на вторник
  </BODY>
</HTML>
```

2. Самостоятельно измените размер текста «занятий на вторник», используя тег .
3. Измените оформление текста HTML-документа, используя тег выделения фрагментов и тег перевода строки и абзаца.

Задание № 7. Установка гарнитуры и цвета шрифта












Тег предоставляет возможности управления гарнитурой, цветом и размером текста. Изменение гарнитуры текста выполняется простым добавлением к тегу атрибута FACE. Например, для отображения текста шрифтом Arial необходимо записать:

Для изменения цвета шрифта можно использовать в теге атрибут COLOR="X". Вместо "X" надо подставить английское название цвета в кавычках (" "), либо его шестнадцатеричное значение. При задании цвета шестнадцатеричным числом необходимо представить этот цвет разложенным на три составляющие: красную (R – Red), зелёную (G – Green), синюю (B – blue), каждая из которых имеет значение от 00 до FF. В

этом случае мы имеем дело с так называемым форматом RGB. Примеры записи текста в формате RGB приведены в Таблице 1:

Таблица 1

Запись текста в формате RGB

Цвет	RRGGBB	Цвет	RRGGBB	Цвет	RRGGBB
Black Черный	 000000	Purple Фиолетовый	 FF00FF	Green Зеленый	 00FF00
White Белый	FFFFFF	Yellow Желтый	 FFFF00	Azure Бирюзовый	 00FFFF
Red Красный	 FF0000	Brown Коричневый	 996633	Blue Синий	 0000FF
Orange Оранжевый	 FF8000	Violet Лиловый	 B000FF	Gray Серый	 A0A0A0

1. Внесите изменения в файл RASP.HTML

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY>
    <U><I><B><FONT COLOR="#FF0000" FACE="ARIAL "
SIZE="7">Расписание</FONT></B></I></U>
    занятий на вторник
  </BODY>
</HTML>
```

2. Самостоятельно измените размер, цвет, гарнитуру стиль текста документа.

Задание № 8. Выравнивание текста по горизонтали

1. Внесите изменения в файл RASP.HTML

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY>
    <P ALIGN="CENTER">
      <FONT COLOR="#008080" SIZE="7">
        <B>Расписание</B>
      </FONT><BR>
      <FONT SIZE="6">
        <I>занятий на вторник</I>
      </FONT>
    </P>
  </BODY>
</HTML>
```

2. Просмотрите изменения в браузере. На экране вы увидите то, что показано на рисунке 4.

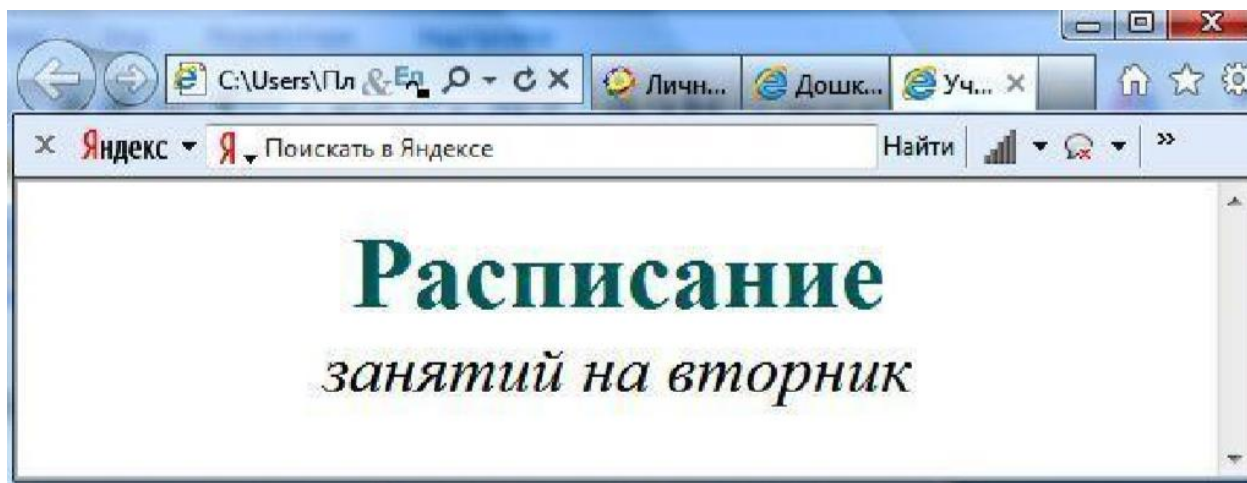


Рисунок 4

Задание № 9. Задание цвета фона и текста

При изображении фона и цвета браузеры используют цвета, установленные по умолчанию, – они заданы параметрами настройки браузера. Если вы хотите задать другие цвета, то это надо сделать в начале файла HTML в теге <BODY>. Атрибут BGCOLOR= определяет цвет фона страницы, атрибут TEXT= задает цвет текста для всей страницы, атрибуты LINK= и VLINK= определяют соответственно цвета непросмотренных и просмотренных ссылок (последние два примера будут рассмотрены позже).

1. Внесите изменения в файл RASP.HTML

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFCC" TEXT="#330066">
    <P ALIGN="CENTER">
      <FONT COLOR="#008080" SIZE="7">
        <B>Расписание</B>
      </FONT><BR>
      <FONT SIZE="6">
        <I> занятий на вторник</I>
      </FONT>
    </P>
  </BODY>
</HTML>
```

2. Просмотрите изменения Web-страницы в браузере.

Задание № 10. Размещение графики на Web-странице

Тег позволяет вставить изображение на Web-страницу. Оно появится в том месте документа, где находится этот тег. Тег является одиночным. Необходимо помнить, что графические файлы должны находиться в той же папке, что и файл HTML, описывающий страницу. Графика в Web, как правило, распространяется в трех форматах: GIF, JPG, PNG.

Для выполнения следующего задания поместите файл с именем CLOCK.JPG (или другим именем) в рабочую папку.

Следует помнить, что для браузера важно, в каком регистре вы задаете описание имени и типа файла. Выработайте для себя определенное правило и строго следуйте ему. Если вы размещаете файл графического изображения во вложенной папке, то при описании изображения необходимо указывать путь доступа к файлу изображения, отображая вложенность папок.

1. Внесите изменения в файл RASP.HTML

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFCC" TEXT="#330066">
    <P ALIGN="CENTER">
      <FONT COLOR="#008080" SIZE="7">
        <B>Расписание</B>
      </FONT><BR>
      <FONT SIZE="6">
        <I>занятий на вторник</I>
      </FONT><BR><BR>
      <IMG SRC="CLOCK.PNG">
    </P>
  </BODY>
</HTML>
```

2. Просмотрите изменения вашей Web-страницы в браузере.

На экране вы увидите то, что показано на рисунке 5.

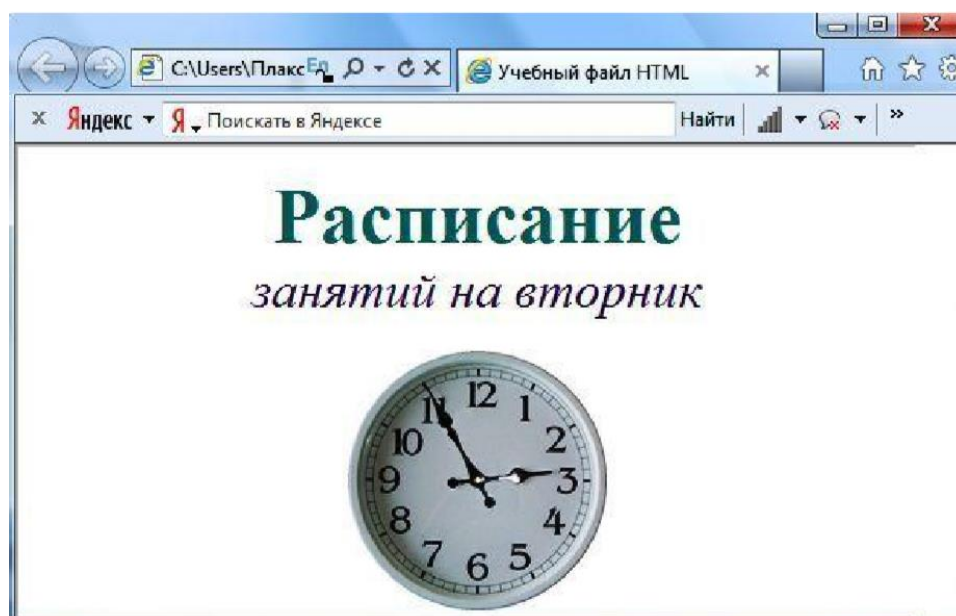


Рисунок 5

Тег имеет немало атрибутов, описанных в таблице 2. Эти атрибуты можно задавать дополнительно и располагаться они могут в любом месте тега после кода IMG.

Таблица 2.

Атрибуты изображения.

Атрибут	Формат	Описание
ALT		Задаёт текст, заменяющий изображение в том случае, если браузер не воспринимает изображение
BORDER		Задаёт толщину рамки вокруг изображения. Измеряется в пикселях
ALIGN		Задаёт выравнивание изображения относительно текста: <ul style="list-style-type: none"> • относительно текста выровнена верхняя часть изображения – "TOP", • относительно текста выровнена нижняя часть изображения – "BOTTOM", • относительно текста выровнена средняя часть изображения – "MIDDLE".
HEIGHT		Задаёт вертикальный размер изображения внутри окна браузера
WIDTH		Задаёт горизонтальный размер изображения внутри окна браузера
VSPACE		Задаёт добавление верхнего и нижнего пустых полей
HSPACE		Задаёт добавление левого и правого пустых полей

Задание № 11. Использование атрибутов изображения

1. Самостоятельно внесите изменения в текст файла RASP.HTML: опробуйте использование таких атрибутов графики, как ALT, BORDER, ALIGN, HEIGHT, WIDTH, VSPACE, HSPACE.

Всегда обращайте внимание на размер графического файла (в байтах), так как это влияет на время загрузки Web-страницы.

2. Просмотрите изменения вашей Web-страницы в браузере.

Задание № 12. Установка фонового изображения на Web-странице

Фоновое изображение – это графический файл с небольшим рисунком, который многократно повторяется, заполняя все окно браузера независимо от его размеров. Графика, используемая в качестве фоновой, задается в теге <BODY>.

1. Внесите изменения в файл RASP.HTML, предварительно подготовив и сохранив в рабочей папке графический файл фонового рисунка (FON.PNG).

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY BACKGROUND="FON.PNG" TEXT="#330066">
    <P ALIGN="CENTER">
      <FONT COLOR="#008080" SIZE="7">
        <B>Расписание</B>
      </FONT><BR>
      <FONT SIZE="6">
        <I>занятий на вторник</I>
      </FONT><BR><BR>
      <IMG SRC="CLOCK.PNG" ALIGN="MIDDLE">
    </P>
```

```
</BODY>  
</HTML>
```

На экране вы увидите то, что изображено на рисунке 6.

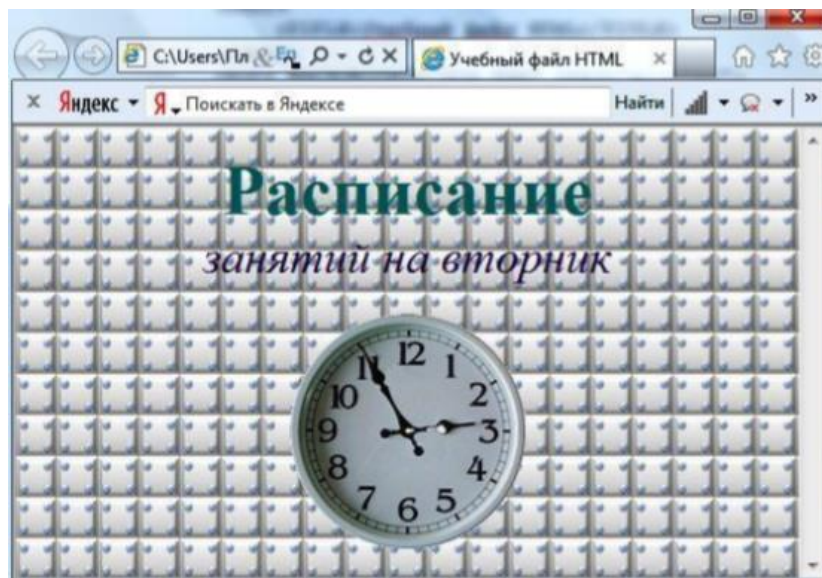


Рисунок 6

Рисунок, который использовался в качестве фонового, имеет вид

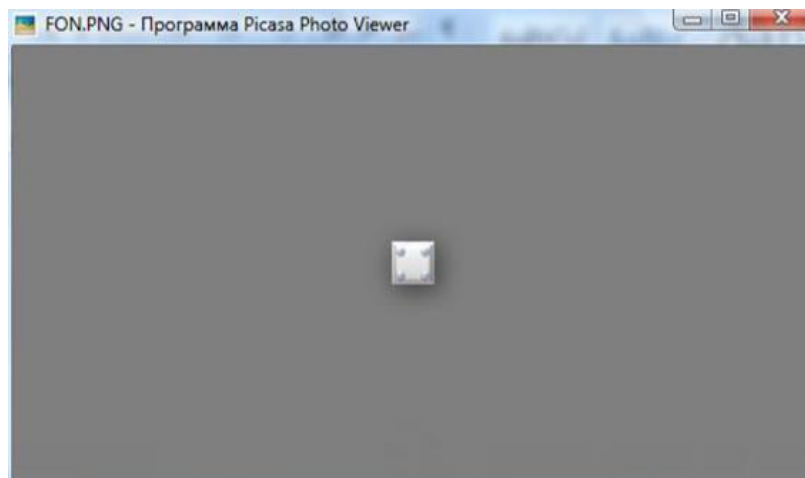


Рисунок 7

2. Поэкспериментируйте с фоновым рисунком Web-страницы и выберите оптимальный с вашей точки зрения.

Задание № 13. Создание таблицы

Таблица является частью HTML-документа. Она представляет собой прямоугольную сетку, состоящую из вертикальных столбцов и горизонтальных строк. Пересечение строки и столбца называется ячейкой таблицы. Ячейка может содержать в себе текст, графику или другую таблицу. Таблица состоит из трех основных частей:

- названия таблицы,
- заголовков столбцов,
- ячеек таблицы.

Таблица в Web-документе заполняется по строкам (слева направо по строке, затем переход на новую строку). Каждая ячейка таблицы должна быть заполнена (хотя бы пробелом, которые используются для создания пустых ячеек).

1. Запустите программу Блокнот и наберите текст следующей Web-страницы. Применяйте приемы копирования при создании таблицы, работая в программе Блокнот.

2. Сохраните файл в личной рабочей папке под именем 5.HTML 3. Для просмотра созданной Web-страницы в окне личной рабочей папки двойным щелчком левой клавиши мыши загрузите браузер.

```
<HTML>
<HEAD>
<TITLE>Расписание занятий 5 классов</TITLE>
</HEAD>
  <BODY BGCOLOR="FFFFFF">
    <P ALIGN="CENTER">
      <FONT COLOR="RED" SIZE="6" FACE="ARIAL">
        <B>5 класс</B>
      </FONT><BR>
    </P>
    <FONT COLOR="BLUE" SIZE="4" FACE="COURIER">
      <B>Понедельник</B>
    </FONT><BR>
    <TABLE BORDER="1" WIDTH=100% BGCOLOR="#99CCCC">
      <TR BGCOLOR="#CCCCFF" ALIGN="CENTER">
        <TD>Урок</TD>
        <TD>5 "А"</TD>
        <TD>5 "Б"</TD>
        <TD>5 "В"</TD>
      </TR>
      <TR>
        <TD>1</TD>
        <TD>Русский язык</TD>
        <TD>Литература</TD>
        <TD>История</TD>
      </TR>
      <TR>
        <TD>2</TD>
        <TD>Математика</TD>
        <TD>Информатика</TD>
        <TD>Английский язык</TD>
      </TR>
      <TR>
        <TD>3</TD>
        <TD>История</TD>
        <TD>Математика</TD>
        <TD>Информатика</TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```

На экране вы увидите то, что показано на рисунке 8.

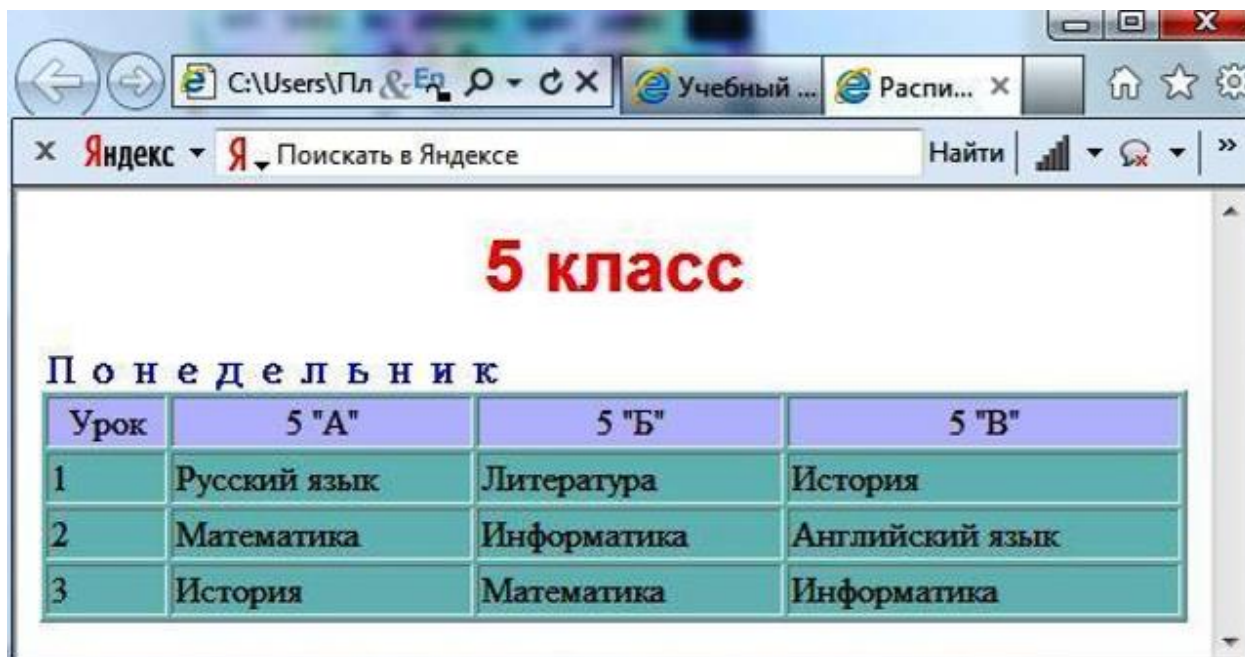


Рисунок 8

Задание № 14. Построение гипертекстовых связей

Важнейшим средством языка HTML является возможность включения в документ ссылок на другие документы. Возможны ссылки:

- на удаленный HTML-файл,
- на некоторую точку в текущем HTML-документе,
- на любой файл, не являющийся HTML-документом.

В качестве ссылки можно использовать любой текст или графику.

Ссылки в пределах одного документа

Такие ссылки требуют двух частей: метки и самой ссылки. Метка определяет точку, на которую происходит переход по ссылке. Ссылка использует имя метки. Ссылки выделяют цветом или подчеркиванием в зависимости от того, как настроен браузер. Для изменения цвета ссылки используются атрибуты LINK= и VLINK= тега <BODY>.

Описание ссылки

```
<A HREF="#ПН">Понедельник</A>
```

Перед именем метки (ПН), указывающей, куда надо перейти по ссылке, ставится символ #. Между символами ">" и "<" располагается текст ("Понедельник"), на котором должен быть произведен щелчок для перехода по ссылке.

Определим метку

```
<A NAME="ПН">Понедельник</A>
```

1. Дополните файл 5.HTML описанием таблицы, содержащей названия дней недели, поместив его в начало Web-страницы.

```
...
<TABLE WIDTH=100%>
  <TR>
    <TD>Понедельник</TD>
```



```

        <TD>Вторник</TD>
        <TD>Среда</TD>
        <TD>Четверг</TD>
        <TD>Пятница</TD>
        <TD>Суббота</TD>
    </TR>
</TABLE>
<BR>
...

```

2. Вставьте в файл 5.HTML метку, указывающую на понедельник.

```

...
<FONT COLOR="BLUE" SIZE="4" FACE="COURIER">
    <B>
        <A NAME="#ПН">Понедельник</A>
    </B>
</FONT><BR>
...

```

3. Вставьте в таблицу с названиями дней недели ссылку для выбранной метки:

```

...
<TABLE WIDTH=100%>
    <TR>
        <TD><A HREF="#ПН">Понедельник</A></TD>
        <TD>Вторник</TD>
        <TD>Среда</TD>
    </TR>
...

```

4. Создайте таблицы расписаний для других дней недели. 5. Сохраните файл 5.HTML в личной рабочей папке. 6. Просмотрите полученную Web-страницу.

На экране вы увидите то, что изображено на рисунке 9.

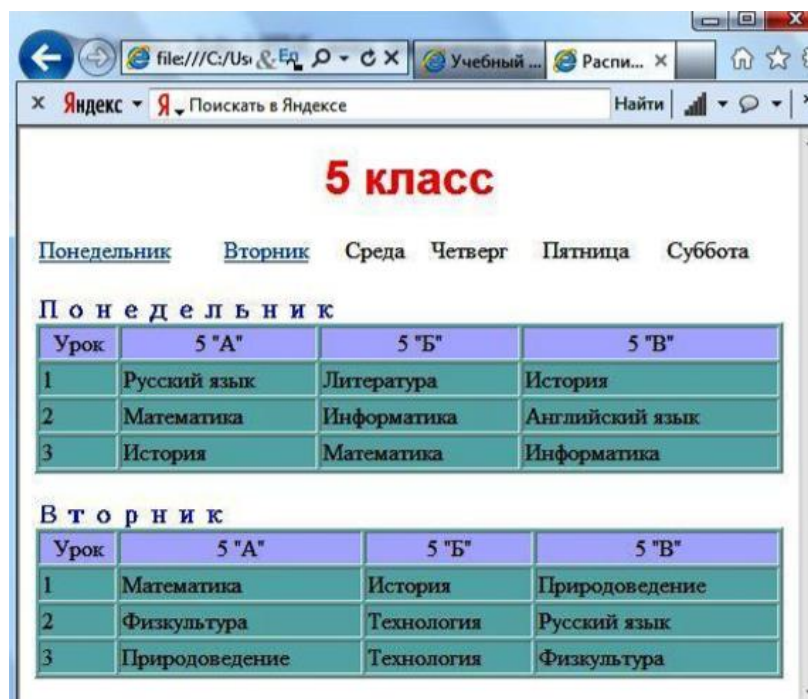


Рисунок 9

Задание № 15. Создание ссылки на другой HTML-документ

Ссылки позволяют щелчком на выделенном слове или фразе перейти к другому файлу.

Опишем ссылку:

```
<A HREF="5.HTML">5 класс</A>
```

После имени файла (5.HTML) между символами «>» и «<» располагается текст («5 класс»), на котором должен быть произведен щелчок для перехода к этому файлу.

1. Внесите изменения в файл RASP.HTML

```
<HTML>
  <HEAD>
    <TITLE>Учебный файл HTML</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF" TEXT="#330066">
    <P ALIGN="CENTER">
      <FONT COLOR="#008080" SIZE="7">
        <B>Расписание</B>
      </FONT><BR>
      <FONT SIZE="6">
        <I> занятий на вторник</I>
      </FONT><BR><BR>
      <IMG SRC="CLOCK.PNG" ALIGN="TOP">
    </P>
    <CENTER>
      <TABLE WIDTH=60%>
        <TR>
          <TD><A HREF="5.HTML">5 класс</A></TD>
          <TD>6 класс</TD>
        </TR>
        <TR>
          <TD>7 класс</TD>
          <TD>8 класс</TD>
        </TR>
        <TR>
          <TD>9 класс</TD>
          <TD>10 класс</TD>
        </TR>
        <TR>
          <TD>11 класс</TD>
        </TR>
      </TABLE>
    </CENTER>
  </BODY>
</HTML>
```

2. Сохраните файл RASP.HTML 3. Просмотрите полученную Web-страницу. На экране вы увидите то, что изображено на рисунке 10.

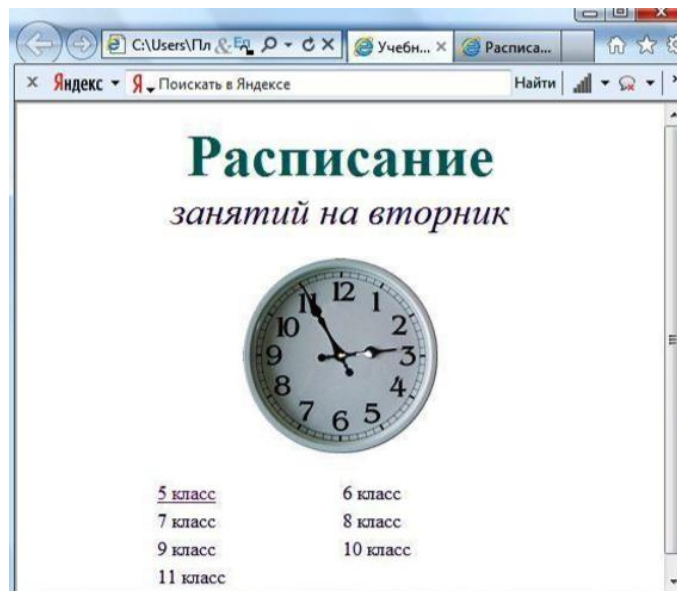


Рисунок 10

Задание № 16. Создание ссылки на другой HTML-документ

1. Внесите изменения в файл 5.HTML так, чтобы в конце страницы была ссылка на главную страницу Расписание занятий 5 классов (RASP.HTML). В качестве ссылки используйте графический файл (HOME.GIF) следующим образом:

```
...
</TABLE><BR>
  <CENTER>
    <A HREF="RASP.HTML">
      <IMG SRC="HOME.PNG" BORDER="0">
    </A>
  </CENTER>
...
```

2. Просмотрите полученную Webстраницу.

На экране вы увидите то, что показано на рисунке 11.

В качестве ссылки выступает рисунок – стрелка

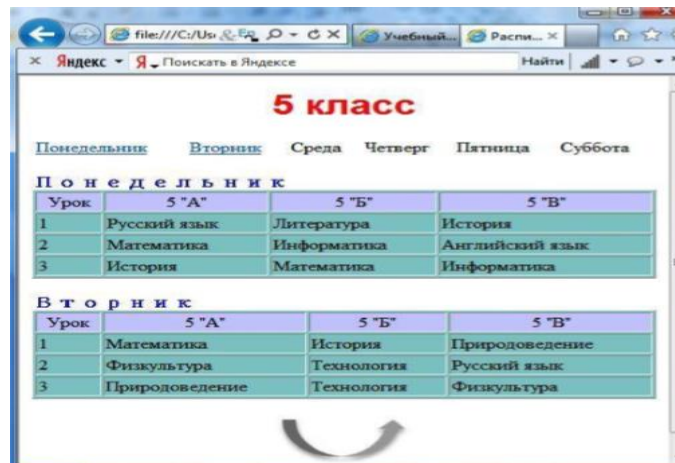


Рисунок 11

Задание № 17. Самостоятельное итоговое задание

Разработайте Web-страницы, рассказывающие о вашем классе. На головной странице разместите рассказ о классе, классном руководителе. Рассказы об учениках разместите на отдельных Web-страницах. Укажите ссылки на страницы учеников с головной Web-страницы. Не забудьте разместить ссылки возврата на головную страницу.

Как подготовить хорошую Web-страницу

1. Следует обратить внимание на простоту и логичность расположения информации на ваших страницах. Один из способов сделать информацию более легкой для восприятия – оставить на странице достаточно свободного места, не содержащего ни текста, ни рисунков. Страница, содержащая много информации, только отпугнет посетителя.
2. Постарайтесь представить информацию в виде списков или таблиц так, чтобы можно было достаточно легко найти важные сведения.
3. Не размещайте одно изображение сразу за другим. Попробуйте распределить их по документу, оставив достаточно свободного пространства.
4. Информация должна размещаться частями, легкими для восприятия. Обратите внимание на длину абзацев. Если абзац слишком длинный, разбейте его на несколько небольших абзацев.
5. Если Web-страница имеет большой объем, то, возможно, вам следует вставить ссылки, позволяющие пользователю быстро перемещаться между частями одного документа. Иногда имеет смысл вместо одного длинного документа подготовить одну страницу, содержащую перечень тем, каждую из которых раскрыть на отдельной Web-странице, и установить ссылки на соответствующие Web-страницы.
6. Использование графики может дополнительно привлечь пользователей. Но необходимо помнить о времени загрузки вашей страницы, которое определяется количеством и объемом графической информации. Красивая картинка не произведет никакого впечатления, если для того, чтобы ее увидеть, придется долго ждать, пока она загрузится.

Тестирование

Перед тем как выставлять свои Web-страницы на сервер необходимо их протестировать. Созданные документы должны пройти «локальную проверку» в пределах вашего жесткого диска. При проверке используйте разные браузеры. Вы увидите различия, которые могут оказаться довольно существенными.

В рамках тестирования необходимо сделать следующее:

1. Проверить правописание. Выполните автоматизированную проверку правописания текста (для этого можно использовать Microsoft Word) или попросите кого-нибудь выполнить корректуру.
2. Проверить навигацию. Убедитесь, что на каждой странице присутствуют необходимые средства навигации, все ссылки работают правильно.
3. Проверить доступ к внешним файлам. Выясните, размещены ли графические, звуковые или видеофайлы там, где они могут быть найдены и откуда их можно загрузить (должен быть правильно указан путь доступа). Для неграфических браузеров требуется задать подменяющие текстовые сообщения.
4. Проверить, допустимо ли время загрузки.

5. Осуществить проверку ваших Web-страниц посторонним лицом. Попросите кого-нибудь, кто не знаком с вашими документами, пройти их от начала до конца. Иногда при этом выясняются такие факты, каких вы сами ни за что бы не заметили.

Критерии оценки:

«Отлично» - студент выполнил работу в полном объеме и самостоятельно может объяснить каждую строчку кода. При обращении с сайтом весь функционал работает правильно (На усмотрение учителя возможна одна незначительная ошибка).

«Хорошо» - студент выполнил работу с 1 значительной ошибкой и одной незначительной ошибкой (ошибка считается незначительной на усмотрение преподавателя).

«Удовлетворительно» - студент выполнил работу с 2 значительными ошибками и одной незначительной из списка ошибок.

«Неудовлетворительно» - студент выполнил работу с 3 значительными ошибками и одной незначительной ошибкой.

Список ошибок:

1. Ссылка не работает.
2. Расписание заполнено, данных меньше необходимого.
3. Функционал страницы работает с ошибками.
4. Допущены синтаксические ошибки в коде (Некоторые синтаксические ошибки считаются отдельно).
5. Студент не может объяснить некоторые элементы кода.

Блочные элементы

Блочные элементы характеризуются тем, что занимают всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки.

`<blockquote>`

Предназначен для выделения длинных цитат внутри документа. Текст, обозначенный этим тегом, традиционно отображается как выровненный блок с отступами слева и справа (примерно по 40 пикселей), а также с пустым пространством сверху и снизу.

`<div>`

Тег `<div>` относится к универсальным блочным контейнерам и применяется в тех случаях, где нужны блочные элементы без дополнительных свойств. Также с помощью тега `<div>` можно выравнивать текст внутри этого контейнера с помощью атрибута `align`.

`<h1>,...,<h6>`

Эта группа тегов определяет текстовые заголовки разного уровня, которые показывают относительную важность секции, расположенной после заголовка.

`<hr>`

Рисует горизонтальную линию, которая по своему виду зависит от используемых атрибутов. Линия всегда начинается с новой строки, а после нее все элементы отображаются на следующей строке.

`<p>`

Определяет параграф (абзац) текста.

`<pre>`

Задаёт блок предварительно форматированного текста. Такой текст отображается обычно моноширинным шрифтом и со всеми пробелами между словами. В HTML любое количество пробелов идущих в коде подряд на веб-странице показывается как один. Тег `<pre>` позволяет обойти эту особенность и отображать текст как требуется разработчику.

Следующие теги не должны размещаться внутри контейнера `<pre>`: `<big>`, ``, `<small>`, `<sub>` и `<sup>`.

Строчные элементы

Строчными называются такие элементы веб-страницы, которые являются непосредственной частью другого элемента, например, текстового абзаца. В основном они используются для изменения вида текста или его логического выделения.

`<a>`

Тег `<a>` является одним из важных элементов HTML и предназначен для создания ссылок. В зависимости от присутствия атрибутов `name` или `href` тег `<a>` устанавливает ссылку или якорь.

``

Определяет жирное начертание шрифта.

`<big>`

Тег `<big>` увеличивает размер шрифта на единицу по сравнению с обычным текстом. В HTML размер шрифта измеряется в условных единицах от 1 до 7, средний размер текста, используемый по умолчанию, принят 3. Таким образом, добавление тега `<big>` увеличивает текст на одну условную единицу.

Тег **
** устанавливает перевод строки в том месте, где этот тег находится. В отличие от тега параграфа **<p>**, использование тега **
** не добавляет пустой отступ перед строкой.

Тег **** предназначен для акцентирования текста. Браузеры отображают такой текст курсивным начертанием.

<i>

Устанавливает курсивное начертание шрифта.

Тег **** предназначен для отображения на веб-странице изображений в графическом формате GIF, JPEG или PNG. Если необходимо, то рисунок можно сделать ссылкой на другой файл, поместив тег **** в контейнер **<a>**. При этом вокруг изображения отображается рамка, которую можно убрать, добавив атрибут **border="0"** в тег ****.

<small>

Тег **<small>** уменьшает размер шрифта на единицу по сравнению с обычным текстом. По своему действию похож на тег **<big>**, но действует с точностью до наоборот.

Универсальный тег, предназначенный для определения строчного элемента внутри документа.

Тег **** предназначен для акцентирования текста. Браузеры отображают такой текст жирным начертанием.

<sub>

Отображает шрифт в виде нижнего индекса. Текст при этом располагается ниже базовой линии остальных символов строки и уменьшенного размера — H₂O.

<sup>

Отображает шрифт в виде верхнего индекса. По своему действию похож на **<sub>**, но текст отображается выше базовой линии текста — m².

Разница между блочными и строчными элементами следующая.

- Строчные элементы могут содержать только данные или другие строчные элементы, а в блочные допустимо вкладывать другие блочные элементы, строчные элементы, а также данные. Иными словами, строчные элементы никак не могут хранить блочные элементы.
- Блочные элементы всегда начинаются с новой строки, а строчные таким способом
- не акцентируются.

Блочные элементы занимают всю доступную ширину, например, окна браузера, а ширина строчных элементов равна их содержимому плюс значения отступов, полей и границ.

Универсальные элементы

Особенность этих тегов состоит в том, что они в зависимости от контекста могут использоваться как блочные или встроенные элементы.

Тег **** используется для выделения текста, который был удален в новой версии документа. Подобное форматирование позволяет отследить, какие изменения в тексте

документа были сделаны. Браузеры обычно помечают текст в контейнере `` как перечеркнутый.

`<ins>`

Тег `<ins>` предназначен для акцентирования вновь добавленного текста и обычно применяется наряду с тегом ``.

Браузеры помечают содержимое контейнера `<ins>` подчеркиванием текста.

Теги для списков

Списком называется взаимосвязанный набор отдельных фраз или предложений, которые начинаются с маркера или цифры. Списки предоставляют возможность упорядочить и систематизировать разные данные и представить их в наглядном и удобном для пользователя виде.

``

Тег `` устанавливает нумерованный список, т.е. каждый элемент списка начинается с числа или буквы и увеличивается по нарастающей.

``

Устанавливает маркированный список, каждый элемент которого начинается с небольшого символа — маркера.

``

Тег `` определяет отдельный элемент списка. Внешний тег `` или `` устанавливает тип списка — маркированный или нумерованный.

`<dd>`, `<dt>`, `<dl>`

Тройка элементов предназначена для создания списка определений. Каждый такой список начинается с контейнера `<dl>`, куда входит тег `<dt>` создающий термин и тег `<dd>` задающий определение этого термина. Закрывающий тег `</dd>` не обязателен, поскольку следующий тег сообщает о завершении предыдущего элемента. Тем не менее, хорошим стилем является закрывать все теги. Теги для таблиц

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления табличных данных.

`<table>`

Служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью тегов `<tr>` и `<td>`.

`<td>`

Предназначен для создания одной ячейки таблицы. Тег `<td>` должен размещаться внутри контейнера `<tr>`, который в свою очередь располагается внутри тега `<table>`.

`<th>`

Тег `<th>` предназначен для создания одной ячейки таблицы, которая обозначается как заголовочная. Текст в такой ячейке отображается браузером обычно жирным шрифтом и выравнивается по центру.

`<tr>`

Тег `<tr>` служит контейнером для создания строки таблицы.

Правила вложений для тега `<a>`

Любая ссылка является встроенным элементом, поэтому для нее действуют те же правила, что и для встроенных элементов. А именно, нельзя размещать внутри тега `<a>`

блочные элементы, но допустимо делать наоборот, и вкладывать ссылку в блочный контейнер. В примере 1.9 показано ошибочное и правильное использование тегов.

Пример 1.9. Вложение тегов

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8"> <title>Ошибки при
использовании ссылок</title>
</head>
<body>
  <a href="lion.html"><h1>Охота на льва</h1></a>
  <h1><a href="lion.html">Как поймать льва в пустыне</a></h1> </body>
</html>
```

В строке 8 данного примера содержится типичная ошибка — тег `<h1>` располагается внутри контейнера `<a>`. Поскольку `<h1>` это блочный элемент, то его недопустимо вкладывать внутрь ссылки. В строке 9 этого же примера показан корректный вариант.

Атрибуты ссылок

Основной атрибут `href` тега `<a>` мы уже освоили, рассмотрим еще несколько полезных, но необязательных атрибутов этого тега.

target

По умолчанию, при переходе по ссылке документ открывается в текущем окне или фрейме. При необходимости, это условие может быть изменено атрибутом `target` тега `<a>`. Синтаксис следующий.

```
<a target="имя окна">...</a>
```

В качестве значения используется имя окна или фрейма, заданное атрибутом `name`. Если установлено несуществующее имя, то будет открыто новое окно. В качестве зарезервированных имен применяются следующие.

- `_blank` — загружает страницу в новое окно браузера.
- `_self` — загружает страницу в текущее окно (это значение задается по умолчанию).
- `_parent` — загружает страницу во фрейм-родитель, если фреймов нет, то это значение работает как `_self`.
- `_top` — отменяет все фреймы и загружает страницу в полном окне браузера, если фреймов нет, то это значение работает как `_self`.

В примере 1.10 показано, как сделать, чтобы ссылка открывалась в новом окне.

Пример 1.10. Открытие ссылки в новом окне

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Ссылка в новом окне</title>
</head>
<body>
  <p><a href="new.html" target="_blank">Открыть      в новом
окне</a></p>
```

```
</body>
</html>
```



Атрибут **target** корректно использовать только при переходном **<!DOCTYPE>**, при строгом **<!DOCTYPE>** будет сообщение об ошибке, поскольку в этой версии HTML **target** уже не поддерживается.

Учтите также, что пользователи не любят, когда ссылки открываются в новых окнах, поэтому используйте подобную возможность осмотрительно и при крайней необходимости.

title

Добавляет поясняющий текст к ссылке в виде всплывающей подсказки. Такая подсказка отображается, когда курсор мыши задерживается на ссылке, после чего подсказка через некоторое время пропадает. Синтаксис следующий.

```
<a title="текст">...</a>
```

В качестве значения указывается любая текстовая строка. Строка должна заключаться в двойные или одинарные кавычки. В примере 2.1 показано, как использовать атрибут **title** для ссылок.

Пример 1.11. Создание всплывающей подсказки

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"> <title>Подсказка к
ссылке</title>
</head>
<body>
<p><a href="zoo.html" title="Рисунки различных животных и не только...">Рисунки</a></p>
</body> </html>
```

Результат данного примера показан на рис. 1.4.

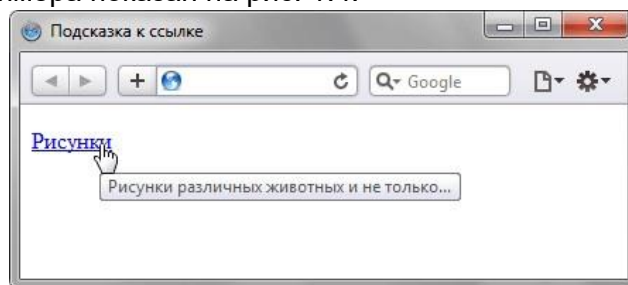


Рис. 1.4. Вид всплывающей подсказки в браузере

Цвета и оформления всплывающей подсказки зависят от настроек операционной системы и браузера, и меняться разработчиком не могут.


ЧАСТЬ 2

Простейшая анимация с использованием CSS

Цель работы: создать два блока с разными цветами, и одному из блоков добавить свойства при наведении.

Ход работы:

```
<HTML lang="ru">
<head>
  <link rel="stylesheet" href="myfile.css">
</head>
<body>
  <div class="block1">css</div>
  <div class="block2">css</div>
</body>
</HTML>
```



```
body{
  display: flex;
  justify-content: space-between
}
.block1{
  background-color: red;
  height: 500px;
  width: 500px;
  margin: 0 auto;
  color: black;
  font-size: 350px;
  border: 3px solid black;
  border-radius: 25%;
  text-align: center
}
.block1: hover{
  background-color: rgb(255, 255, 0);
  color: rgb(0, 17, 255);
  transform: rotate(45deg);
  margin-top: 50px;
```

```

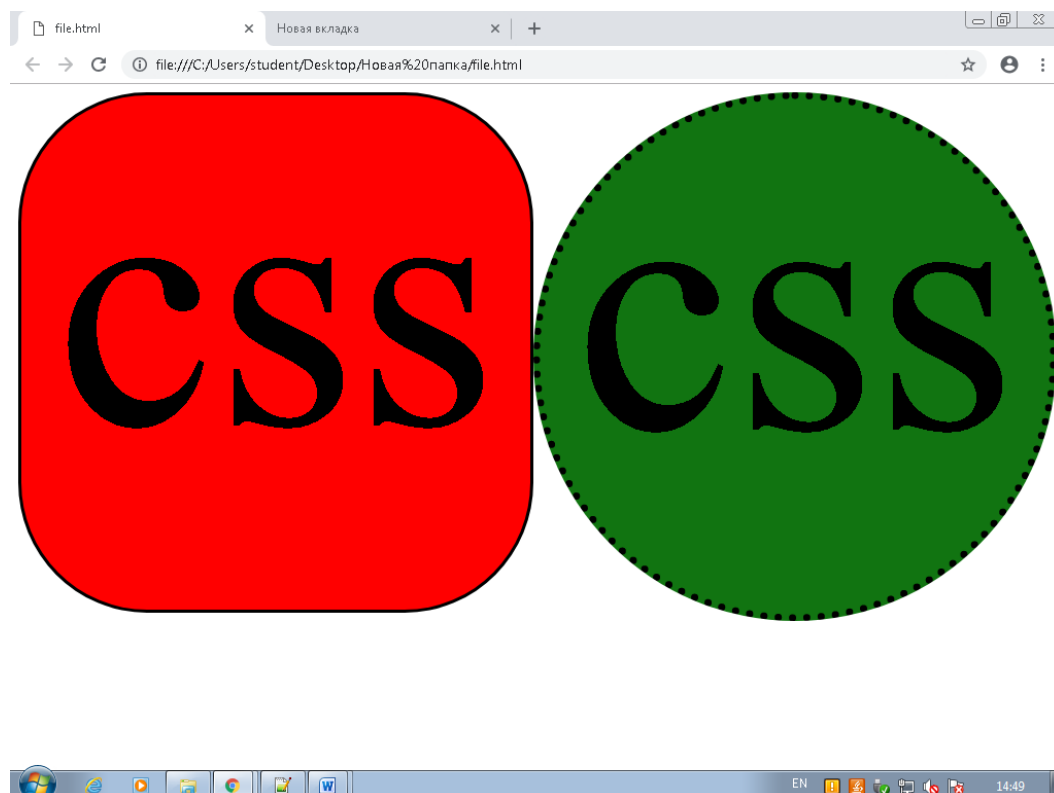
}

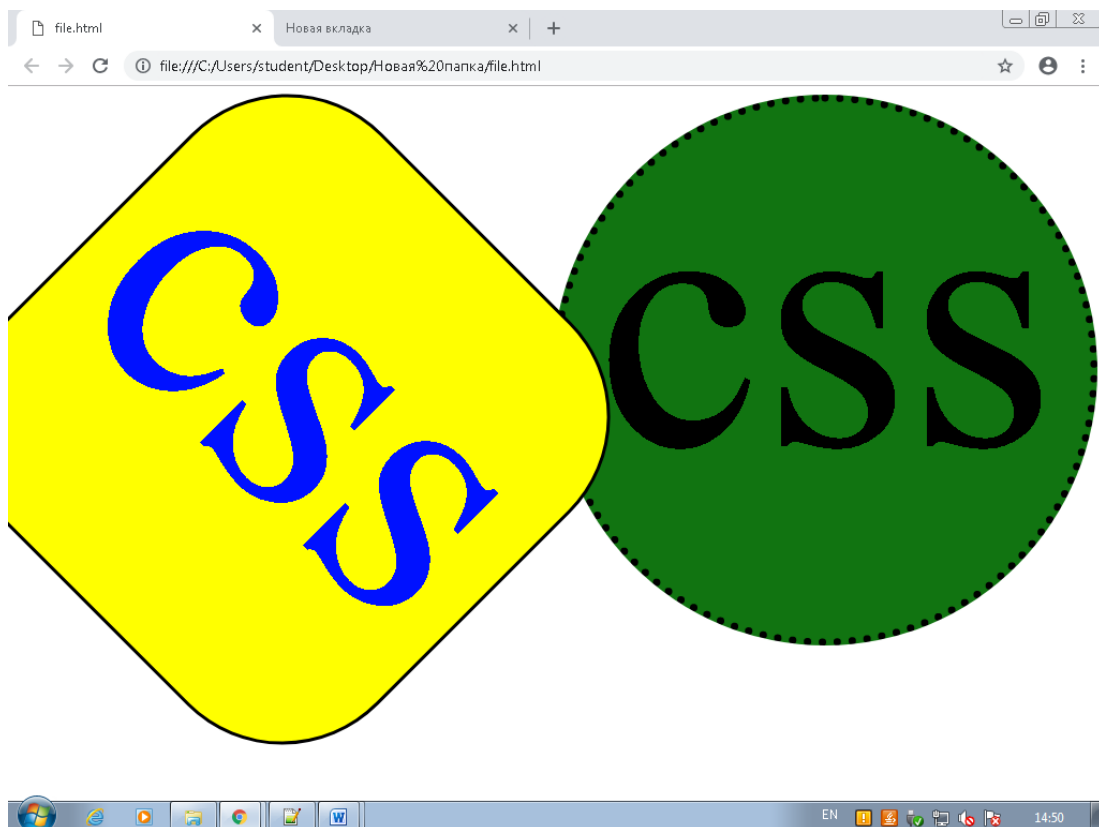
.block2{
  background-color:rgb(17,116,17);
  height:500px;
  width:500px;
  border: 7px dotted black;
  font-size: 350px;
  border-radius:50%;
  text-align:center
}

.block2:hover{
  background-color:rgb(255,255,0);
  color: rgb(0,17,255);
  transform:rotate(45deg);
  margin-top:50px;
}

```

Вот так будет выглядеть наша страничка в браузере.





Дополнительное задание: изменение шрифта, изменение цвета фона, добавление прозрачности элементам и тп.

Критерии оценки:

«Отлично» - студент выполнил работу в полном объеме и самостоятельно может объяснить каждую строчку кода. При обращении с сайтом весь функционал работает правильно (На усмотрение учителя возможна одна незначительная ошибка).

«Хорошо» - студент выполнил работу с 1 значительной ошибкой и одной незначительной ошибкой (ошибка считается незначительной на усмотрение преподавателя).

«Удовлетворительно» - студент выполнил работу с 2 значительными ошибками и одной незначительной из списка ошибок.

«Неудовлетворительно» - студент выполнил работу с 3 значительными ошибками и одной незначительной ошибкой.

Список ошибок:

1. Студент не смог выполнить поворот элемента.
2. Свойство :hover не работает.
3. Студент не смог выполнить скругление.
4. Не выполнено дополнительное задание преподавателя.
5. Допущены синтаксические ошибки в коде (Некоторые синтаксические ошибки считаются отдельно).
6. Студент не может объяснить некоторые элементы кода.

Свойства CSS. Часть 2.

Расположение элементов во флекс контейнере.

`justify-content: center;`(по центру)

`justify-content: space-around;`(пустота между элементами)

`justify-content: space-between;`

`justify-content: space-evenly;`(свойство схожее с `space-around`, но приближены к центру)

`justify-content: flex-end;`(расположение с правой стороны)

`justify-content: flex-start;`

Выравнивание блоков по вертикали во флекс контейнере.

`align-items: center;`



`align-items: flex-end;`

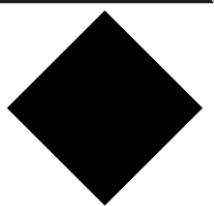


`align-items: flex-start;`



Увеличение элемента и его поворот.

`transform: rotate(45deg);`(поворот элемента на 45 градусов)



`transform: rotateY(45deg) rotateX(45deg);` (поворот элемента по оси Y и X на 45 градусов, так же можно добавить ось Z)

Поворот в
трехмерной
системе

`transform: scale(1.2);`(исходный размер у обоих квадратов одинаковый)

Обычный
квадрат

Увеличенный
квадрат

Расположение текста.

`text-align: center;`(по центру)

Обычный квадрат

`text-align: right;` (справа)

Обычный квадрат

text-align: left;

Обычный квадрат