Филиала ГБПОУ «Троицкий технологический техникум»

в с. Октябрьское

Сборник практических работ

ПМ 02 Хранение, передача и публикация цифровой информации

Профессия 09.01.03 Мастер по обработке цифровой информации

Составил мастер п/о Лысенко Н.В

Октябрьское,2023

Пояснительная записка

Сборник практических работ ПМ 02 Хранение, передача и публикация цифровой информации предназначен для студентов обучающихся по профессии 09.01.03 Мастер по обработке цифровой информации.

Сборник практических работ разработан для раздела «Язык гипертекстовой разметки HTML»

Включает в себя задания по обработке текста, таблиц, анимации и специальных символов при помощи языка гипертекстовой разметки HTML

Сборник рассмотрен и утвержден на заседании методической комиссии преподавателей профессиональных дисциплин и мастеров производственного обучения

Протокол № 4 от 23.11.2023 г

_____

Содержание

Практическая работа № 1Элементы и стили в HTML 4
Практическая работа № 2 Фреймы 11
Практическая работа № 3 Организация списков 13
Практическая работа № 4 Организация таблиц 16
Практическая работа № 5 Спецсимволы 21
Практическая работа № 6 HTML формы с использованием языка программированияJjavascript 23
Практическая работа № 7 Работа с изображением 38
Проверочная работа по теме язык гипертекстовой разметки HTML 50
1 вариант 50
Проверочная работа по теме язык гипертекстовой разметки HTML 51
2 вариант 51
Задания для самостоятельной работы 53
Список использованной литературы 58
Приложения 59
Спецсимволы 59

Практическая работа № 1Элементы и стили в HTML

Оформление текстов

Задание 1. Наберите с помощью программы «Блокнот» следующий программный код.

Сохраните его и просмотрите результат в браузере.

```
<HTML> <HEAD> <TITLE> Стихотворение
</TITLE> </HEAD>
<BODY> <H1> Александр Блок </H1>
<P> Ночь. Улица. Фонарь. Аптека. <BR> Бессмысленный и тусклый свет.
<BR> Живи еще хоть четверть века – <BR>
Все будет так. Исхода нет. </P>
<P> Умрешь, начнешь опять сначала <BR> И повторится все, как встарь:
<BR> Ночь. Ледяная мгла канала.
<BR> Аптека. Улица. Фонарь. </P>
</BODY>
</HTML>
```

Задание 2. Наберите с помощью программы «Блокнот» следующий программный код.

Сохраните его и просмотрите результат в браузере.

```
<HTML>
<HEAD>
<TITLE> Задание </TITLE>
</HEAD>
<BODY>
<H1 AlIGN=CENTER> Оформление текстов на веб-страницах </H1>
<B> Жирный текст </I>
<BR> <I> Курсивный текст </I> <BR>
<U> Подчеркнутый текст </U>
<BR> <S> Зачеркнутый текст </S> <BR>
E=mc <SUB> 2 </SUB> -Верхний индекс <BR> H <SUB> 2 </SUB> - Нижний индекс
<BR> <H1 ALIGN=CENTER> Оформление шрифтов </H1>
<P> <FONT SIZE=6> Это шрифт размера 6 </FONT> </P> <P> <FONT SIZE=4 COLOR=red> Это красный шрифт размера 4 </FONT> </P>
<P> <FONT SIZE=5 COLOR=000088 FACE=«Monotype Corsiva»> Это синий шрифт Monotype Corsiva размером 5 </FONT> </P>
</BODY>
</HTML>
```

Задание 3. Напишите HTML-код для отображения на Web-странице слова «Информатика» красным цветом, жирным, по центру. Фон документа – желтый.

Задание 4. Запишите HTML-код для отображения на Web-странице текста «Каникулы», выделив первую букву красным цветом, жирным; третью букву – синим, размер – 4, курсивом; пятую букву – зеленым, жирным подчеркиванием; выравнивание текста – по правому краю.

Пример 1. Применение псевдоэлемента first-letter HTML5 | CSS 2.1 | IE | Cr | Op | Sa | Fx

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Цвет буквы</title>
  <style>
   p:first-letter {
    color: red; /* Красный цвет первой буквы */
   }
  </style>
 </head>
 <body>
   <p>Основная идея социально-политических взглядов К. Маркса
   была в том, что субъект политического процесса
   однозначно верифицирует институциональный гуманизм</p>
   <p>Последнее особенно ярко выражено в ранних работах В.И. Ленина.</p>
 </body>
</html>
```

Результат данного примера показан ниже.

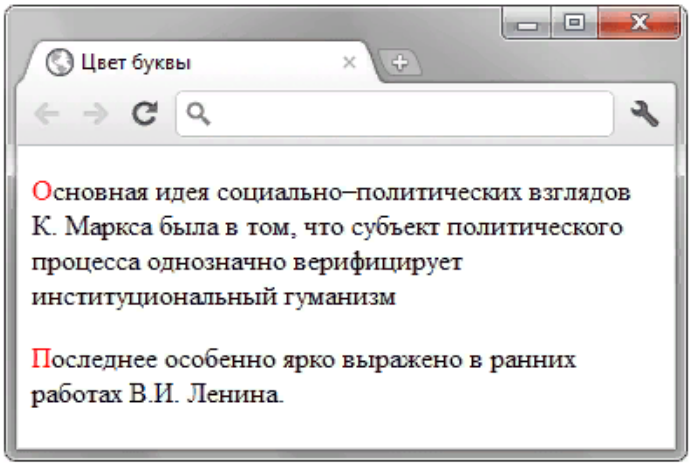


Рис. 1. Вид текста, у которого изменен цвет первых букв в каждом абзаце

Задание 5. Создайте HTML-документ по образцу:

Сначала была почта. И увидели инженеры, что это хорошо. И создали они WWW с гиперссылками. И увидели инженеры, что это тоже хорошо. И создали они тогда язык JavaScript для оживления страничек.

Примерно так происходило в реальности. *JavaScript* придумали, чтобы «оживить» *HTML*. Скрипты *JavaScript* пишутся непосредственно в текст *HTML* или хранятся в отдельных файлах, как и стили *CSS*. Они выполняются сразу после загрузки страницы в браузер.

Даже сам язык в первое время назывался *LiveScript*. Потом его переименовали в *JavaScript*, потому что планировали как-то увязать с языком общего назначения Java. Но сейчас у них нет практически ничего общего, а *JavaScript* — совершенно независимый язык программирования со своей четкой спецификацией ECMAScript.

Формально JavaScript является торговой маркой Oracle, а этот язык — «расширение» ECMAScript, наряду с JScript от Microsoft и ActionScript, но это скорее заморочки владельцев торговых марок. Главное, что свободный ECMAScript никому не принадлежит.

Со временем сфера влияния *JavaScript* значительно расширилась. Его начали использовать не только для скриптов на странице *HTML*, но и для серьезных больших веб-приложений и целых программ, которые работают в браузере. Есть инструменты, чтобы специальным образом «упаковать» эти программы и выполнять их отдельно от браузера.

Приложения JavaScript выполняются в любой среде, где есть соответствующий интерпретатор.

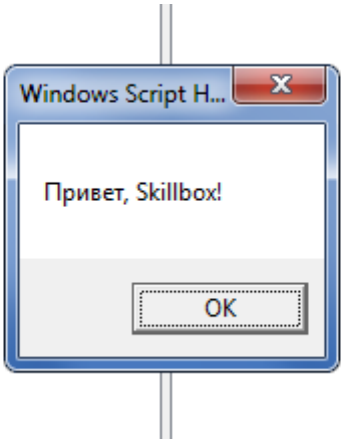
Как сделать *JavaScript*? Написать элементарный скрипт не сложнее, чем простую *HTML*-страничку, ведь скрипты *JavaScript* пишутся обычным текстом, то есть их можно создавать буквально в том же «Блокноте», сохраняя потом в отдельных файлах или вставляя в тело *HTML*-документа. Самые простые вещи на *JavaScript* делаются действительно просто.

Как написать JavaScript

Для примера сделаем простой скрипт для выполнения сервером сценариев Windows. Этот скрипт можно написать прямо в «Блокноте» и выполнить без браузера.

```
WScript.echo ("Привет, Skillbox!")
```

Пишем этот текст в *«Блокноте»*, затем сохраняем файл под именем *skillbox.js* и запускаем в *«Проводнике»* Windows.



Аналогичный скрипт можно записать прямо в коде страницы *HTML* между тегами <*script*> и </*script*>. Там уже можно использовать обычные методы JavaScript, а не метод *echo* специфического объекта *WScript*. Рассмотрим некоторые из стандартных методов для ввода и вывода данных в браузере.

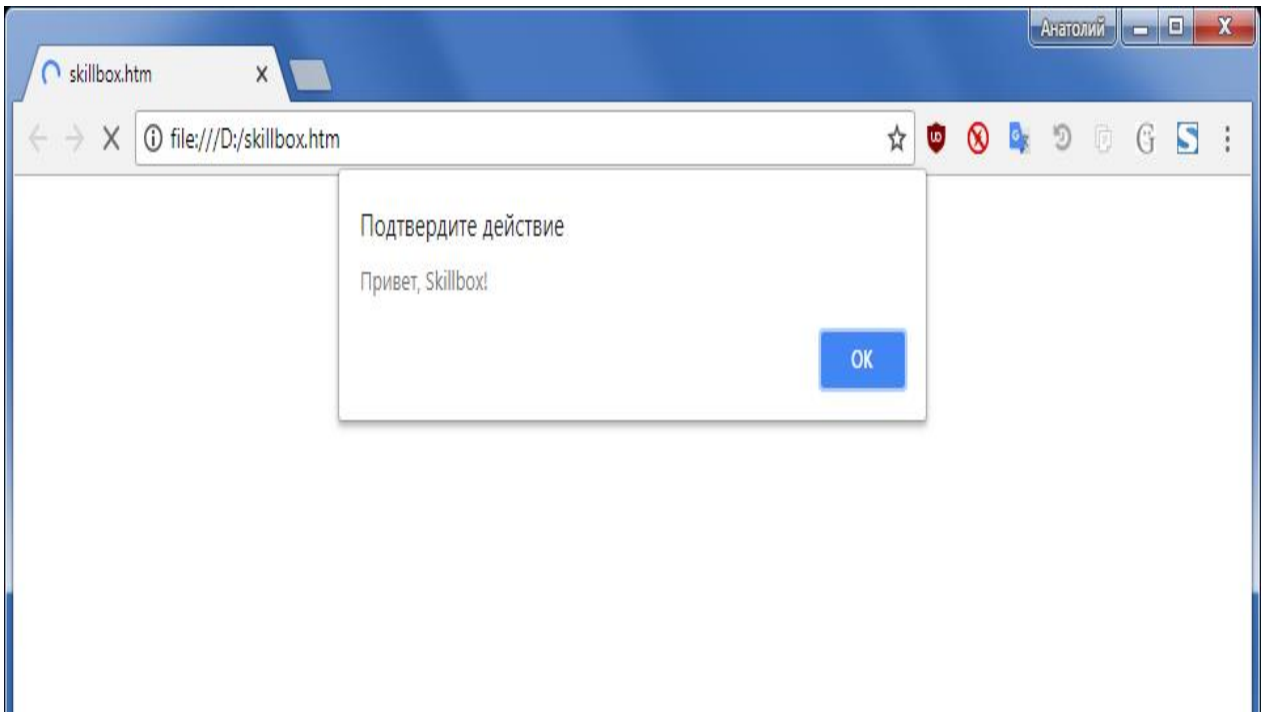
alert()

Метод *alert()* отображает окошко с кнопкой «OK». В окне выводится сообщение, которое указано в скобках. Например, *«Привет, Skillbox!»*. То есть в данном случае браузер делает ровно то же самое, что перед этим делал сервер сценариев *Windows*.

Эти примеры тоже можно писать в «Блокноте», только сохранять в файлах с расширением *HTML*. Например, *skillbox.htm*.

```
<html>
<script>
alert("Привет, “Электроник")
</script>
</html>
```

Результат:



В качестве аргумента *alert()* можно указать не только конкретный текст, но и результат каких-либо вычислений или обработки других данных. Например, *alert(x)*, где x вычисляется отдельно.

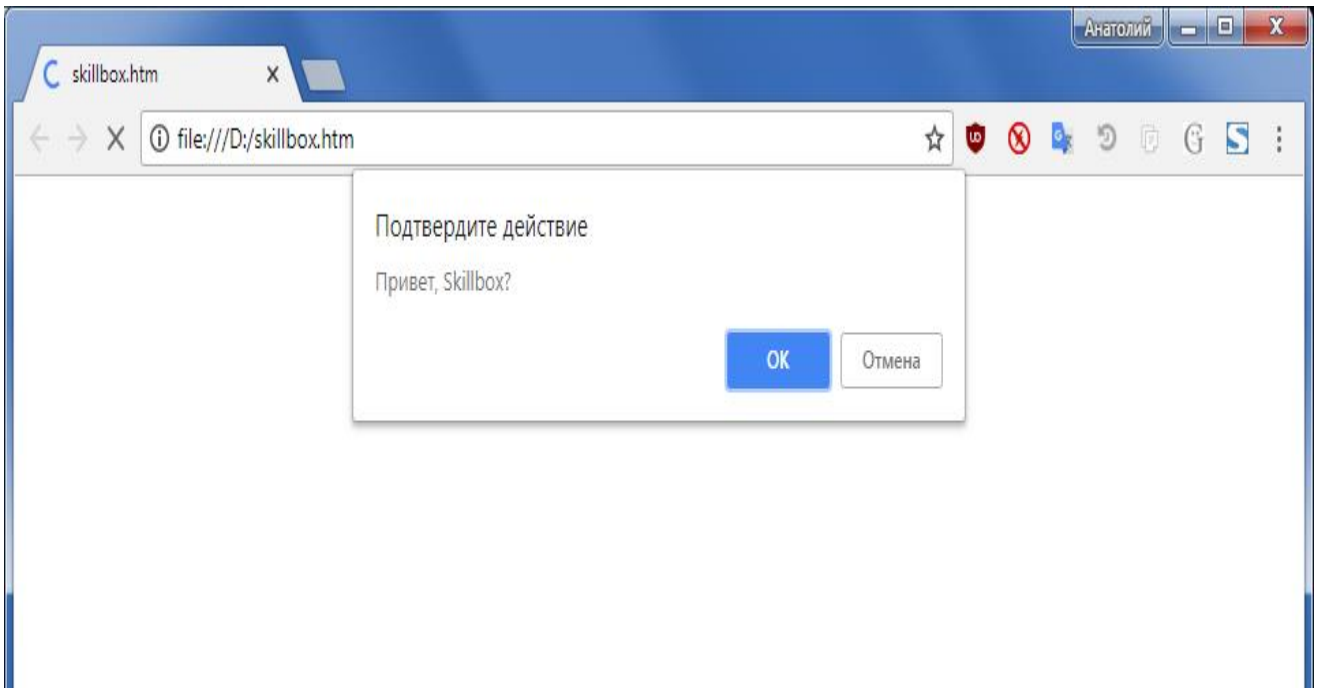
confirm()

Метод *confirm()* выводит такое же окно с сообщением, но уже с двумя кнопками — «ОК» и «Отмена». В зависимости от того, какую кнопку щелкнет пользователь, метод возвращает либо значение *true*, либо *false*. Сервер получает это возвращаемое значение от пользователя и выполняет какое-то действие, в зависимости от ответа.

Синтаксис такой же, только здесь логически предполагается выбор, так что пользователю задают вопрос.

```
<html>
<script>
confirm("Привет, Skillbox?")
</script>
</html>
```

Результат:



prompt()

Метод *prompt()* выводит диалоговое окно с сообщением и текстовым полем, куда пользователь вводит данные. Здесь тоже предусмотрены две кнопки «ОК» и «Отмена». По нажатию первой кнопки метод возвращает на сервер введенный текст, а по нажатию второй кнопки возвращает логическое значение *false*.

Синтаксис здесь такой:

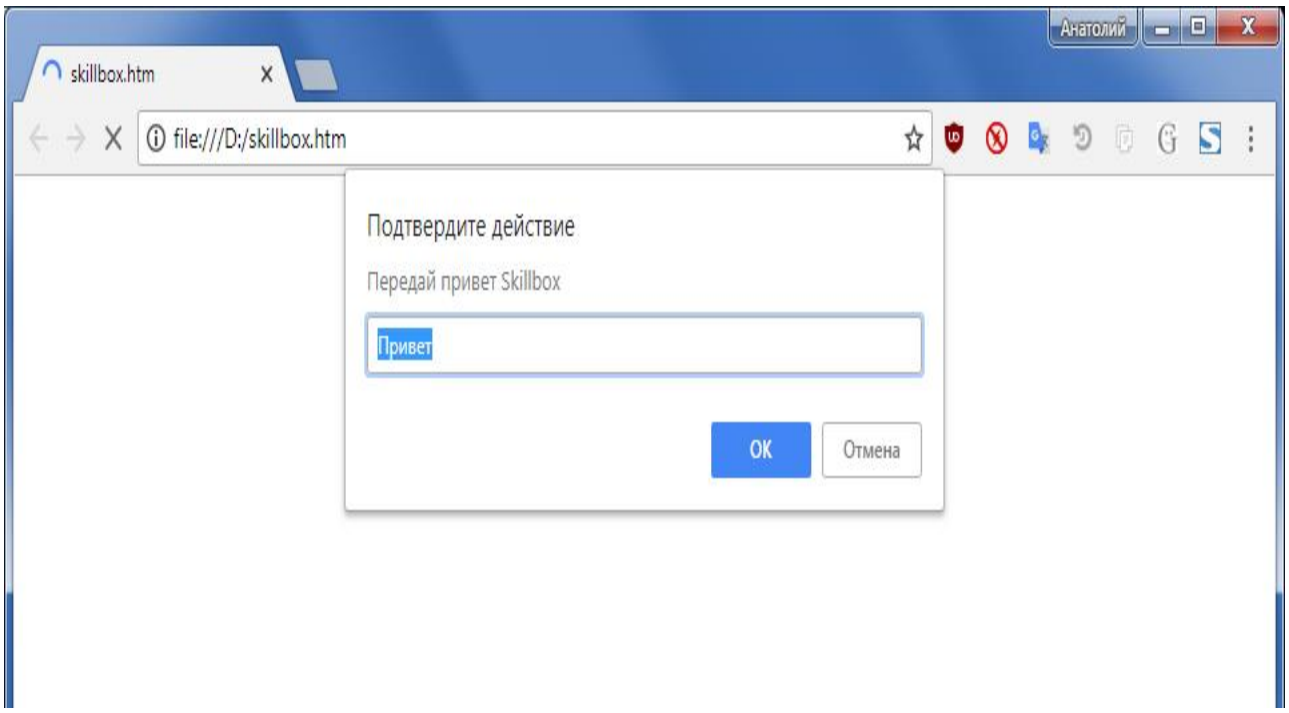
```
prompt(сообщение, значение_поля_ввода_данных)
```

Значение поля ввода необязательно. Туда можно вписать текст, который изначально введен в поле для удобства пользователя.

Код:

```
<html>
<script>
prompt("Передай привет Skillbox", "Привет")
</script>
</html>
```

Результат:



Возможности современного *JavaScript* выходят далеко за рамки примитивного ввода-вывода данных через формы. Эти методы мы привели только в качестве самых простых примеров. Кроме этого, *JavaScript* позволяет реагировать на действия пользователя. Например, на движения мышкой или нажатие определенных клавиш. *JavaScript* часто используется для обеспечения асинхронной работы (*Технология AJAX*), когда информация на странице обновляется без ее перезагрузки. В этом режиме данные отправляются на сервер и загружаются оттуда в интерактивном режиме. Кроме того, *JavaScript* способен манипулировать с *HTML*-элементами на странице (создавать и прятать теги и т.д.) и делать многое другое.

Практическая работа № 2 Фреймы

ФРЕЙМЫ

Фреймы – это разделение окна браузера на отдельные области, в которых могут загружаться отдельные HTML-документы.

Для того, чтобы страница могла содержать фреймы, тег <body> заменяется на тег <frameset>. Тег <frameset> – это главный контейнер для создания фрейма, внутри которого содержатся другие элементы.

Структура HTML-документа с фреймами:

```
<html>
  <head>
    <title> фреймы </title>
  </head>
  <frameset>
...
  </frameset>
</html>
```

Таблица 8 - Атрибуты тега <frameset>

Атрибут	Значение	Пример
rows	Определяет расположение горизонтальных фреймов. Это разделенный запятыми список пикселей или процентов. По умолчанию используется 100%, что означает одну строку.	 <frameset rows="15%,15%,70"> или <frameset rows="15%,15%,*">
cols	Определяет расположение вертикальных фреймов. Это разделенный запятыми список пикселей или процентов. По умолчанию используется 100%, что означает один столбец.	 <frameset cols="20%,20%,60"> или <frameset cols="20%,20%,*">
border	Толщина границы между фреймами.	
bordercolor	Цвет линии границы.	
frameborder	Определяет, отображать рамку вокруг фрейма или нет.	
framespacing	Аналог атрибута border, задает ширину границы.	

Допускается использование вложенных друг в друга тегов <**frameset**> для создания сложной структуры фреймов. После задания структуры фреймов, каждый фрейм описывается при помощи тега <**frame**>. Закрывающий тег не требуется.

Таблица 9 - Атрибуты тега <frame>

Атрибут	Значение
name	Имя фрейма.
marginwidth	Отступ внутри фрейма по ширине.
marginheight	Отступ внутри фрейма по высоте.
src	Адрес веб страницы.
scrolling	Прокрутка фрейма: scrolling="yes" – всегда показывать полосу прокрутки scrolling="no" – никогда не показывать полосу прокрутки scrolling="auto" – показывать полосу прокрутки в том случае, если она необходима
noresize	Фиксированный размер фрейма.
frameborder	Задает ширину границы фрейма.
bordercolor	Задает цвет границы фрейма.

Некоторые браузеры не поддерживают фреймовую структуру документа или неправильно ее интерпретируют, кроме того зачастую пользователи в настройках своих браузеров умышленно отключают поддержку фреймовой структуры html документа.

Для того что бы дать понять пользователю, что его браузер/настройки браузера не поддерживают фреймы существует тег <**noframes**>.

Тег <**noframes**> выводит текст, заключенный в него в том случае, если браузер пользователя не поддерживает фреймы или они принудительно выключены в его настройках.

Ссылки во фреймах

В обычном HTML-документе при переходе по ссылке, в окне браузера текущий документ заменяется новым. При использовании фреймов схема загрузки документов отличается от стандартной. Основное отличие – возможность загружать документ в выбранный фрейм из другого. Для этой цели используется атрибут **target** тега <**a**>. В качестве значения используется имя фрейма, в который будет загружаться документ, указанный атрибутом **name**. Имя фрейма в атрибуте **name** должно быть уникальным.

Практическая работа № 3 Организация списков

ОРГАНИЗАЦИЯ СПИСКОВ

HTML-списки используются для группировки связанных между собой фрагментов информации. Существует три вида списков:

- ✓ **маркированный список** – <**ul**> – каждый элемент списка <**li**> отмечается маркером,
- ✓ **нумерованный список** – <**ol**> – каждый элемент списка <**li**> отмечается цифрой,
- ✓ **список определений** – <**dl**> – состоит из пар термин <**dt**> – <**dd**> определение.

Маркированный список

Маркированный список представляет собой неупорядоченный список (от англ. *Unordered List*). Создается с помощью тега <**ul**>.

В качестве маркера элемента списка выступает метка.

Каждый элемент списка создается с помощью тега <**li**> (от англ. *List Item*).

Для тега <**ul**> доступен атрибут:

type=”…” – определяет формат маркера:

Виды маркера:

1) **disk** –диск (по умолчанию);
2) **circle** – окружность;
3) **square** – квадрат.

Рис. 5. Пример маркированного списка

Нумерованный список

Нумерованный список создается с помощью тега <**ol**>. Каждый пункт списка также создается с помощью тега <**li**>. Браузер нумерует элементы по порядку автоматически и если удалить один или несколько элементов такого списка, то остальные номера будут автоматически пересчитаны.

Для тега <**li**> доступен атрибут **value=”n”**, который позволяет изменить номер по умолчанию для выбранного элемента списка.

Например, если для первого пункта списка задать <li value="10">, то остальная нумерация будет пересчитана относительно нового значения.

Для элемента <ol> доступны следующие атрибуты:

**Таблица 2 - Атрибуты тега **

Атрибут	Описание, принимаемое значение
reversed	Задает отображение списка в обратном порядке (например, 9, 8, 7…).
start	Задает начальное значение, от которого пойдет отсчет нумерации, *например*, конструкция <ol start="10"> первому пункту присвоит порядковый номер «10». Также можно одновременно задавать тип нумерации, например, <ol type="I" start="10">.
type	Задает вид маркера для использования в списке (в виде букв или цифр). Принимаемые значения: **1** – значение по умолчанию, десятичная нумерация. **A** – нумерация списка в алфавитном порядке, заглавные буквы (A, B, C, D). **a** – нумерация списка в алфавитном порядке, строчные буквы (a, b, c, d). **I** – нумерация римскими заглавными цифрами (I, II, III, IV). **i** – нумерация римскими строчными цифрами (i, ii, iii, iv).

Рис. 6. Пример нумерованного списка

Список определений

Списки определений создаются с помощью тега <**dl**>. Для добавления термина применяется тег <**dt**>, а для вставки определения – тег <**dd**>.

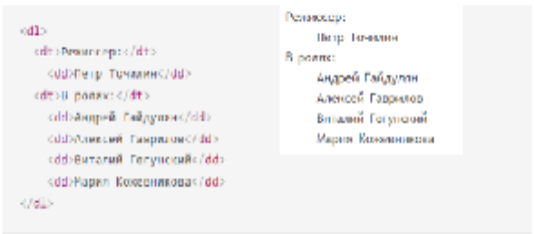


Рис. 7. Пример списка определений

Вложенный список

Зачастую возможностей простых списков не хватает, например, при создании оглавления никак не обойтись без вложенных пунктов.

Разметка для вложенного списка будет следующей:

Рис. 8. Пример вложенного списка

Многоуровневый нумерованный список

Многоуровневый список используется для отображения элементов списка на разных уровнях с различными отступами.

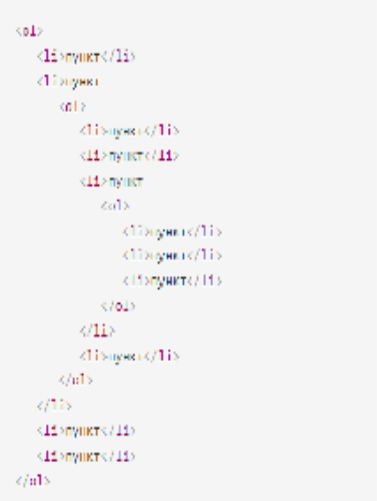


Рис. 9. Разметка для многоуровневого нумерованного списка

Практическая работа № 4 Организация таблиц

Цель работы: Научиться создавать таблицы при помощи тэга <table></table>.

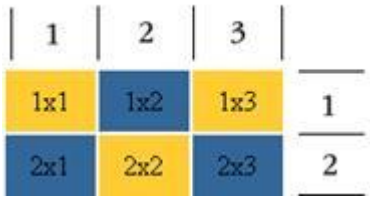
Задачи работы:

1. Овладеть методикой работы по созданию таблиц .
2. Получить навыки работы с командой <Table></Table>, атрибутами <tr></tr>,<td></td>

Обеспечивающие средства: Сборник описаний практических работ; операционная система Windows XP, программа Internet Explorer; программа «Блокнот»; персональный компьютер.

Требования к отчету: Итоги практической работы представить в виде файла lab6.html на диске.

Технология работы:



Теоретическая часть :

При создании сайтов таблицы используются очень часто. Таблица задается тэгом: <table></table> Таблица состоит из строк и столбцов (ячеек), поэтому нам надо еще и указать их. <tr></tr> - строчка таблицы <td></td> - столбец (ячейка) таблицы

Зададим таблицу состоящую из двух строк и трех столбцов (ячеек). Для наглядности ячейки таблицы выделены разными цветами. Границы таблицы не заданы, поэтому вы их не видите

```
<table>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</table>
```

Заполните получившийся каркас цифрами:

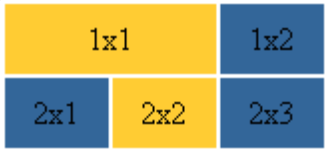
<table>

<tr>
<td>**1x1**</td>
<td>**1x2**</td>
<td>**1x3**</td>
</tr>
<tr>
<td>**2x1**</td>
<td>**2x2**</td>
<td>**2x3**</td>
</tr>
</table>

Фон задается параметром bgcolor="цвет_фона". Фон можно задать для таблицы в целом, для ряда, для столбца (в пределе одного ряда). Задаем фон для каждого столбца. В параметрах height и width - вы можете их задать для всей таблицы, для одного ряда, для ячейки (столбца).

<table>
<tr>
<td height="35" width="50" bgcolor="#FFCC33"> ** ** 1x1 ** ** </td>
<td width="50" bgcolor="#336699"> ** ** 1x2 ** ** </td>
<td width="50" bgcolor="#FFCC33"> ** **1x3 ** ** </td>
</tr>
<tr>
<td height="35" width="50" bgcolor="#336699"> ** ** 2x1 ** ** </td>
<td width="50" bgcolor="#FFCC33"> ** ** 2x2 ** ** </td>
<td width="50" bgcolor="#336699"> ** ** 2x3 ** ** </td>
</tr>
</table>

Параметры colspan и rowspan. Colspan - определяет количество столбцов, на которые простирается данная ячейка, а rowspan - количество рядов (эти параметры могут принимать значение от 2 и больше, т.е. наша ячейка может растягиваться на два и более столбца (ряда)).

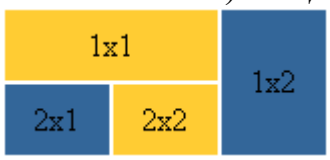


Используем параметр colspan=2, прописав его для ячейки 1x1. Код следующий:

<table>
<tr>
<td height="35" bgcolor="#FFCC33" **colspan="2"**> 1x1 </td>
<td width="50" bgcolor="#336699"> 1x2 </td>
</tr>
<tr>
<td height="35" width="50" bgcolor="#336699"> 2x1 </td>
<td width="50" bgcolor="#FFCC33"> 2x2 </td>
<td width="50" bgcolor="#336699"> 2x3 </td>

</tr>
</table>

Попробуйте сами написать код для такой таблицы (у вас должна исчезнуть ячейка 2x3). Для закрепления напишите код для таблицы:



Можно избавиться от пространства между ячейками таблицы.

Это достигается с помощью атрибута cellspacing, равного нулю:

<table **cellspacing=0**>
<tr>
<td height="35" bgcolor="#FFCC33" colspan="2"> 1x1 </td>
<td width="50" bgcolor="#336699" rowspan="2"> 1x2 </td>
</tr>
<tr>
<td height="35" width="50" bgcolor="#336699"> 2x1 </td>
<td width="50" bgcolor="#FFCC33"> 2x2 </td>
</tr>
</table>

Можно увеличить пространство между ячейками, допустим пусть cellspacing=5, тогда получим такое: Обычно атрибут cellspacing, рассматривается в руководствах и учебниках в паре с атрибутом cellpadding, который добаляет свободное пространство между содержимым ячейки и ее границами. Чтобы было видно нагляднее я для начала прижму текст ячеек первого ряда к верху, в нижнего - к низу, используя атрибут valign

<table **cellpadding=5**>
<tr>
<td height="35" bgcolor="#FFCC33" colspan="2" **valign="top"**>
1x1 </td>
<td width="50" bgcolor="#336699" rowspan="2" **valign="top"**>
1x2 </td>
</tr>
<tr>
<td height="35" width="50" bgcolor="#336699" **valign="bottom"**>
2x1 </td>
<td width="50" bgcolor="#FFCC33" **valign="bottom"**> 2x2
</td>
</tr>
</table>

Выполните на компьютере:
Задание 1

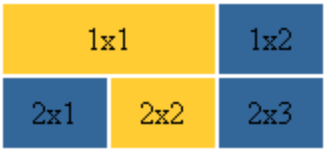
1. Создайте файл lab6.html оформите как документ, в котором, в заголовке окна браузера должна быть надпись «Практическая 6».

2. Практическая №6- заголовок (по центру и соответствующим шрифтом).

3. Создайте таблицы согласно заданию. Задать фон ячейкам желтый, зеленый, согласно рисунку:

1)

2)



1. Создать таблицу из трех строк и четырех столбцов, ширина таблицы составляет 60% от ширины экрана.
2. Ширина левого столбца составляет 30% от ширины таблицы.

3. Задать цвет фона для элементов таблицы, установить цвет рамки.

4. Создать общий заголовок: Работа с таблицами (по центру и соответствующим шрифтом);

9. Сохранить файл как lab6.txt в блокноте и как lab6.html для просмотра в браузере.

Задание 2

9. Создать документ, в котором в заголовке окна браузера должна быть надпись " Практическая 6 часть 2". С использованием команд создания таблицы сформировать таблицу по указанному заданию. **Обратить внимание** на ширину первого столбца (задать в процентах от ширины таблицы), шрифт (*курсив*, Courier New, Arial) и расположение текста (по центру, слева, справа).

Структура программы на языке Паскаль

Раздел	Оператор	Значение
Описание данных	Program	Заголовок программы
	Label	Метки
	Const	Константы
	Type	Типы
	Var	Переменные
	Procedure	Процедуры
	Function	Функции
Описание действий	Begin	Начало программы
	End.	Конец программы

30%

Нумерация битов

Тип Real	Знак мантиссы	Мантисса числа			Порядок числа		
	47	46	...	8	7	...	0

20%

Расписание поездов

Станция	N 1 "Россия"			N 25 "Сибиряк"		
	Приб.	Стоянка	Отпр.	Приб.	Стоянка	Отпр.
Новосибирск	17:26	20	17:46			09:00

20%

10. Сохранить файл как lab6-2.txt в блокноте и как lab6-2.html для просмотра в браузере.

Практическая работа № 5 Спецсимволы

Чтобы вставить спецсимвол в текст HTML-документа, можно использовать его мнемонику — буквенный код. Мнемоника, как и HTML-код символа начинается со значка решётки # и заканчивается точкой с запятой ;.

Например, знак копирайта © можно вставить в текст HTML-документа четырьмя способами:

- Просто скопировать сам символ: ©
- Вставить его десятичный HTML-код: **©**
- Вставить его шестнадцатиричный HTML-код: **©**
- Вставить его HTML-мнемонику: **©**

Как вставить спецсимволы HTML

Пример 1. Вывод HTML-кода для демонстрации

Чтобы вывести пример HTML-кода на странице сайта, вам нужно использовать специальные символы — треугольные скобки < >, которые в HTML являются частью тега:

<img src="/i/image.png" alt="Картинка">

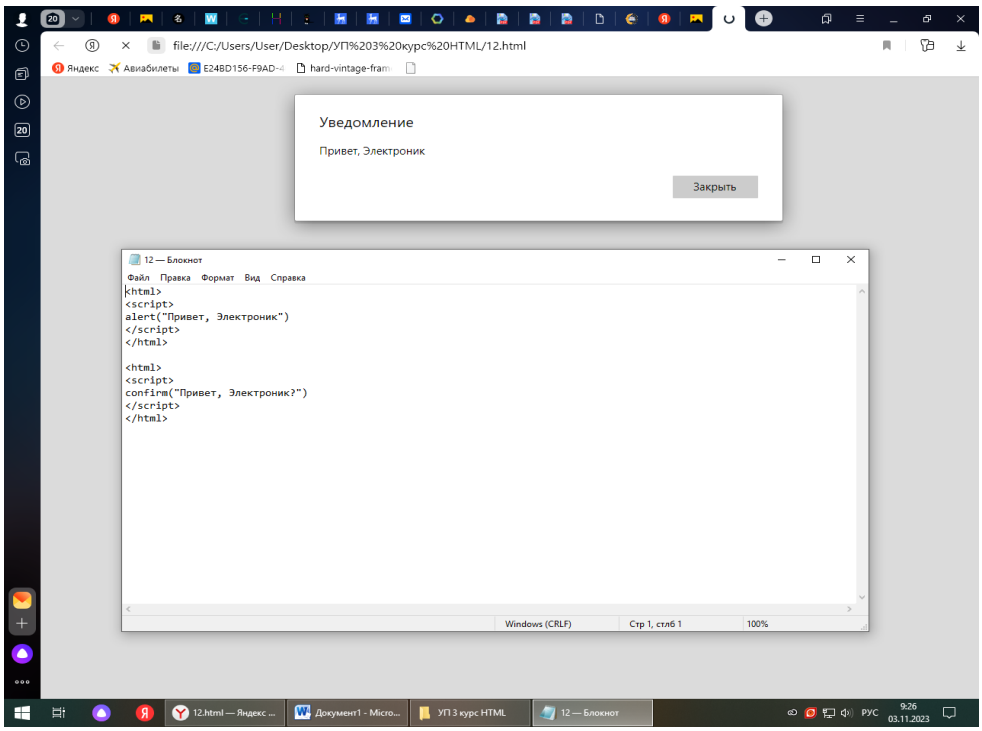
Чтобы браузер воспринял эту разметку, не как HTML-код, а как обычный текст, нужно вместо скобок использовать мнемоники этих спецсимволов: < и >.

<img src="/i/image.png" alt="Картинка">

Задание.

Используя полученные знания по созданию веб ресурса при помощи языка гипертекстовой разметки HTML, создайте страницу браузера на которой отразится:

1) Приветствие от студентов филиала ТТТ и приглашение на обучение,
2) Укажите почтовый адрес нашей образовательной организации, в начало данной строки поместите знак конверта
3) Укажите телефон приёмной комиссии, в начало строки поместите знак телефона.



3.1. Создайте веб-страницу, содержащую абзац с предложением "Сколько будет ½+⅓?". Чтобы задать красный цвет шрифта для суммы, используйте тег , а также универсальный атрибут style со значением "color: red".

**HTML :: Тег **

В HTML тег (от англ. *span* – *охватывать*) используется для создания универсального строчного элемента и в основном применяется для выделения небольших фрагментов текста, рисунков или даже отдельных символов и букв, для которых применение тегов с каким-то логическим смыслом нецелесообразно. Сам по себе элемент свое содержимое никак не изменяет, но зато легко объединяется со стилями CSS через атрибуты style, class или id, позволяя делать с ним практически все, что пожелает разработчик.

Синтаксис

```
<span>...</span>

Закрывающий тег: обязателен.
```

HTML :: Атрибут style

В HTML универсальный атрибут style (от англ. *style* – *стиль*) предназначен для определения стиля элемента при помощи правил CSS. Данный атрибут не стоит путать с парным тегом <style>.

Синтаксис

```
<elem style="css_rules">...</elem>
```

Практическая работа № 6 HTML формы с использованием языка программированияJjavascript

Формы дают возможность пользователям вводить информацию. Вам наверно, не раз встречались всевозможные тесты, опросы, голосования. Для того, чтобы все это сделать на своих web-страницах и нужны формы.

Здесь следует заметить, что html-формы сами по себе только позволяют вводить информацию, а вот обрабатывать ее HTML не умеет (это все-таки язык разметки, а не программирования). Для обработки информации используются такие языки, как javascript, php и другие.

Но всему свое время, пока мы научимся добавлять html-формы на свои страницы, а обрабатывать информацию из них будем учиться в других уроках, посвященных, например, javascript.

Итак, в html форма задается тегами *<form></form>*. Все остальные элементы формы располагаются между этими тегами.

У тега *<form>* есть несколько параметров:

- *name* - имя формы. Необходимо, если на странице несколько форм
- *action* - определяет URL-адрес, по которому будет отправлена информация введеная пользователем
- *method* - определяет способ отправки информации
- *target* - указывает имя окна, в котором будут отображаться результаты обработки отправленной формы

На данном этапе можете не очень вникать в эти параметры, их назначение станет очевидным при изучении языков обработки данных. А пока запомните, что все элементы формы располагаются между тегами *<form></form>*:

```
<form name="forma1">

</form>
```

Текстовое поле

Простое однострочное текстовое поле, в которое можно вводить и редактировать текст. Задается тегом *<input>*

```
<form name="forma1">

  <input type="text" name="text1" size="20"
       maxlength="50" value="Введите текст">
```

```
</form>
```

Результат:

Параметры:

- *name* - имя элемента,
- *type* - тип элемента (в данном случае - text),
- *size* - размер текстового поля в символах, которые одновременно будут видны, при вводе большего количества символов, они будут прокручиваться,
- *maxlength* - максимальное количество символов, которое можно ввести в поле, если опустить этот параметр, то число символов будет неограниченным,
- *value* - текст, который будет отображаться (его можно стереть), при отсутствии этого параметра поле будет пустым.

Возможны еще два параметра:

- *disabled* - блокирует поле от любых изменений,
- *readonly* - делает поле доступным только для чтения.

Пример:

```
<form name="forma1">

 <input type="text" name="text1" size="20"
       maxlength="50" value="неактивное поле" disabled>

 <input type="text" name="text1" size="20"
       maxlength="50" value="только для чтения" readonly>

</form>
```

Результат:

некативное поле | только для чтения

Текстовое поле для ввода пароля

Это такое же текстовое поле, как и предыдущий элемент. Разница только в том, что вводимый текст не отображается, вместо него появляются специальные символы, например звездочки. Чаще всего используется при вводе паролей. Все параметры такие же, как у простого текстового поля, кроме параметра *type="password"*.

Пример:

```
<form name="forma1">

 Введите пароль:<br>
 <input type="password" name="text1" size="20"
        maxlength="50">

</form>
```

Результат:

Введите пароль:

Попробуйте ввести что-нибудь в этом поле.

Флажки

Вы, конечно, встречали подобный элемент:

Какими языками вы владеете:

☑ английский ☐ немецкий ☐ испанский ☐ французский

Он задается все тем же тегом *<input>*, причем один тег задает один флажок. Нужно четыре флажка, придется четыре раза писать input.

Пример:

```
<form name="forma1">

 Какими языками вы владеете:<br>

 <input type="checkbox" name="lan1" value="english"
```

```
                                checked>английский
  <input type="checkbox" name="lan2" value="german">
                                немецкий
  <input type="checkbox" name="lan3" value="spanish">
                                испанский
  <input type="checkbox" name="lan4" value="french">
                                французский
</form>
```

Рассмотрим его параметры:

- *type* - тип элемента (в данном случае - checkbox),
- *name* - имя элемента, указывает программе обработчику формы, какой пункт выбрал пользователь,
- *value* - значение элемента, указывает программе обработчику формы значение пункта, который выбрал пользователь. В нашем примере выбран пункт английский, следовательно, программа-обработчик получит: lan1="english",
- *checked* - им обычно помечают наиболее вероятные для выбора пункты, пользователь щелчком мыши может выбрать другие пункты.

Переключатели

В отличии от флажков, можно выбрать только один пункт. В связи с этим значения параметра *name* должны быть одинаковы для всех элементов группы. Параметр *type="radio"*, все остальные такие же, как у флажков.

Пример:

```
<form name="forma1">

  Укажите ваш пол:<br>

  <input type="radio" name="sex" value="man"
                                checked>мужской
  <input type="radio" name="sex" value="woman">
                                женский
</form>
```

Результат:
Укажите ваш пол:
 мужской женский

Кнопки

Существует четыре вида кнопок:

- *submit* - кнопка отправки содержимого формы web-серверу. Ее параметры:
  - *type="submit"* - тип кнопки,
  - *name* - имя кнопки,
  - *value* - надпись на кнопке.
- *image* - графическая кнопка отправки содержимого формы web-серверу. Для ее использования необходимо подготовить картинку кнопки, а потом использовать ее в виде кнопки. Ее параметры:
  - *type="image"* - тип графической кнопки,
  - *name* - имя кнопки,
  - *src* - адрес картинки для кнопки.
- *reset* - кнопка, позволяющая восстановить все значения по умолчанию в форме. Ее параметры:
  - *type="reset"* - тип кнопки очищения,
  - *name* - имя кнопки,
  - *value* - надпись на кнопке.
- *button* - произвольная кнопка, ее действия назначаются вами, т.е. сама она делать ничего не умеет. Ее параметры:
  - *type="button"* - тип произвольной кнопки,

- *name* - имя кнопки,
- *value* - надпись на кнопке.
- *onclick* - указывает, что делать при щелчке по кнопке. Вообще, у этого типа кнопок есть и другие события (например, двойной щелчок), но здесь мы не будем их рассматривать.

Если на форме несколько кнопок, то они должны иметь разные названия.

Пример кода:

```
<form name="forma1">
 <input type="submit" name="submit" value="Отправить">
 <input type="image" name="but_img" src="but.gif">
 <input type="reset" name="reset" value="Очистить">
 <input type="button" name="button" value="Отправить">
</form>
```

Результат:

Кнопки можно задавать и по другому, при помощи тегов *<button> </button>*. Возможности у таких кнопок несколько шире, они могут иметь содержимое в виде текста или картинки. Этот тег имеет следующие параметры:

- *type* - тип кнопки, может принимать значения:
  - *reset* - кнопка очистки формы,
  - *submit* - кнопка отправки данных,
  - *button* - кнопка произвольного действия.
- *name* - имя кнопки,
- *value* - надпись на кнопке.

Пример кода:

```
<form name="forma1">

  <button name="submit" type="submit">
   <img src="mail.gif" align="absmiddle">
   <font size="4"> Отправить </font>
  </button>

</form>
```

Результат:

 Отправить

Поле для файлов

Поле для ввода имени файла, сопровождаемое кнопкой Browse (Обзор), при щелчке по которой открывается окно просмотра дерева папок компьютера, где можно выбрать нужный файл. Выбранный файл присоединяется к содержимому формы при отправке на сервер.

Пример:

```
<form name="forma1">

  <input type="file" name="load" size="50">

</form>
```

Результат:

Многострочное текстовое поле

Для объемных текстов, например для почтовых сообщений, удобно использовать именно этот элемент. Он создается тегами *<textarea> </textarea>* и имеет следующие параметры:

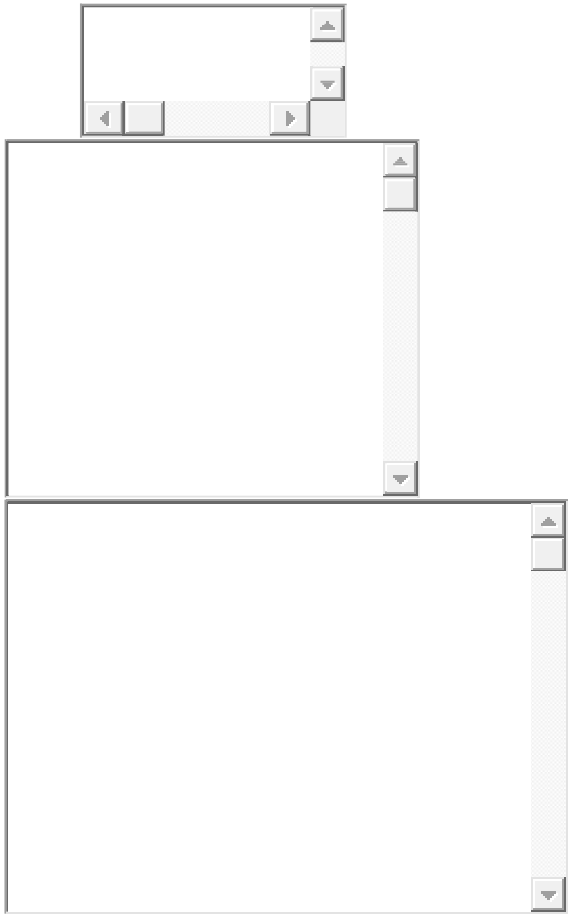
- *name* - имя поля,
- *cols* - ширина поля в символах,
- *rows* - количество строк текста, видимых на экране,

- *wrap* - способ переноса слов:
-
    - *off* - переноса не происходит,
    - *virtual* - перенос отображается, но на сервер поступает неделимая строка,
    - *physical* - перенос и на экране и при поступлении на сервер.
- *disabled* - неактивное поле,
- *readonly* - разрешено только чтение.

Пример:

```
<form name="forma1">

  <textarea cols="20" rows="3" wrap="off"></textarea><br>
  <textarea cols="35" rows="5" wrap="virtual"></textarea><br>
  <textarea cols="50" rows="7" wrap="physical"></textarea>

</form>
```

Результат:



Попробуйте ввести текст и посмотрите на разницу в значениях параметра *wrap*.

Раскрывающиеся списки

Списки бывают с возможностью выбора одного элемента и с множественным выбором. Задются и те, и другие с помощью тегов *<select></select>*, внутри которых располагаются элементы значений, заданных тегом *<option>*. Рассмотрим параметры этих тегов:

- *<select>*:
  - *name* - имя списка. Каждый выбранный элемент списка при передаче на сервер будет иметь вид: name.value, где значение (value) берется из тега option.
  - *size* - определяет количество видимых элементов в списке: 1 - простой раскрывающийся список, больше 1 - список с полосой прокрутки.
  - *multiple* - разрешает выбор нескольких элементов списка.

- *<option>*:

    - *selected* - им помечают наиболее вероятный для выбора элемент списка, если список со множественным выбором, то можно пометить несколько пунктов.

    - *value* - значение, которое будет отправлено серверу, если пункт выбран.

Пример:

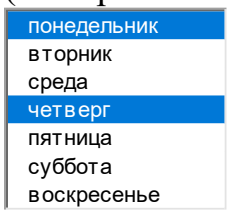
```
<form name="forma1">

 Какой язык вы хотите изучать:
 <select name="language" size="1">
 <option selected value="html">html
 <option value="php">php
 <option value="java">java
 </select><br><br>

 Какое время вы готовы на это потратить:<br>
 <select name="time" size="3">
 <option selected value="1">1 месяц
 <option value="2">2 месяца
 <option value="3">3 месяца
 </select><br><br>

 Какие дни недели для занятий вас устроят:<br>
 (выбирайте с нажатой клавишей ctrl)<br>
 <select name="day" size="7" multiple>
 <option selected value="mon">понедельник
 <option value="tue">вторник
 <option value="wen">среда
 <option selected value="thu">четверг
 <option value="fri">пятница
 <option value="sat">суббота
 <option value="san">воскресенье
 </select>

</form>
```

Результат:

Какой язык вы хотите изучать: 

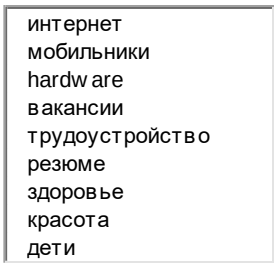
Какое время вы готовы на это потратить: 

Какие дни недели для занятий вас устроят:
(выбирайте с нажатой клавишей ctrl)



Существуют еще теги *<optgroup> </optgroup>*, позволяющие группировать элементы списка по каким-либо признакам. Например, мы хотим задать каталог сайтов в виде списка, тогда удобнее разбить его на группы по интересам:

Каталог сайтов:



Для этого нам и нужны теги *<optgroup> </optgroup>* с одним параметром *label*, который и задает название группе элементов.

Пример кода:

```
<form name="forma1">

 Каталог сайтов:<br>

 <select name="catalog" size="9">

  <optgroup label="Компьютеры">
   <option value="1">интернет</option>
   <option value="2">мобильники</option>
   <option value="3">hardware</option>
  </optgroup>

  <optgroup label="Работа">
```

```
 <option value="4">вакансии</option>
 <option value="5">трудоустройство</option>
 <option value="6">резюме</option>
</optgroup>

<optgroup label="Дом">
 <option value="7">здоровье</option>
 <option value="8">красота</option>
 <option value="9">дети</option>
</optgroup>

</select>

</form>
```

Обратите внимание, в данном случае необходимо указывать закрывающие теги *</option>*.

Надписи

Все элементы формы можно связать с их надписями при помощи элемента *<label>* и его параметра *for*, значением которого является имя элемента, с которым связываем надпись. Например:

```
<form name="forma1">

 <label for="load">Выбирайте файл: </label>
 <input type="file" name="load" size="30">

</form>
```

Результат:
Выбирайте файл:

Стоит ли его использовать решать вам. Мне кажется, что без него код короче, а результат тот же.

Обобщающий пример

```
<html>
```

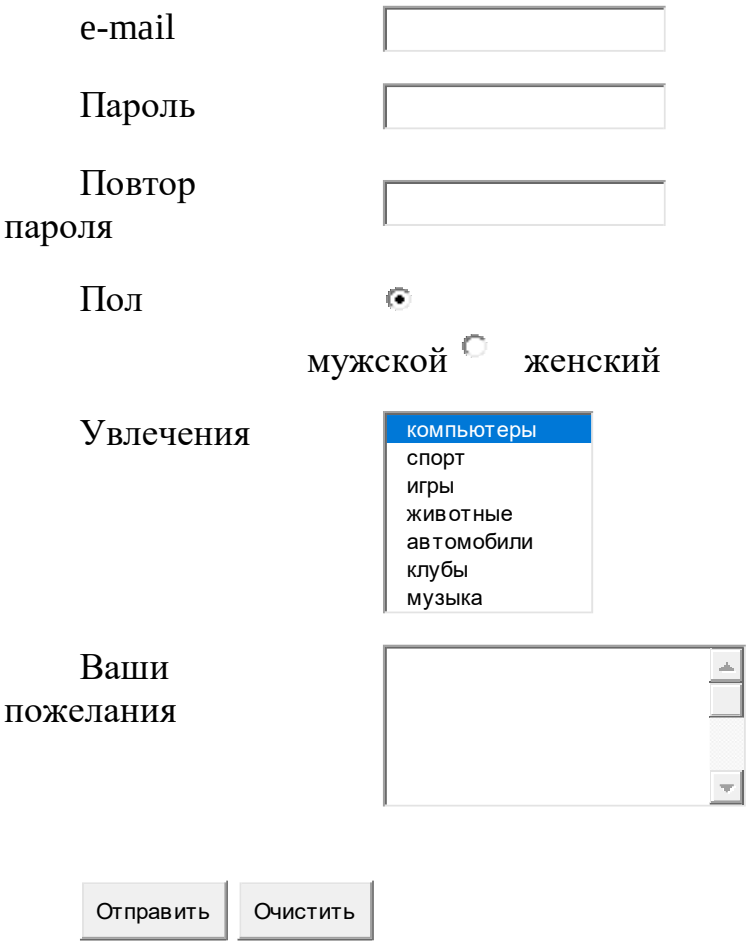
```
  <head>
      <title>Заголовок документа</title>
  </head>

  <body>

<form name="forma1">

 <table  border="0" cellspacing="5" cellpadding="5">

  <caption>Форма регистрации</caption>

  <tr>
   <td align="right" valign="top">Имя</td>
   <td><input type="text" name="name" size="25"></td>
  </tr>

  <tr>
   <td align="right" valign="top">e-mail</td>
   <td><input type="text" name="e-mail" size="25"></td>
  </tr>

  <tr>
   <td align="right" valign="top" >Пароль</td>
   <td>
    <input type="password" name="password" size="25">
   </td>
  </tr>

  <tr>
   <td align="right" valign="top" >Повтор пароля</td>
   <td>
    <input type="password" name="password2" size="25">
   </td>
  </tr>

  <tr>
   <td align="right" valign="top" >Пол</td>
   <td>
     <input type="radio" name="sex" value="man" checked>
     мужской
     <input type="radio" name="sex" value="woman">
     женский
   </td>
```

```
  </tr>

  <tr>
   <td align="right" valign="top">Увлечения</td>
   <td><select name="hobby" size="7" multiple>
            <option selected value="1">компьютеры
            <option value="2">спорт
            <option value="3">игры
            <option value="4">животные
            <option value="5">автомобили
            <option value="6">клубы
            <option value="7">музыка
      </select>
   </td>
  </tr>

  <tr>
   <td align="right" valign="top">Ваши пожелания</td>
   <td>
    <textarea cols="30" rows="3" wrap="physical">
    </textarea>
   </td>
  </tr>

  <tr>
   <td align="right" colspan="2">
    <input type="submit" name="submit" value="Отправить">
    <input type="reset" name="reset" value="Очистить">
   </td>
  </tr>

 </table>

</form>

  </body>

</html>
```

Результат:

Форма регистрации

Имя



На этом урок, посвященный формам, закончен. В следующем уроке мы узнаем, что такое фреймы.

Проверочная работа по теме язык гипертекстовой разметки HTML.

Практическая работа № 7 Работа с изображением

Инструкция для практического задания

На этом уроке разберёмся со вставкой изображения на практике. На страницу, созданную в предыдущем уроке, поместим любую фотографию.

Но для начала нужно конвертировать наше изображение, а точнее изменить под необходимый нам размер. Самым простым и удобным способом конвертации изображения является онлайн-конвертер, например,

http://www.softorbits.ru/resize- images-online/

1. Выбери на своём компьютере изображение, которое хочешь изменить (кнопка «**Обзор**»).

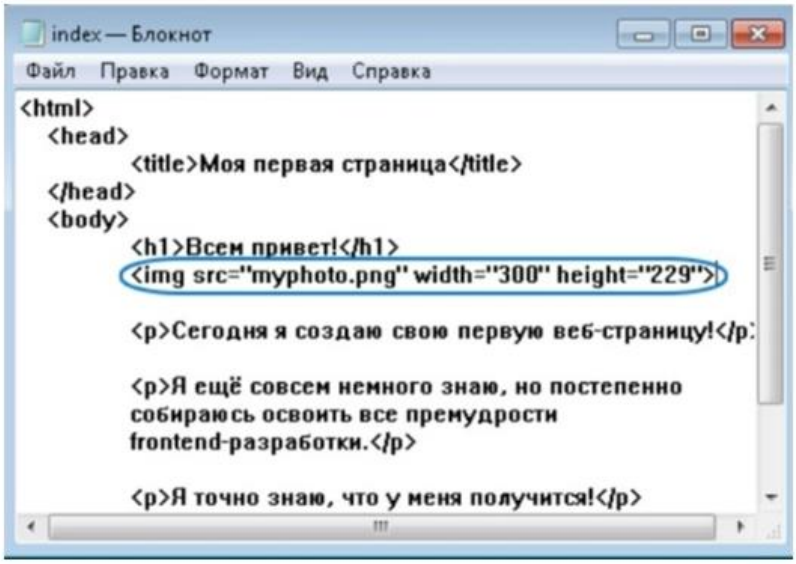
Укажем желаемый размер изображения (ты можешь указать ширину или высоту в пикселях, вторая величина будет изменена пропорционально).

2. Нажми кнопку **«Изменить размер (Resize)»** и немного подожди, через некоторое время браузер предложит тебе сохранить изменённый файл.

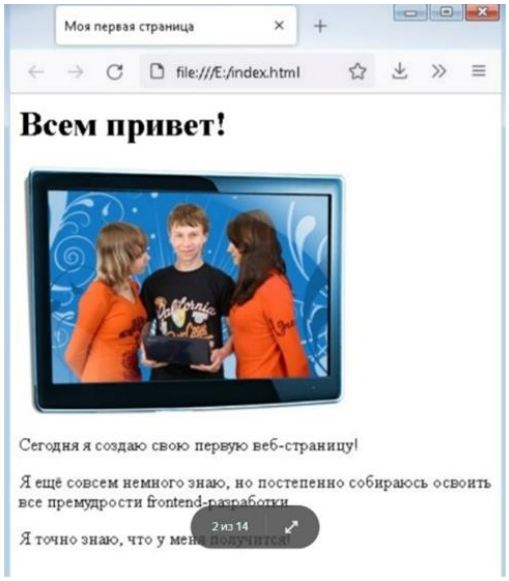
3. Сохрани или скопируй уменьшенную фотографию в то место, где размещена сейчас твоя страница.

4. Переименуй файл изображения, например, назови его «myphoto».

5. Запиши HTML-код вставки изображения на страницу сразу после заголовка, не забудь указать реальные размеры изображения (изображение может выглядеть искажённым, если ты не сохранишь правильное соотношение сторон фотографии, поэтому, если ты точно знаешь только высоту или только ширину, можешь указать всего один атрибут):



6. Сохрани внесённые в файл изменения и посмотри на страницу в браузере, должно получиться примерно так (только с твоей фотографией):



А ТЕПЕРЬ ПРОВЕДЁМ МАЛЕНЬКИЙ ЭКСПЕРИМЕНТ!

Допиши в теге <img> атрибут «alt», указав в нём альтернативный текст, например:
<img src="myphoto.png" width="300" height="229" alt="Это я">

Сохрани изменения и скопируй файл «index.html» в какую-нибудь другую папку, не копируя при этом файл самого изображения. Запусти файл в браузере из его нового места размещения:

Так как файл изображения теперь для страницы недоступен, вместо фотографии на странице будет присутствовать зарезервированное под её размеры место, а также будет отображён альтернативный текст, поясняющий, что изображено на отсутствующей картинке.

Эксперимент завершён, поэтому копию файла «index.html» теперь можно удалить.

Файл растрового изображения содержит информацию о расположении и цвете каждого пикселя, поэтому для качественного отображения такого файла на веб-странице необходимо, чтобы реальный размер изображения совпадал с указанными в атрибутах «width» и «height» размерами в пикселях.

Чтобы убедиться в этом, проведём ещё один эксперимент

Открой файл «index.html» и запиши в атрибутах «width» и «height» числа, в три раза превышающие исходные размеры твоего изображения.

Сохрани внесённые изменения и посмотри на обновлённую страницу в браузере.

Наверняка твоя фотография теперь выглядит зернистой или размытой, как будто она плохого качества.

Файл векторного изображения содержит алгоритм, по которому компьютер может вычислить, как должно выглядеть изображение, когда выводится на экран, а не информацию о каждом пикселе. Благодаря этому файлы векторных изображений намного меньше растровых по размеру и имеют высокую масштабируемость — при увеличении масштаба изображения пиксели не увеличиваются вместе с графикой, а потому векторное изображение при увеличении продолжает выглядеть ровным и красивым.

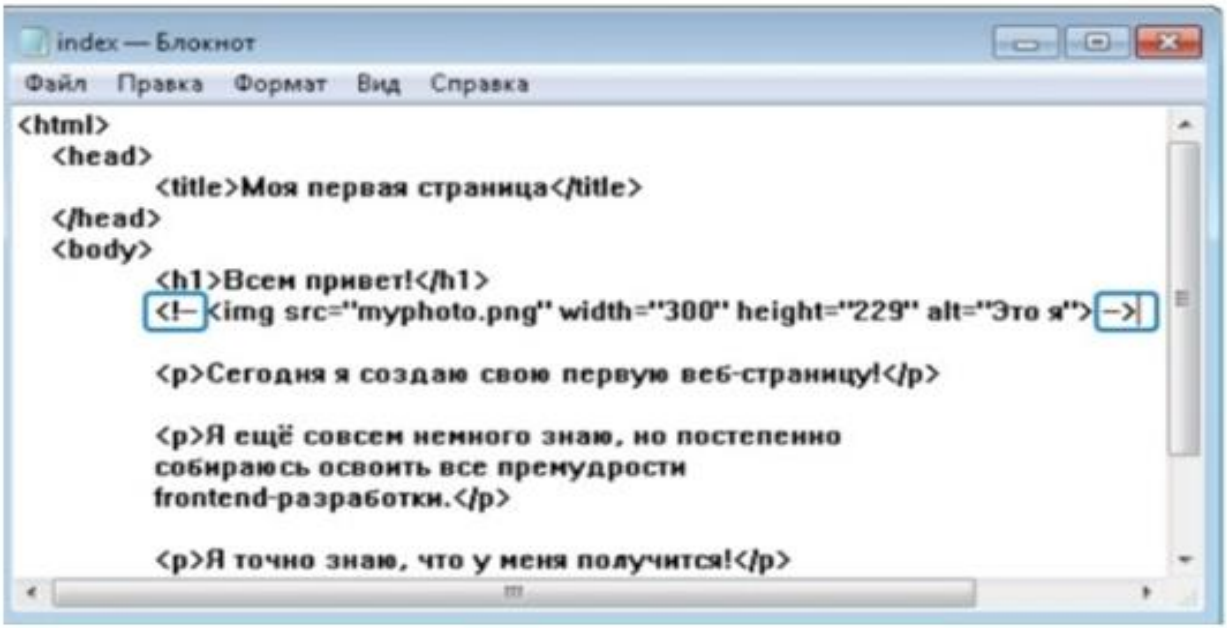
Чтобы убедиться в сказанном, заменим фотографию на нашей странице изображением формата SVG. Для этого тебе потребуется скачать файл «smile.svg» на свой компьютер и поместить его рядом с файлом «index.html».

Вместо удаления кода вставки фотографии на страницу, просто закомментируем его.

Комментарий в HTML-коде задаётся так:

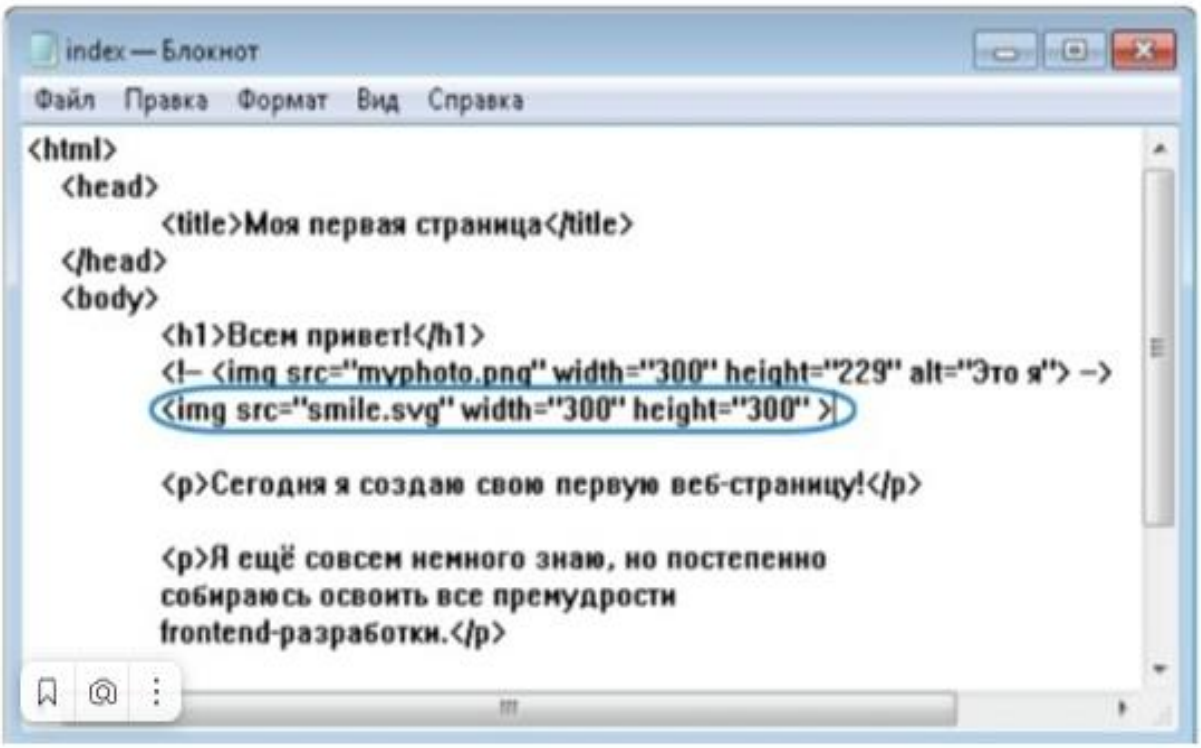
<!-- любой текст -->

Текст внутри комментария не отображается браузером на странице, поэтому для временного отключения кода достаточно просто поместить его в комментарий. Закомментируй код вставки изображения:

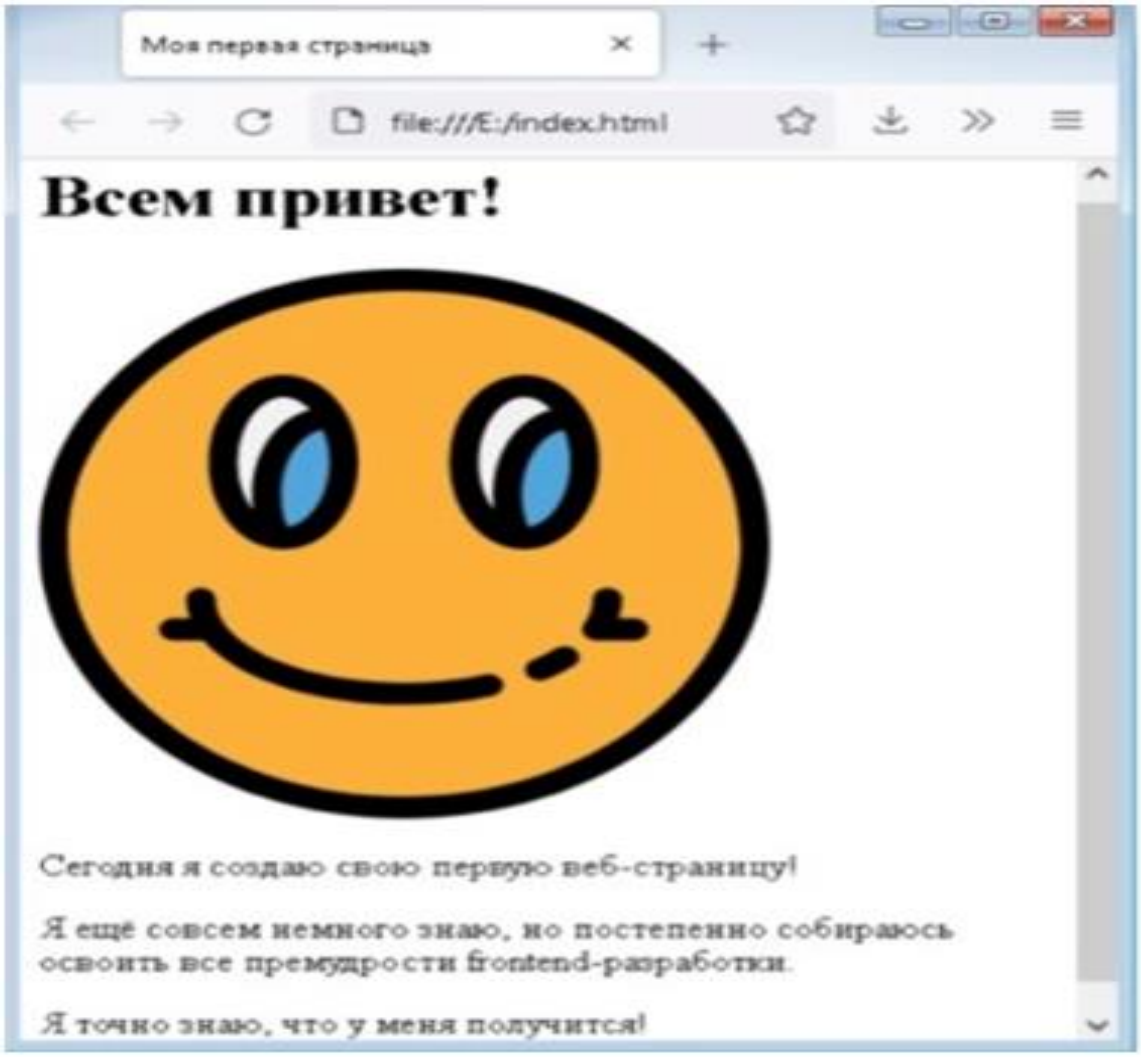


Сохрани изменения и обнови страницу в браузере, чтобы убедиться, что фотография больше не отображается на странице.

Допиши код вставки векторного изображения, указав в атрибутах «width» и «height» одинаковые размеры, так как данное изображение квадратное:



Сохрани изменения и обнови страницу в браузере. Теперь вместо фотографии ты увидишь на странице изображение смайла:



Поэкспериментируй с размером изображения, изменяя ширину и высоту на большие или меньшие значения, и убедись, что при любом размере качество изображения не ухудшается.

Инструкция для практического задания

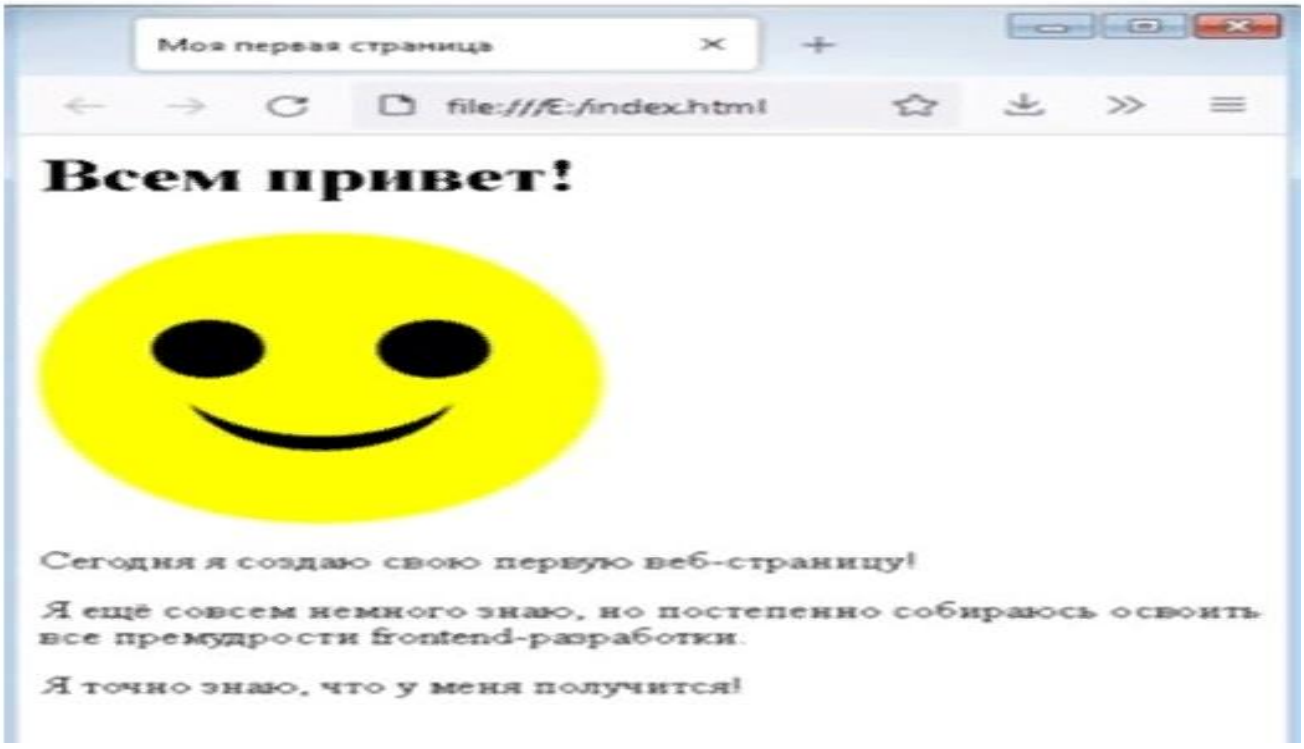
Простые SVG можно легко создавать вручную прямо в текстовом редакторе, не создавая при этом отдельного файла изображения.

Давай попробуем!

В файле «index.html» закомментируй ещё и код вставки векторного изображения, а затем помести под ним следующий код:

```
<svg width="200" height="200" viewBox="0 0 200 200">
<circle cx="100" cy="100" r="100" fill="yellow" />
<circle cx="100" cy="100" r="50" fill="black" />
<circle cx="100" cy="80" r="60" fill="yellow" />
<circle cx="60" cy="80" r="20" fill="black" />
<circle cx="140" cy="80" r="20" fill="black" />
</svg>
```

Сохрани изменения и обнови страницу в браузере. Теперь ты увидишь на странице изображение смайла, которое вычислено и отрисовано на основе этого кода:



В данном коде задана последовательная отрисовка пяти окружностей с соответствующими координатами центров этих окружностей, радиусами и цветами заливки.

Координаты фигур заданы относительно координат, указанных в атрибуте **«viewBox»**, это как бы исходный размер изображения.

А атрибуты «width» и «height» определяют, сколько места изображение занимает в браузере.

Поэтому если в теге <svg> просто поменять размеры «width» и «height», то вся картинка перерисуется, новые координаты будут вычислены автоматически.

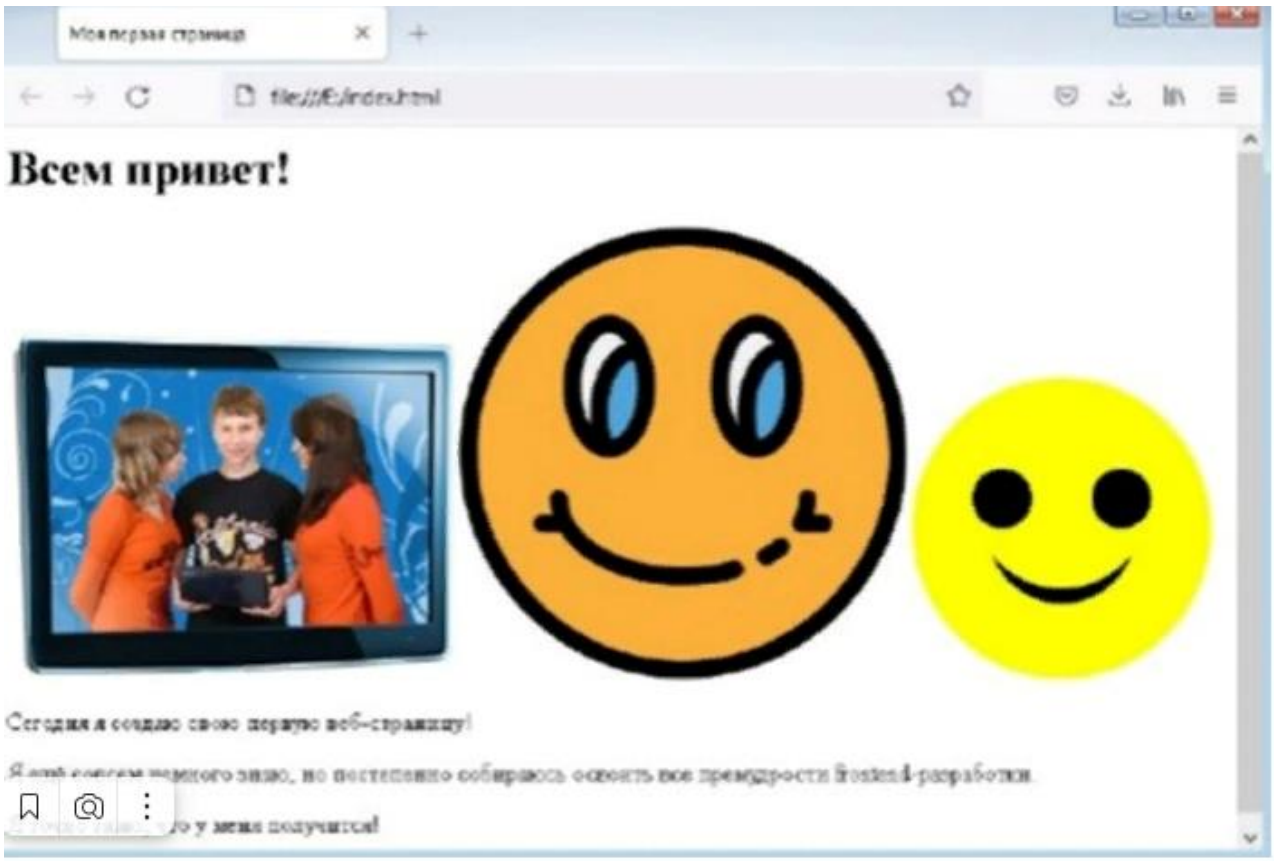
Обрати внимание на символ «/» перед закрывающей тег <circle> угловой скобкой, он обязателен для всех непарных тегов в SVG!

Что ещё следует знать об изображениях на веб-странице?

Изображение — это строчный элемент!

В языке HTML существуют блочные и строчные элементы. Браузеры отображают блочные элементы с переводом строки до и после элемента, то есть блочные элементы всегда начинаются с новой строки, а строчные элементы размещаются непосредственно в строке.

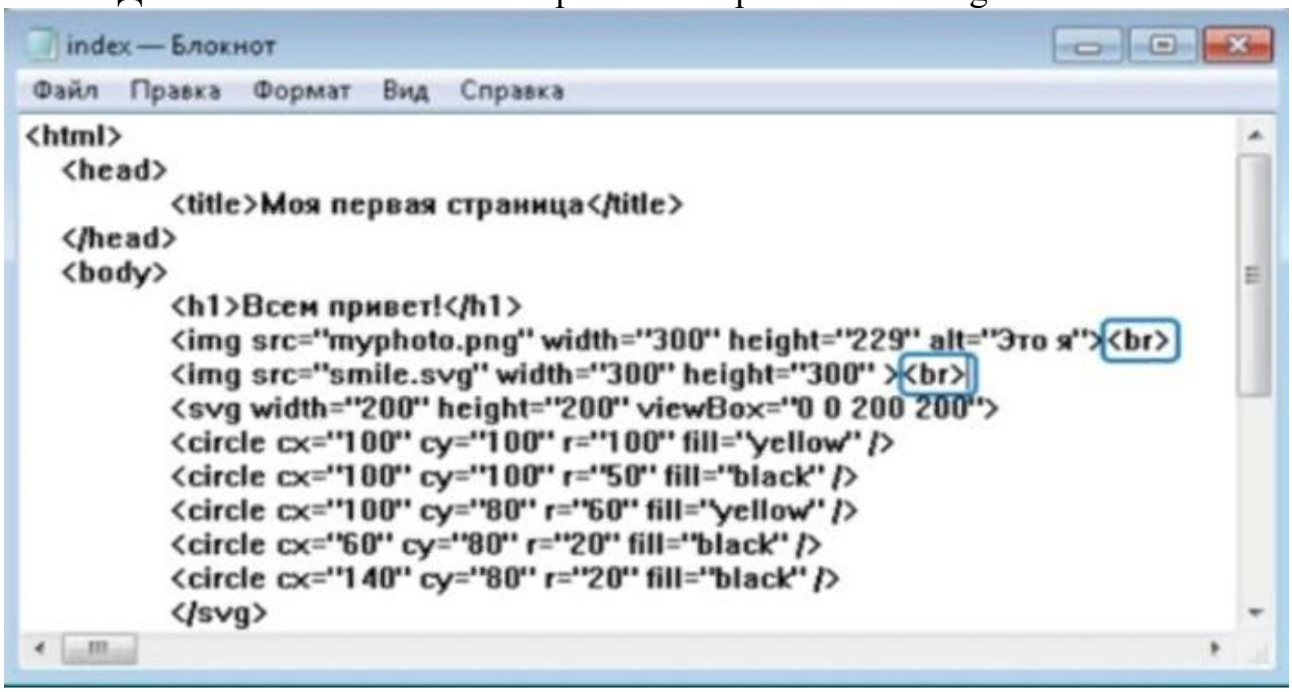
Чтобы понять, о чём речь, вернёмся к нашей первой веб-странице. Открой в текстовом редакторе файл «index.html» и убери комментарии, в которые были помещены первые два изображения. Открой страницу в браузере и убедись, что все три изображения отображаются в одной строке (они могут переместиться на новую строку, только если ширина окна браузера не позволит им поместиться в одной строке):



Для того чтобы разместить каждое изображение на отдельной строке, придётся воспользоваться тегом принудительного переноса строки
 или поместить изображения внутрь блочных элементов. К блочным элементам относятся, например, абзац ( ) или заголовок (<h1></h1>).

Проверим оба способа, чтобы понять, как это работает.

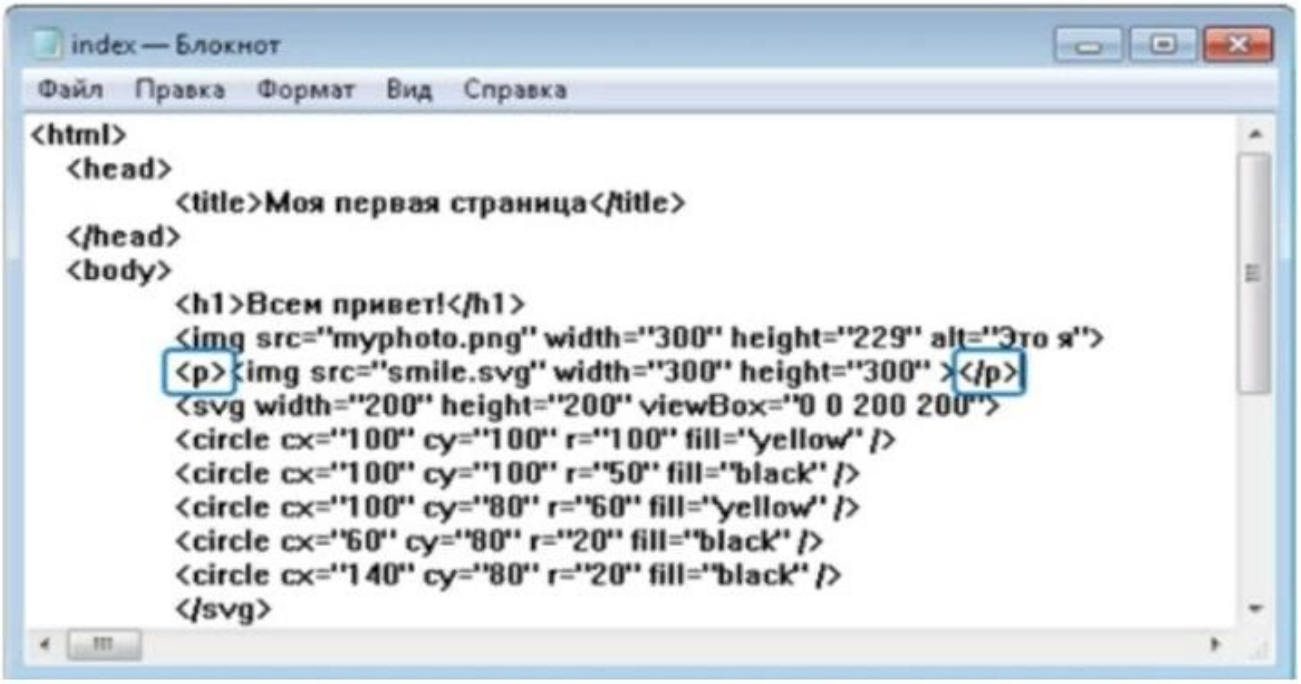
1. Допиши теги
 после первого и второго тегов <img>:



Обрати внимание, что вставлять тег
 после третьего изображения, которое внедрено на страницу с помощью тегов <svg></svg>, мы не стали. Но это не потому, что элемент <svg></svg> блочный (он так же, как и элемент <img>, является строчным), а потому, что далее в коде идёт блочный элемент , его содержимое браузер автоматически отобразит с новой строки.

2. Сохрани изменения и обнови страницу в браузере, чтобы убедиться, что каждое изображение теперь размещается на новой строке.

3. Теперь удали теги
 и помести второе изображение в теги :



Сохрани изменения и обнови страницу в браузере, чтобы убедиться, что каждое изображение по-прежнему размещается на новой строке.

Инструкция для практического задания

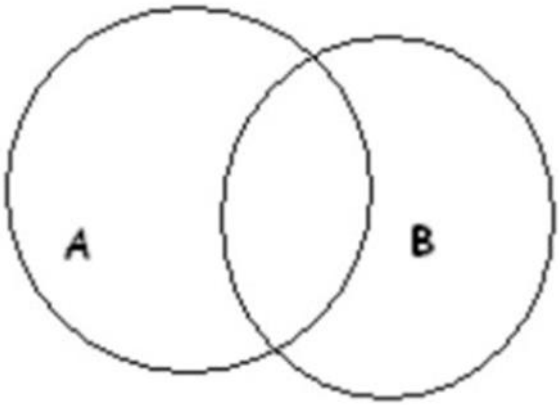
Далее мы рассмотрим очень интересную возможность HTML: научимся накладывать на изображения карты с активными областями!

Для изучения этой возможности на реальном примере создадим новый проект.

1. Создай новую папку (каждый проект лучше хранить в отдельной директории), а в ней создай новый HTML-документ, назови его «test.html».
2. Создай в директории проекта папку «img», скачай и сохрани в ней файл изображения «test.png».

Теперь поясним, что именно мы будем делать. Мы создадим простой тест для пользователя: поместим на веб-страницу изображение двух пересекающихся кругов и попросим щёлкнуть в области их пересечения. При

этом, наложив на изображение карту активных областей, мы сможем определить, по какой именно области пользователь щёлкнул!

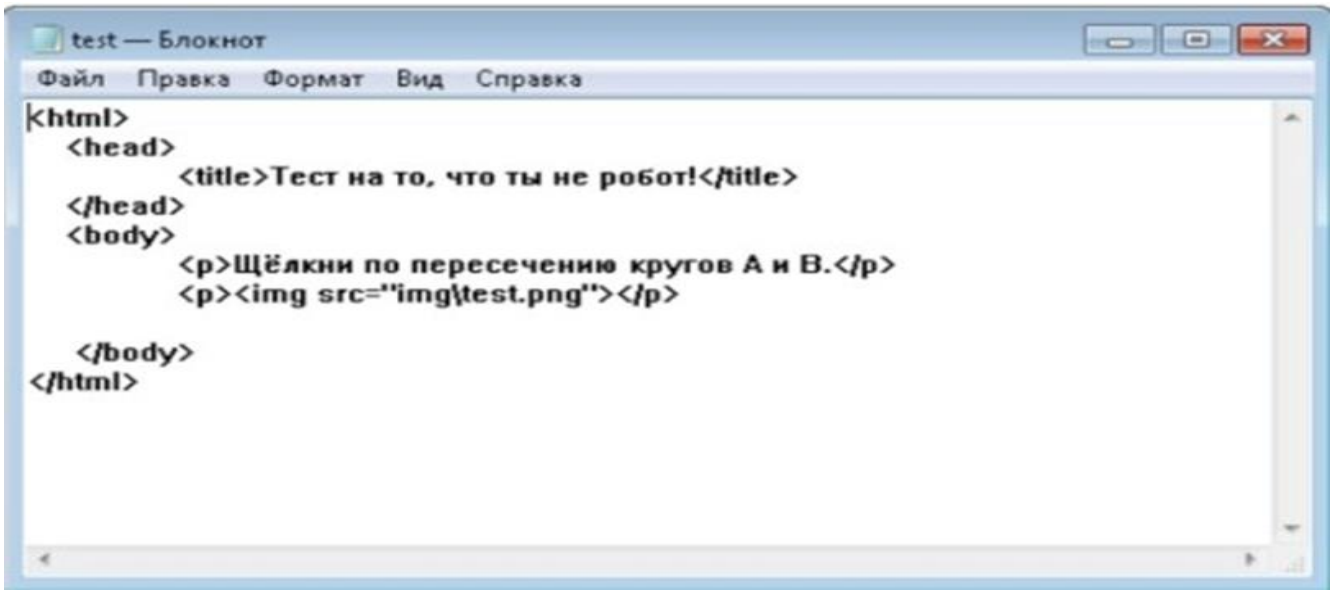


Приступим!

1. Открой файл «test.html» в текстовом редакторе и запиши теги структуры HTML-документа.

2. Дай документу оригинальный заголовок, например: «Тест на то, что ты не робот!».

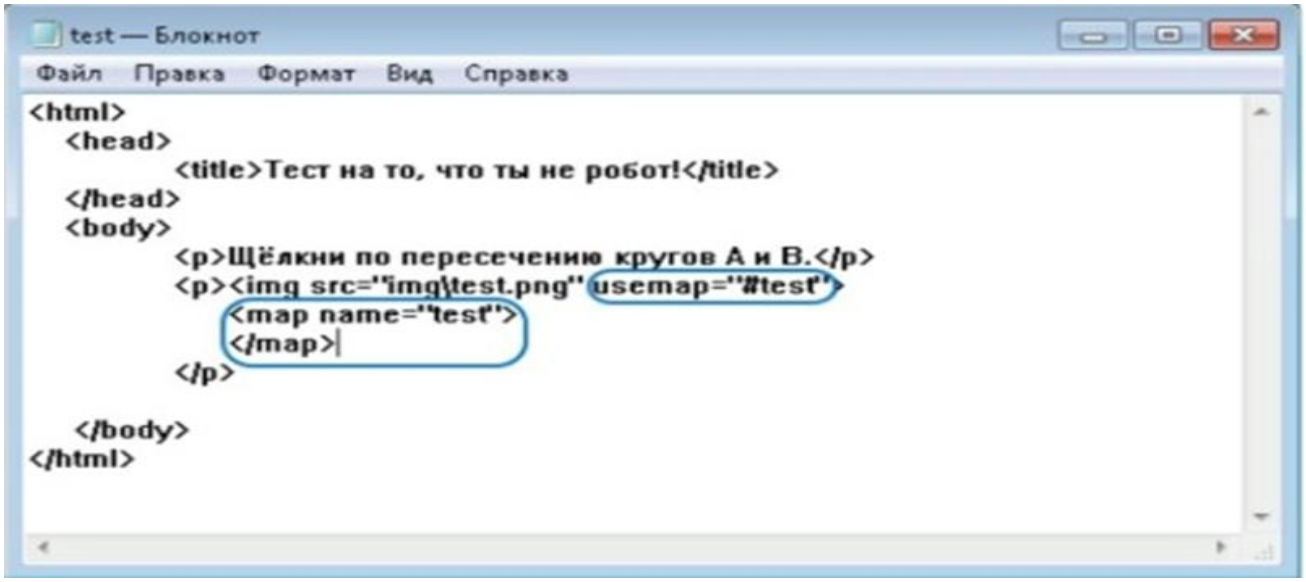
3. Создай на странице два абзаца, в первый помести текст задания, а во второй — изображение:



Для представления графического изображения в виде карты с активными областями служит элемент <map>.

Для этого элемента доступен атрибут «name», который задаёт имя карты. Значение атрибута «name» для элемента <map> должно соответствовать имени в атрибуте «usemap» элемента <img>.

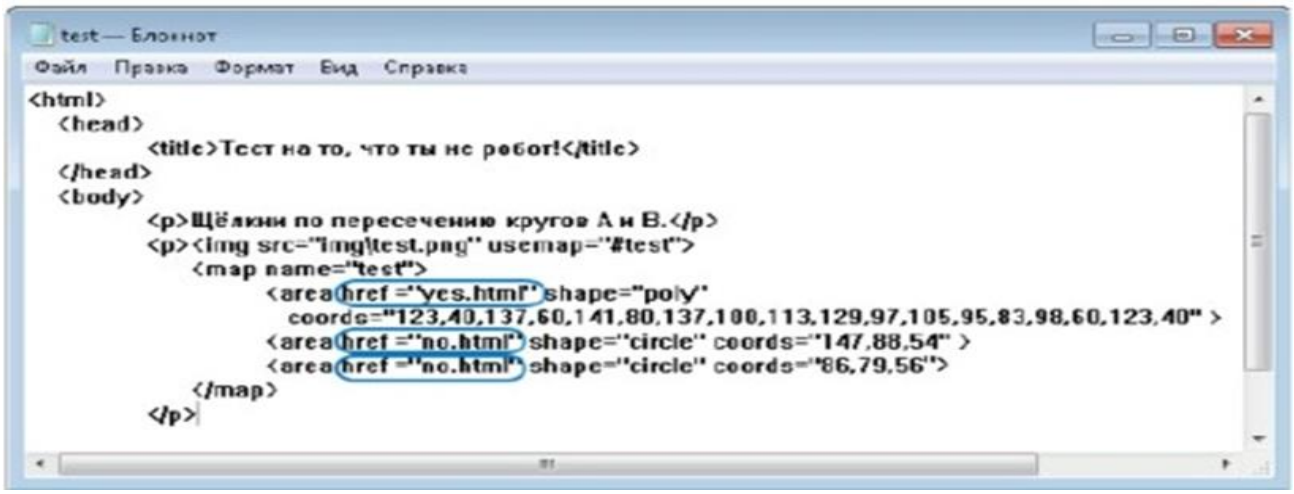
Запиши соответствующие теги и атрибуты в своём коде:



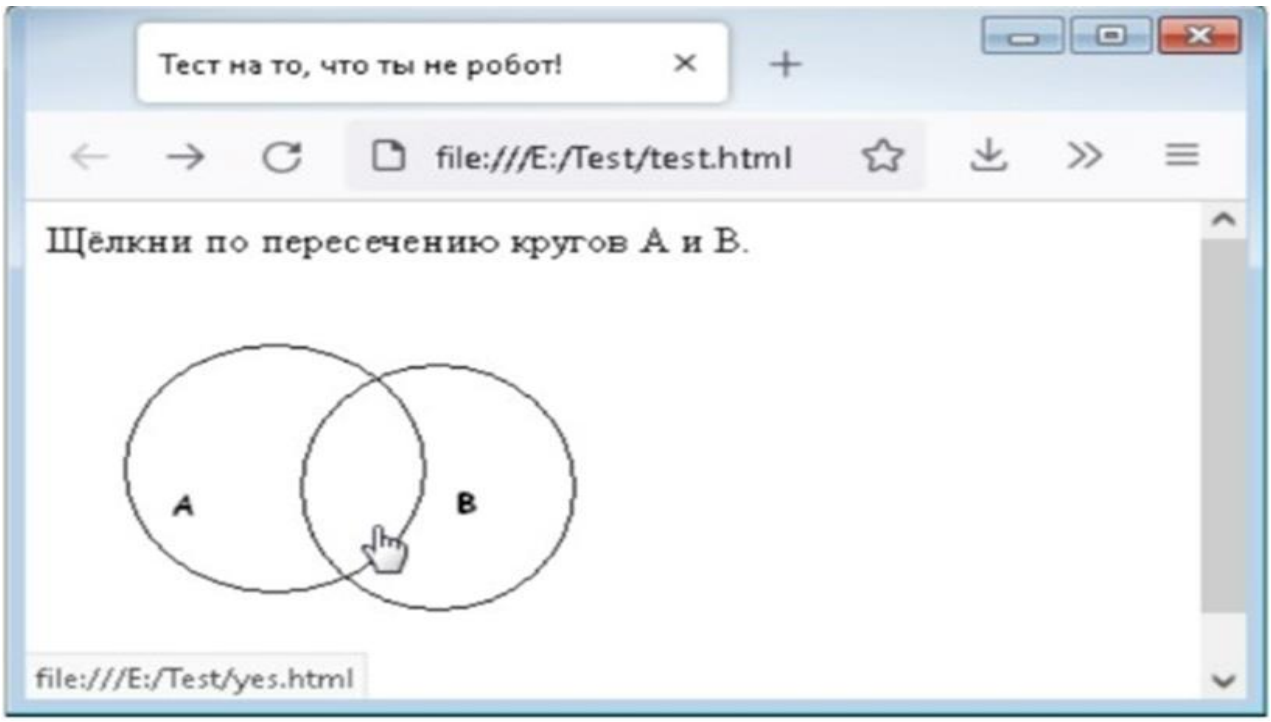
Итак, чтобы создать активную область на карте, нужно поместить в теги <map></map> новый тег <area> и указать в атрибутах форму и координаты активной области, а также ссылку, по которой будет происходить переход при щелчке по данной активной области.

4. Запиши в тегах <map></map> соответствующие области <area> и допиши в них атрибуты «href» со ссылками на страницы «yes.html» и «no.html».

Обязательным также является атрибут, задающий альтернативный текст, но так как текст в атрибуте «alt» отображается только тогда, когда изображение недоступно, а в нашем примере смысл задания при этом совершенно теряется, то мы можем опустить этот атрибут или записать «alt="»».



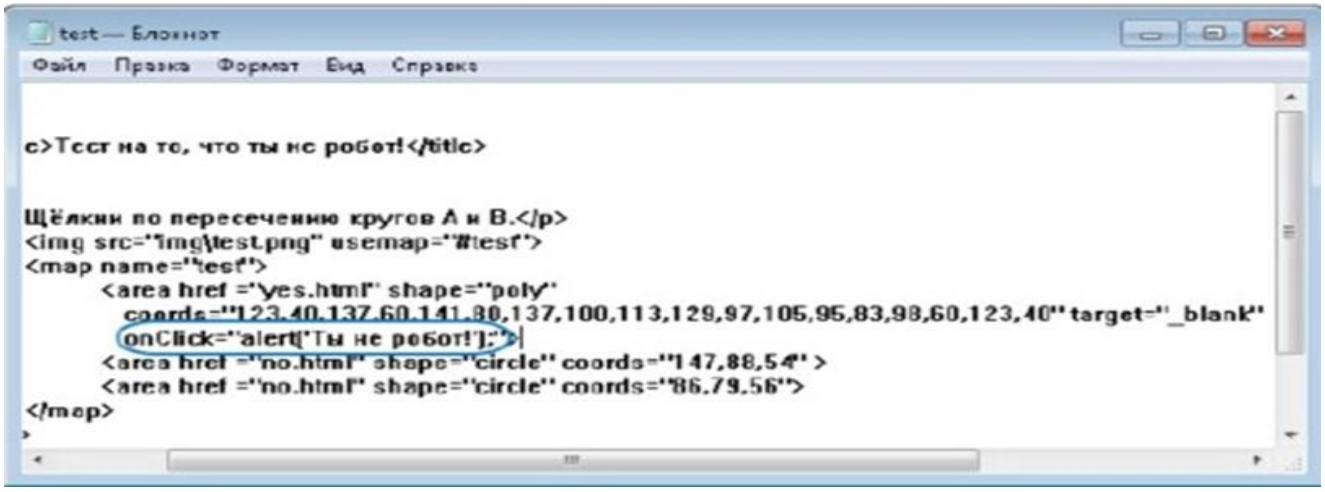
Сохрани изменения в документе и обнови страницу в браузере. Теперь при наведении мыши на изображение ты увидишь, что курсор изменяет свой внешний вид, как бы приглашая кликнуть:



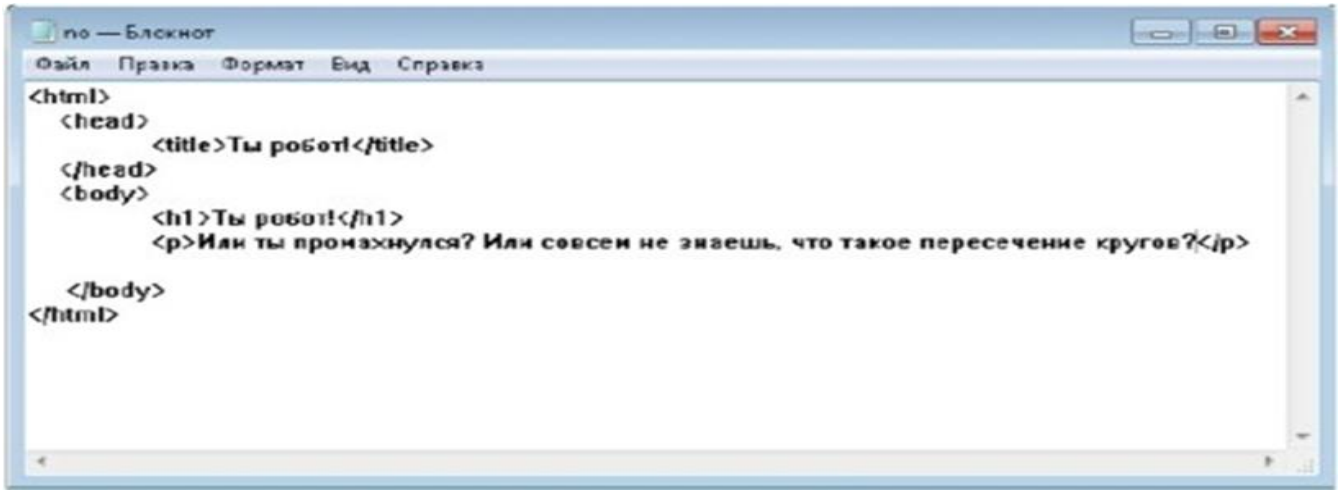
В атрибутах «href» тегов <area> мы указали ссылки на так называемые связанные документы. Это веб-страницы yes.html и no.html, которые теперь связаны с нашей первой страницей. Вот только их ещё не существует, поэтому щелчок по любой из активных областей изображения приведёт к тому, что браузер сообщит нам о том, что страница не найдена. Создадим эти страницы, чтобы система связанных документов заработала!

1. В папке со страницей test.html создай новые файлы yes.html и no.html.

2. В коде страницы yes.html запиши реакцию на правильное нажатие:



В коде страницы «no.html» запиши реакцию на промах:

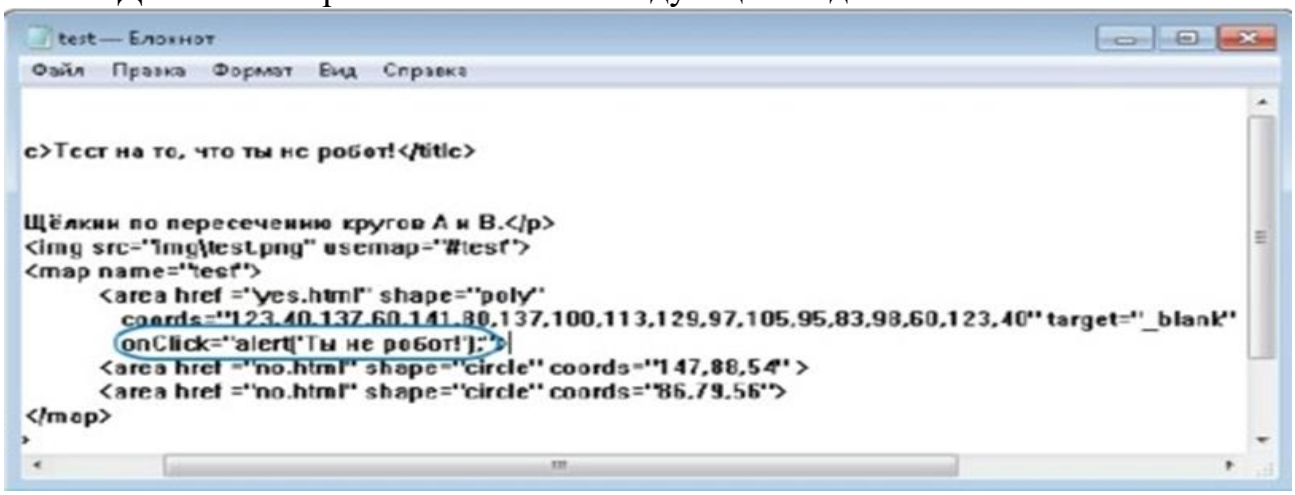


3. Сохрани все изменения и протестируй работу теста в браузере.

4. Добавь атрибут «target» в тегах <area>, чтобы настроить открытие связанных с первой страницей документов в новых вкладках браузера. Использовать карты изображений с активными областями можно не только для перехода по ссылкам к связанным документам. Событие щелчка по активной области изображения можно обработать при помощи языка программирования JavaScript.

Данный язык программирования будет изучаться в отдельном модуле, поэтому сейчас мы не будем реализовывать сложную обработку, а просто поэкспериментируем, чтобы понять, как это работает.

5. Допиши в первом теге <area> следующий код:



6. Сохрани изменения и протестируй страницу в браузере.

Теперь перед переходом к связанному документу «yes.html» появится окно с сообщением «Ты не робот!». Именно так сработает функция «alert» языка JavaScript, записанная в событии «onClick» тега <area>.

Проверочная работа по теме язык гипертекстовой разметки HTML

1 вариант

Каждый правильный ответ оценивается в 0,5 балла. Оценка выставляется по совокупности набранных баллов.

_____ (ФИ)

№ п.п.	Вопрос	Ответ	Правильный ответ -0,5 Неправильный ответ -0
1.	Что такое теги?		
2.	Гипертекстовый документ –это…..		
3.	Перечислите основные части HTML –документа		
4.	Определите что это за тег <head>, является ли этот тег двойным?		
5.	Что такое тело документа <body> </ body >		
6.	Определите, какой тег является тегом создания размера заголовков		
7.	Как изменить гарнитуру текста, а документе HTML?		
8.	Опишите файл. Определите его название, и в каком приложении он был создан	a) b)	

	a)Klop. mov b)портрет.png c) koleso.pptx	c)	
	Что произойдет, если разместить данный тег в документе <A HREF=”#ПН”> понедельник</A>		
	Дайте описание тегу <TD> </TD>		

Проверочная работа по теме язык гипертекстовой разметки HTML

2 вариант

Проверочная работа по теме язык гипертекстовой разметки HTML.

Каждый правильный ответ оценивается в 0,5 балла. Оценка выставляется по совокупности набранных баллов.

_____ (ФИ)

№ п.п.	Вопрос	Ответ	Правильный ответ -0,5 Неправильный ответ -0
1.	Что такое теги?		
2.	Гипертекстовый документ –это…..		
3.	Перечислите основные части HTML –документа		
4.	Определите что это за тег <head>, является ли этот тег двойным?		

5.	Что такое тело документа <body> </ body >		
6.	Определите, какой тег является тегом создания размера заголовков		
7.	Как изменить гарнитуру текста, а документе HTML?		
8.	Опишите файл. Определите его название, и в каком приложении он был создан a)Klop. mov b)портрет.png c) koleso.pptx	a) b) c)	
9.	Что произойдет, если разместить данный тег в документе <A HREF=”#ПН”> понедельник</A>		
10.	Дайте описание тегу <TD> </TD>		

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

1. Создать web-страницу с заголовком **Текст_Фамилия**.
2. Ввести текст «**Самостоятельная работа №1. Размещение и форматирование текста**», используя заголовки любого уровня.
3. Создать web-страницу (содержание произвольное, не менее 3 абзацев), для оформления применить теги управления внешним видом web-страницы, форматирования символов. В коде документа изменить **цвет фона** и **текста** (на усмотрение студента), **размеры полей** (произвольно). В конце страницы вставить символ авторского права и указать Фамилию Имя Отчество, институт, группу.
4. Разделить абзацы горизонтальными линиями с различными значениями атрибутов.

Задание 2.

1. Создать web-страницу с заголовком **Списки_Фамилия**.
2. Ввести текст «**Самостоятельная работа №2. Организация списков**», используя заголовки любого уровня.
3. Использовать элементы оформления внешнего вида HTML-документа (начертание, цвет шрифта, фон страницы и т.д).
4. Создать HTML-списки:
   - *нумерованный:*

Интернет и World Wide Web

Здесь вы можете узнать:

A. Что такое Интернет
B. Историю создания Интернета
C. Основные понятия среды Интернета

Интернет и World Wide Web

Здесь вы можете узнать:

5. Что такое Интернет
6. Историю создания Интернета
7. Основные понятия среды Интернета

   - *список определений:*

Интернет и World Wide Web

Здесь вы можете узнать:

Что такое Интернет
: Определение понятия
Историю создания Интернета
: Кто придумал Интернет
Основные понятия среды Интернета
: Составляющие Интернета

– ***многоуровневый:***

Интернет и World Wide Web

Здесь вы можете узнать:

1. Что такое Интернет
   - Определение понятия
2. Историю создания Интернета
   - Кто придумал Интернет
3. Основные понятия среды Интернета

Составляющие Интернета:

Электронная почта, телеконференции, поисковые системы ...

– ***вложенный список:***

ОГЛАВЛЕНИЕ

I. Общие сведения о программировании.

1. Введение.
2. Что такое информатика?
3. Что такое информация?
4. Что такое вычислительная машина?
- *Общие положения.*
- *Подробнее о памяти.*
- *Ввод-вывод.*
- *Оперативная память, внешняя память.*
- *Порядок некоторых величин.*

5. Что может делать вычислительная машина?
6. Что такое программирование?
7. Несколько ключевых слов.
8. Краткая история информатики.
- *a. Прединформатика.*
- *b. Протоинформатика.*
- *c. Информатика.*
- *d. Библиография.*
- *e. Задача.*

II. Введение в языки программирования.

5. Создать бегущую текстовую строку (направление указать произвольно): Фамилия Имя Отчество, Институт, Группа, Дата.

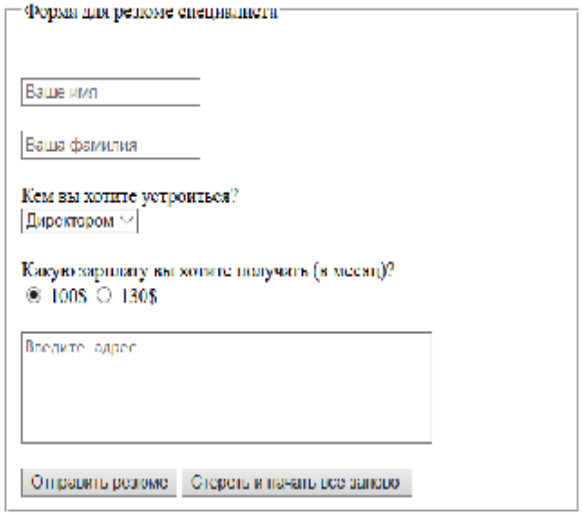
Задание 3.

1. Создать web-страницу с заголовком **Таблицы_Фамилия**.
2. Ввести текст «**Самостоятельная работа №3. HTML-таблицы**», используя заголовок любого уровня.
3. Создать таблицу с заголовком, включающую не менее 4 столбцов и 5 строк с текстовой, числовой информацией и специальными символами. Применить к таблице объединение ячеек, размер и выравнивание таблицы, использовать цвет и выравнивание текста в таблице.
4. Создать web-страницу с заголовком **Изображения_Фамилия**.
5. Ввести текст «**Самостоятельная работа №3. Ссылки. Изображения**», используя заголовок любого уровня.

6. Вставить изображение на web-страницу, сделать изображение-ссылку на страницу с таблицей.
7. Выполнить внутреннюю гиперссылку на странице с таблицей.

Задание 4.

1. Создать web-страницу с заголовком **Формы_Фамилия**.
2. Ввести текст «**Самостоятельная работа №3. HTML-формы**», используя заголовок любого уровня.
3. Создать форму по образцу, содержащую поля для ввода текстовой информации, список с возможностью выбора, радиопереключатель, 2 кнопки, рамку оформления:



Задание 5.

Создать личный сайт на любую тему, отвечающий требованиям:

1. Продумать структуру сайта: число HTML-страниц, их содержимое;
2. Создать главную страницу сайта с помощью фреймов (не менее трех);
3. На главной странице должно быть навигационное меню для возможности перехода на любую другую страницу сайта;
4. Сайт должен иметь оригинальное оформление;
5. Стилистика оформления должна быть единой для всех страниц сайта;
6. У сайта должен быть логотип (фото), соответствующий тематике сайта и расположенный в шапке сайта, желательно использовать бегущую строку;
7. Добавить на сайт страницу с обратной связью, выполненной в виде HTML-формы.

ПРИЛОЖЕНИЕ

Приложение 1

Таблица 1 - Полезные знаки и символы

Вид	HTML-код	Описание
	&#128386;	Обратная сторона конверта
	&#9731;	Снеговик
□	&#9875;	Якорь
	&#9990;	Знак телефона
	&#9742;	Телефон
□	&#9749;	Горячие напитки
	&#9998;	Карандаш, направленный вправо-вниз
	&#9999;	Карандаш
	&#10000;	Карандаш, направленный вправо-вверх
	&#10001;	Незакрашенное острие пера
	&#10002;	Закрашенное острие пера
□	&#9884;	Геральдическая лилия

❄	&#10052;	Снежинка
❤	&#10084;	Закрашенное жирное сердце
❅	&#10053;	Зажатая трилистниками снежинка
❆	&#10054;	Жирная остроугольная снежинка
★	&#9733;	Закрашенная звезда
☆	&#9734;	Незакрашенная звезда
✪	&#10026;	Незакрашенная звезда в закрашенном круге
✯	&#10027;	Закрашенная звезда с незакрашенным кругом внутри
✰	&#10031;	Вращающаяся звезда
❇	&#10057;	Звёздочка с шарообразными окончаниями
✳	&#10059;	Жирная восьмиконечная каплеобразная звездочка-пропеллер
✴	&#10034;	Звёздочка с незакрашенным центром
☀	&#9728;	Закрашенное солнце с лучами
☁	&#9729;	Облака

Список использованной литературы

1. https://reader.lanbook.com/book/304958?lms=b509701765d0404a973f107ea8b24c31#30
2. Крис Минник, Эд Титтел - "HTML5 и CSS3 для чайников ", Диалектика, 2016, 400 стр. (ориг. название: "Beginning HTML5 and CSS3 For Dummies", John Wiley & Sons)
3. А. Хрусталев, А. Кириченко "HTMLS + CSS3. Основы современного WEB-дизайна", Наука и Техника, 2018, 352 стр.
4. Бен Фрейн (Ben Frain) - "HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств", Питер, 2017, 272 стр. (ориг. название: "Responsive Web Design with HTML5 and CSS3", Packt Publishing)
5. Дженнифер Нидерст Роббинс "HTML5, CSS3 и JavaScript. Исчерпывающее руководство". 4-ое издание (2014)
6.

Спецсимволы

№		&#8470;	Знак номера
		&#128269;	Лупа (наклонённая влево)
		&#128270;	Лупа (наклонённая вправо)
		&#9742;	Телефон
		&#9993;	Конверт, email, почта
		&#128190;	Дискета
		&#128736;	Молоток и гаечный ключ, настройка
		&#128274;	Замок закрыт
		&#128275;	Замок открыт
		&#128276;	Колокольчик
		&#128277;	Колокольчик перечеркнутый
		&#128465;	Урна
		&#128293;	Огонь
		&#128711;	Запрещено
		&#9940;	Вход запрещен (кирпич)
		&#9971;	Флаг в воронке, местоположение, место встречи, гольф

Дроби

Символ	HTML-код	CSS-код	Юникод®	Мнемоника	Название
¼	&#188;	\00BC	U+00BC	&frac14;	Дробь – одна четверть
½	&#189;	\00BD	U+00BD	&frac12;	Дробь – одна вторая
¾	&#190;	\00BE	U+00BE	&frac34;	Дробь – три четверти
⅓	&#8531;	\2153	U+2153	&frac13;	Простая дробь одна треть
⅔	&#8532;	\2154	U+2154	&frac23;	Простая дробь две трети
⅕	&#8533;	\2155	U+2155	&frac15;	Простая дробь одна пятая
⅖	&#8534;	\2156	U+2156	&frac25;	Простая дробь две пятые
⅗	&#8535;	\2157	U+2157	&frac35;	Простая дробь три пятые
⅘	&#8536;	\2158	U+2158	&frac45;	Простая дробь четыре пятые
⅙	&#8537;	\2159	U+2159	&frac16;	Простая дробь одна шестая
⅚	&#8538;	\215A	U+215A	&frac56;	Простая дробь пять шестых
⅛	&#8539;	\215B	U+215B	&frac18;	Простая дробь одна восьмая
⅜	&#8540;	\215C	U+215C	&frac38;	Простая дробь три восьмые
⅝	&#8541;	\215D	U+215D	&frac58;	Простая дробь пять восьмых
⅞	&#8542;	\215E	U+215E	&frac78;	Простая дробь семь восьмых

Символ	HTML-код	CSS-код	Юникод®	Мнемоника	Название
©	&#169;	\00A9	U+00A9	&copy;	Знак авторского права
®	&#174;	\00AE	U+00AE	&reg;	Зарегистрированный товарный знак
™	&#8482;	\2122	U+2122	&trade;	Знак торговой марки
@	&#64;	\0040	U+0040	&commat;	Символ собака
ℂ	&#8450;	\2102	U+2102	&Copf;	Дважды начерченная заглавная C
℅	&#8453;	\2105	U+2105	&incare;	Забота о
ℊ	&#8458;	\210A	U+210A	&gscr;	Каллиграфическая строчная буква g
ℋ	&#8459;	\210B	U+210B	&hamilt;	Каллиграфическая заглавная буква H
ℌ	&#8460;	\210C	U+210C	&Hfr;	Готическая заглавная буква H
ℍ	&#8461;	\210D	U+210D	&Hopf;	Дважды начерченная заглавная буква H
ℎ	&#8462;	\210E	U+210E	&planckh;	Постоянная Планка
ℏ	&#8463;	\210F	U+210F	&planck;	Постоянная Планка делённая на два пи (константа Дирака)
ℐ	&#8464;	\2110	U+2110	&Iscr;	Каллиграфическая заглавная буква I
ℑ	&#8465;	\2111	U+2111	&image;	Готическая заглавная буква I
ℒ	&#8466;	\2112	U+2112	&Lscr;	Каллиграфическая заглавная буква L
ℓ	&#8467;	\2113	U+2113	&ell;	Каллиграфическая строчная l
ℕ	&#8469;	\2115	U+2115	&Nopf;	Дважды начерченная заглавная буква N
№	&#8470;	\2116	U+2116	&numero;	Знак номера
℗	&#8471;	\2117	U+2117	&copysr;	Авторское право звукозаписи
℘	&#8472;	\2118	U+2118	&weierp;	Каллиграфическая заглавная буква P
ℙ	&#8473;	\2119	U+2119	&Popf;	Дважды начерченная заглавная буква P
ℚ	&#8474;	\211A	U+211A	&Qopf;	Дважды начерченная заглавная буква Q
ℛ	&#8475;	\211B	U+211B	&Rscr;	Каллиграфическая заглавная буква R
ℜ	&#8476;	\211C	U+211C	&real;	Готическая заглавная буква R
ℝ	&#8477;	\211D	U+211D	&Ropf;	Дважды начерченная заглавная буква R
℞	&#8478;	\211E	U+211E	&rx;	Взять рецепт
ℤ	&#8484;	\2124	U+2124	&Zopf;	Дважды начерченная заглавная буква Z

(◕‿◕) SYMBL

ℸ	ℸ	\2138	U+2138	ℸ	Символ далет
𝔻	ⅅ	\2145	U+2145	ⅅ	Дважды начерченная курсивная заглавная буква D
𝕕	ⅆ	\2146	U+2146	ⅆ	Дважды начерченная курсивная строчная буква d
𝕖	ⅇ	\2147	U+2147	ⅇ	Дважды начерченная курсивная строчная буква e
𝕚	ⅈ	\2148	U+2148	ⅈ	Дважды начерченная курсивная строчная буква i
★	★	\2605	U+2605	★	Закрашенная звезда
☆	☆	\2606	U+2606	☆	Незакрашенная звезда
☎	☎	\260E	U+260E	☎	Значок телефона
♀	♀	\2640	U+2640	♀	Венера (Женский знак)
♂	♂	\2642	U+2642	♂	Марс (Мужской знак)
♠	♠	\2660	U+2660	♠	Пики закрашенные
♣	♣	\2663	U+2663	♣	Трефы закрашенные
♥	♥	\2665	U+2665	♥	Черви закрашенные
♦	♦	\2666	U+2666	♦	Бубны закрашенные
◊	◊	\25CA	U+25CA	◊	Ромб
♪	♪	\266A	U+266A	♪	Восьмая нота
♭	♭	\266D	U+266D	♭	Музыкальный знак бемоль
♮	♮	\266E	U+266E	♮	Музыкаль Музыкальный знак бемоль
♯	♯	\266F	U+266F	♯	Музыкальный знак диез
✓	✓	\2713	U+2713	✓	Символ галочка
✗	✗	\2717	U+2717	✗	Крестик в анкете голосования
✠	✠	\2720	U+2720	✠	Мальтийский крест
✶	✶	\2736	U+2736	✶	Шестиконечная закрашенная звезда
❘	❘	\2758	U+2758	❘	Тонкая вертикальная черта
❨	❲	\2772	U+2772	❲	Тонкая левая скобка панцерообразной формы
❩	❳	\2773	U+2773	❳	Тонка правая скобка панцерообразной формы