**Sololearn HTML**

* 1. **УРОК (26.09.2021)**

**Знакомство с html**

**HTML означает** «HyperText Markup Language» «Язык гипертекстовой рассылки»

В отличие от языков программирования (в т.ч. скриптовых), в которых с помощью сценариев программируются функции, в языке разметки с помощью тегов программируется контент (содержимое)

**Структура веб-страниц**

Любому уважающему себя веб-дизайнеру необходимо научиться программировать на языке HTML. Более того, именно со знания HTML должно начинаться любое знакомство с премудростями веб-дизайна.

**Веб-дизайн сегодня:**

* HTML – структура
* CSS - оформление
* JavaScript - поведение
* PHP и похожие языки – серверная часть
* CMS – управление контентом

**2.1 УРОК**

**Тег <HTML>**

Хотя за годы существования HTML появилось много версий языка, основные его принципы остаются неизменными. Структура HTML – документа удобно рассмотреть на примере бутерброда. Подобно тому как бутерброд часто имеет два кусочка хлеба , у HTML – документа есть два тега: открывающий и закрывающий. Между этими тегами, как начинка бутерброда между кусочками хлеба, помещается весь остальной код.

<html> весь остальной код </html>

**Тег <head>**

За открывающим тегом следует заголовок документа, который определяется открывающими и закрывающими тегами head. Заголовок содержит все невизуальные элементы HTML – документа, которые управляют отображением страницы.

**Тег <body>**

За тегом заголовка head следует тег тела документа body. В теги body заключаются все элементы, отвечающие за визуальное и структурное оформление документа. Заголовки, параграфы, списки, цитаты, изображения, ссылки – это лишь некоторые элементы, которые могут помещаться между тегами body. Тег body обрамляет основное содержание HTML – документа.

**3.1 УРОК**

**HTML – файл**

HTML – файлы – обычные текстовые файлы, поэтому Вы можете создать свою первую веб страницу в любом текстовом редакторе. Их существует множество, выбираем любой удобный.

В текстовом редакторе давайте составим HTML – документ с простейшей структурой и добавим «некоторый текст» в тело документа. Не забываем сохранять файл! Название HTML – файлов должны заканчиваться на .html или .htm.

**Тег <litle>**

Чтобы поместить заголовок страницы на вкладку браузера, нужно в блок документа head добавить элемент <litle>

**4.1 УРОК (27.09.2021)** Создаем Блог

**5.1 УРОК** тесты к модулю 1

**Основы HTML**

**6.1 УРОК**

**Заголовки в HTML – документе**

В HTML используется 6 уровней заголовков Н1…Н6 (Н1 самый крупный и самый важный)

Не рекомендуется использовать теги заголовков, когда нужно просто выделить текст жирным шрифтом. Поисковые системы придают заголовкам большое значение: используют их для индексации структуры страницы и ее контента.

**Горизонтальные линии**

Тег <hr> горизонтальная линия. В HTML 5, тег обрамляет тематический разрыв.

**Комментарии**

Комментарии браузером не отображаются, но они помогают в составлении документа и дают возможность добавлять описания, делать заметки, напоминания и прочее. <! текст комментария >

**7.1 УРОК (28.09.2021)**

**Параграф тег <p> </p>**   Браузеры автоматически добавляют пустую строку перед параграфом и после него.

**Перенос строки <br/>** добавляет в документ отдельную строку текста, когда нет надобности создавать новый параграф. Этот тег называют пустым элементом, там как он не требует закрывающегося тега.

**8.1 УРОК**

**Элементы форматирования**

В HTML используется ряд элементов для определения стиля текста. Элементы форматирования были созданы для отображения разных стилей текста

<p> Обычный текст </p>

<p><b> **Текст жирным шрифтом**  </b></p>

<p><strong> **Важный текст** </strong></p> это фразовый тег, он обрамляет важный текст

<p><big> Текст крупным шрифтом </big></p>

<p><i> *Текст курсивом* </i></p>

<p><em> Курсив (Важный!) </em></p>

<p><small> Текст маленьким шрифтом </small></p>

<p><sub> Подстрочный текст </sub></p>

<p><sup> Надстрочный текст  </sup></p>

<p><ins> Вставленный текст </ins></p>

<p><del> ~~Удаленный текст~~ </del></p>

**<b>** Текст жирный **</b>**

**<strong>** жирный и важный текст **</strong>**

**<i>** просто курсив  **</i>**

**<em>** курсив ( важность!) **</em>**

**9.1 УРОК** (занимались с блогом)

**10.1 УРОК (29.09.2021)**

**HTML – элементы**

HTML – элементы – это строительные блоки любого HTML – документа. HTML – элемент состоит из открывающего и закрывающего тегов, которые обрамляют его содержимое. HTML – документ может состоять из вложенных друг в друга HTML – элементов. Некоторые элементы очень короткие. Тег <br/> можно считать единым елементом, он не требует закрытия. В HTML сценарии пишутся с помощью элементов внутри элементов.

**11.1 УРОК**

**HTML – атрибуты**

Атрибуты содержат дополнительную информацию об элементе или теге, а также изменяют их. Большинство элементов имеют значения; значение изменяет атрибут. Атрибуты всегда пишутся в открывающемся теге.

**Единицы измерений в атрибутах**

Ширина элемента может быть определена с помощью пикселей или процентов. (рх или %)

**Атрибут выравнивания** указывает на то, как текст будет выровнен в документе. Если в одном теге протеворечивые атрибуты, то работать будет тот, кто первый прописан.

**12.1 УРОК**

**Тег <img>** используют для вставки изображения. Он содержит только атрибуты и не требует закрывающегося тега. Ссылка на изображение (веб-адрес) добавляется с помощью атрибута **src.** Атрибут **alt** устанавливает текст для изображений.

**Местоположение** изображения указывается между кавычками атрибута **src.**

**Изменения размера изображения** используют атрибуты **width** и **height**

**Рамка вокруг изображения** по умолчанию отсутствует, что бы ее добавить нужен атрибут **border**.

**13.1 УРОК (30.09.2021)**

**Нумерованные списки** начинаются с тега **<ol>** , а каждый пункт указывается с помощью тега **<li>**. Пункты списка будут автоматически пронумерованы цифрами. Маркером - тег **<ul>** .

**14.1 УРОК** (занимались с блогом)

**15.1 УРОК**

**Создаем таблицы**  тег **<table> </table>**

<tr></tr> строка таблицы

<td></td> столбцы таблицы. (Это контейнеры для текст, изображение ,списки, другие таблицы…)

<table **border="5">** атрибут заключает таблицу в рамку

<td **colspan="2">**  атрибут объединения ячеек (1я пожерает вторую)

<td **rowspan="2">** атрибут объединения строк 2 и более

<table **align="center">** таблица по центру

<td **bgcolor="red">** Красный </td> атрибут - ячейка залита красным цветом

**16.1 УРОК**

**Ссылки** важные кирпичики любой веб-страницы. Ссылки можно прикрепить к тексту или изображению, что бы пользователь при нажатии на них был перенаправлен на другую веб-страницу (или к другому файлу). В синтаксисе HTML ссылки обозначаются с помощью тега **<a href=" “> </a>,** атрибут href служит для указания самого адреса ссылки. Что бы сделать изображение ссылкой на другой документ, нужно вставить тег <img> в теги </a>.

Ссылки бывают абсолютными (ссылка на другой сайт) и относительными (название и расширение файла) .

Атрибут **target**="\_blank" , указывает, где нужно открыть привязанный документ. Blank – в новом окне или новой вкладке.

**17.1 УРОК (01.10.2021) мой блог**

**18.1 УРОК**

**Типы элементов:**

* **Блочные –** начинаются с новой строки. Например: (h1, from, li, ol, ul, p, pre, table, div и т.д.)
* **Строчные –** обычно пишутся без переноса строки. Например: (b, a, strong, smg, input, em, span и т.д. )

Элемент **<div>** - **блочный** элемент, часто используемый как контейнер для других элементов HTML. При использовании с CSS, элемент может задавать стиль **блокам контента.** (закрасить ячейку на пример)

Элемент **<span>** **- строчный** элемент, часто используемый как контейнер для текста. При использовании с CSS, элемент может задавать стиль **отдельных частей текста.** (другим цветом одно слово написать)

* **И блочные и строчные:**

1. **APPLET –** встроенное Java-приложение
2. **IFRAME –** встроенный фрейм
3. **INS –** вставленный текст
4. **MAP –** изображение - карта
5. **OBJECT –** встроенный объект
6. **SCRIPT –** встроенные в HTML-документ сценарии

Можно вставить строчные элементы внутри блочных элементов. Например, один тег div может иметь несколько тегов span внутри. НО! Блочные элементы не могут входить в состав строчных элементов.

**19.1 УРОК**

**Формы в** HTML используются для сбора информации от пользователя. **<form> </form>**

Атрибут **action=" action="url"** указывает на веб-страницу , которая должна загрузиться после отправки формы пользователем.

Атрибут **method**="GET" указывает на метод HTTP – запроса (**GET и POST**), который будет использоваться при отправке формы.

**<form action="url" method="GET">**

Когда используется метод GET, в URL-адресе веб-страницы будут отображаться формы.

**<form action="url" method=" POST ">**

Метод POST используется, когда требуется обновить данные в форме или когда в форме содержится конфиденциальная информация (пароли). Этот метод безопасней, потому что, передаваемые данные не видны пользователю в URL-адресе страницы.

Для того что бы получить данные пользователя, нужны соответствующие элементы формы, такие как текстовые поля. Элемент **<input>** имеет множество вариаций, что определяется атрибутом **type="”.** Например: **text, password, radio, URL, sumbit** и другие.

В атрибуте name узазываем название формы.

Если мы изменим тип ввода данных на type="radio", пользователь сможет выбрать только один вариант из представленного списка. type="checkbox" – будет возможность выбрать несколько вариантов.

При нажатии кнопки отправки, форма отправляется атрибуту action="” (GET или POST). После отправки формы, данные будут обработаны на сервере с использованием языка программирования, например, PHP.

**<textarea** name="message"> </textarea> несколько строк ввода

Что бы отправить работающие формы, необходимо написать код с серверной стороны (PHP)

**20.1 УРОК (мой блог)**

**21.1 УРОК**

**Цвет в** HTML задается с помощью шеснадцатеричного кода 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

0 – наименьшее значение

F – наибольшее значение

RGB (Red, Green, Blue) – это комбинация красного, зеленого и синего цветов.

Код цвета состоит из символа (#), за которым идет код из трех или шести символов.

#FF0000 – красный, #00FF00 – зелёный, #0000FF – синий, #С1250F – смешение оранжевого и красного……итд

Комбинаций есть больше 16 миллионов.

Атрибут <body **bgcolor="#000099**"> используют для изменения цвета фона веб-страницы.

<font color=**"#FFFFFF"** > Заголовок белым цветом </font>

**22.1 УРОК (02.10.2021)**

**Тег <frame>**

Веб страницу можно разделить на фреймы (кадры, окна) с помощью спец.документа. Отдельное окно (фрейм) создается с помощью тегов **<frame>,** которые заключаются в пару **<frameset> </frameset>.** Каждый тег <frame> внутри элемента <frameset> может содержать различные атрибуты:

**Border (рамка), scorolling (прокрутка), noresize (запрет изменять размер)** и т.д.

В элементе <frameset> назначается количество строк или столбцов фреймов, а также указывается размер каждого из них в % или хр

Содержимое фрейма описывается с помощью атрибута src.

Элемент <noframes> дает возможность просматривать веб-страницу в браузерах, которые не поддерживают фреймы. Внутри этого элемент может находится альтернативная страница с тегами body и любыми другими элементами.

**23.1 УРОК (мой блог)**

**24.1 УРОК (тесты к модулю)**

**25.1 УРОК (задачи)**

**HTML 5**

**26.1 УРОК** **(3.10.2021)**

**Новое в HTML 5**

Формы

* Спецификация Web Forms 2.0 расширила функциональные возможности форм, стало возможным создание более продуктивных страниц
* Добавлены элементы выбора даты, цвета, Numeric Stepper
* Новые поля для ввода: электронная почта, поиск и URL-адрес
* Добавлена поддержка формами методов PUT и DELETE

Встроенные API (Application Programming Interfaces интерфейсы программирования приложений)

* Функция «перетащи и отпусти»
* Проигрывание ауди и видеофайлов
* Автономные веб-приложения
* История
* Локальное сохранение
* Геолокация
* Веб-сообщения

**27.1 УРОК**

**Категории моделей контента**

В HTML элементы, как правило, были двух моделей: либо блочные, либо строчные.

В HTML 5 введены семь основных **моделей контента**

* Метаданные
* Встроенный контент
* Интерактивный контент
* Заголовочный контент
* Фразообразующий контент
* Потоковый контент
* Секционирующий контент

Модели контента HTML 5 были разработаны, что бы сделать структуру разметки понятней как для браузера, так и для веб-разработчика. Одни и те же элементы могут принадлежать к разным контентным моделям. Модели контента накладиваются друг на друга определенным образом, что зависит от способа их использования.

**Метаданные:** Контент, который определяет представление или поведение остального контента на веб-странице. Эти элементы находятся в элементе head документа.

*Элементы:* <base>, <link>, <meta>, <noscript>, <script>, <style>, <title>

**Встроенный контент:** Контент, который импортирует другие ресурсы вдокумент.

*Элементы:* <audio>, <video>, <canvas>, <iframe>, <img>, <math>, <object>, <svg>

**Интерактивный контент:** Контент, специально предназначенный для взаимодействия с пользователем.

*Элементы:* <a>, <audio>, <video>, <button>, <details>, <embed>, <iframe>, <img>, <input>, <label>, < object >, <select>, <textarea>

**Заголовочный контент:** Создает заголовки.

*Элементы:* <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <hgroup>

**Фразообразующий контент:** К этой модели принадлежат несколько строчных элементов, по аналогии HTML4.

*Элементы:* <img>, <span>, <strong>, <label>, <br />, <small>, <sub> и прочие.

**Потоковый контент:** Большинство HTML 5 – элементов, входящих в эту группу, отвечают за обычное функционирование документа.

**Секционирующий контент:** Отвечает за определение границ разделов на веб-странице (например, заголовков, подвала).

*Элементы:* <article>, <aside>, <nav>, <section>

**28.1 УРОК**

**Строение страницы в HTML 5**

Обычно HTML 5 – страница выглядит так:

|  |  |
| --- | --- |
| **<header>**  Содержит символику компании (логотип и т. п.) | |
| **<nav>**  Содержит навигационную секцию веб-сайта | |
| **<article>**  Основной контент веб-страницы | **<aside>**  Содержит второстепенный / сопутствующий контент или ссылки |
| **<section>**  Отдельный раздел основного контента |
| **<section>**  Отдельный раздел основного контента |
| **<footer>**  Информация об авторских правах, политике конфиденциальности, условий пользования и т. п. | |

Возможно, не все эти элементы могут понадобится – выбор зависит от структуры моей страницы.

**29.1 УРОК (5.10.2021)**

Элемент **<header>** определяет голову документа . Он помещается внутрь тега body. Он не имеет ничего общего с тегом <head>.

Элемент **<footer>** тоже широко используемый элемент. Обычно самая нижняя секция веб-страницы называется подвалом или футером. В эту пару тегов может заключатся:

* Контактная информация
* Политика конциденциальности
* Значки социальных медиа
* Условия предоставления услуг
* Информация об авторских правах
* Карта сайта и сопутствующие документы

Элемент **<nav>** этот тег отвечает за раздел страницы, в котором находятся ссылки на другие страницы или на другие разделы в пределах текущей страницы. Элемент предназначен в основном для больших разделов с ссылками. Обычно ссылки, которые не должны быть в элементе **<nav>**, добавляются в элемент **<footer>.**

**30.1 УРОК**

Элемент **<article>**  представляет автономный, независимый контент, который может использоваться и распространяться отдельно от своей страницы или сайта. Это может быть сообщение на форуме, журнал или статья из газеты или блога, комментарий, интерактивный виджет или гаджет или любой другой независимый элемент контента. Элемент <article>заменил элемент <div>, широко использовавшийся в HTML 4 вместе с атрибутами id или class.

Элемент <article> можно помещать внутрь другого элемента <article>. Такой вложенный элемент представляет контент, связанный с элементом снаружи. Например, комментарии к статье в блоге могут быть элементами <article>, который представляет ему статью.

Элемент **<section>** – это логический контейнер страницы или статьи. С его помощью содержимое элемента <article> разбивается на блоки, или разделы. Домашняя страница, к примеру, может содержать: раздел с представлением компании, раздел с новостями и еще один раздел с контактами.

Каждый блок <section> должен быть правильно указан: обычно проставляются теги заголовка (h1-h6), которые оформляются как его дочерние элементы.

Если нужно оформить контент элемента <section> в отдельный раздел, рекомендуется использовать элемент <article>.

Элемент **<aside>** служит для представления второстепенного или сопутствующего контента. Это контент, не зависимый от основного контента, но напрямую с ним связанный. Он обычно располагается в боковых панелях страницы.

Когда тег <aside>используется внутри тега <article>, контент <aside> должен иметь непосредственное отношение к контенту <article>.

Когда тег <aside>используется вне тега <article>, его контент должен иметь непосредственное отношение к окружающему его контенту.

**31.1 УРОК**

**Аудио в Интернете**

До HTML 5 специального стандарта для проигрывания звуковых файлов в браузере не существовало. С введением элемента <audio> в HTML 5 появился единый стандарт для встраивания звуковых файлов в веб-страницы. Указать путь к исходному звуковому файлу можно двумя способами. С использованием атрибута <source>.

В браузерах, не поддерживающих элемент <audio>, будет отображаться текст, помещенный между тегами <audio> и </audio>

Атрибут <audio **controls="">** указывает браузеру, что нужно отобразить элементы управления воспроизведением (кнопки начала/остановки воспроизведения и другие)

Атрибут <audio **autoplay=””>** воспроизведение звукового файла начинается сразу же после загрузки страницы без надобности в разрешении пользователя.

Атрибут <audio **loop=””>** при использовании звуковой файл будет воспроизводиться циклически.

На сегодняшний день элемент <audio> поддерживает три формата: MP3, WAV и OGG.

**32.1 УРОК**

**Видео в HTML**

Элемент <video> похож на элемент <audio>. Путь к исходному видеофайлу указывается с помощью атрибута source внутри элемента <video> или с использованием элементов source внутри элемента <video>. Другая общая особенность элементов <audio> и <video> это то, что разные браузеры поддерживают разные типы файлов. Если браузеру не удалось проиграть первый тип видео, он попытается воспроизвести следующий.

**Атрибуты <video>** Еще одна общая особенность элементов <audio> и <video> - это одинаковые атрибуты: controls, autoplay и loop.

На сегодняшний день элемент <video> поддерживает три формата: MP4, WebM и OGG.

**33.1 УРОК (05.10.2021)**

**Полоса выполнения**

Элемент <progress> используется для создания полосы выполнения на веб-странице. Его можно добавить внутрь заголовков, абзацев или в любой раздел в теле страницы.

**Атрибуты элемента** <progress>

Value – указывает степень выполнения задачи

Max/Min – указывает на общий размер задачи

Для динамического отображения выполнения задачи используйте тег <progress> в сочетании с JavaScript

**34.1 УРОК**

**HTML 5 и веб-хранение данных**

Благодаря тому, что в HTML 5 есть поддержка веб-хранения, данные веб-сайтов можно хранить на локальном компьютере пользователя. До появления HTML 5 в этих целях мы использовали куки-файлы JavaScript.

Преимущества веб-хранения данных:

* Безопасней
* Быстрее
* Возможность сохранять больший объем данных
* Хранимые данные не передаются при каждом серверном запросе

Локальное хранение данных привязано к домену. Все веб-страницы, зарегистрированы на одном домене, могут хранить и получать одни и те же данные.

**Виды объектов веб-хранения**, есть два типа:

* sessionStorage
* localStorage

Сравнение: сессионное и локальное хранилище

- в сессионном хранилище данные удаляются, как только пользователь закроет браузер

- в локальном хранилище данные хранятся без ограничения по времени

**Теперь о значениях**

Для локального и сессионного веб-хранения используются очень простые и похожие правила синтаксиса. Данные сохраняются в виде пары ключ/значение.

**Сохранение значения:**

localStorage.setItem(“key1”, “value1”);

**Извлечение значения:**

// что бы показать значение

alert(localStorage.getItem(“key1”));

**Удаление значения:**

localStorage.removeItem(“key1”);

**Удаление всех значений**:

localStorage.clear();

Такой же синтаксис используется и для сессионного хранения, с одним отличием: вместо localStorage используется sessionStorage.

**35.1 УРОК**

**Что такое API геолокации?**

В HTML 5 API геолокации используется для установления географического местоположения пользователя. Так как это ставит под угрозу конфиденциальность пользователя, API работает только с его разрешения.

Геолокация работает гораздо точней, если устройство оснащено GPS (смартфоны и т.д.).

**Как использовать геолокацию в HTML**

Основной метод API геолокации – getCurrentPosition, с помощью которого извлекается текущее географическое местоположение пользовательского устройства.

Navigator.geolocation. getCurrentPosition();

**Параметры:**

showLocation (обязательный) – назначает метод обратного вызова, с помощью которого извлекается информация о местоположении.

ErrorHandler (по желанию) – назначает метод обратного вызова, который будет использоваться при возникновения ошибки при обработке асинхронного вызова.

Option (по желанию) – назначает параметры извлечения информации о местоположении.

**Представление данных**

Местоположение пользователя может быть представлено в двух видах**: геодезическом и общественном.**

1. При геодезическом способе описания местоположение используется географические широта и долгота.
2. При общественном способе местоположение представляется в более доступном для специалиста виде.

Каждый параметр содержит данные как геодезического, так и общественного видов:

|  |  |  |
| --- | --- | --- |
| Атрибут | Геодезический | Общественный |
| Местоположение | 59.3, 18.6 | Стокгольм |
| Высота над уровнем моря | 10 метров | 4-й этаж |
| Направление движения | 234 градуса | Центр города |
| Скорость | 5 км/час | Пешеход |
| ориентация | 45 градусов | Северо-восток |

Метод getCurrentPosition () показывает объект только в случае успеха запроса. Широта, долгота и степень точности показываются всегда.

**36.1 УРОК (6.10.2021)**

**Создаем перетаскиваемые элементы**

Функция перетаскивания позволяет «захватить» объект или перетащить его в другое место. Чтобы элемент сделать перетаскиваемым, просто задайте атрибуту draggable значение "true".

<img draggable="true" />

Любой HTML-элемент может перетаскиваться.

**Что перетаскивать**

При перетаскивании элемента атрибут ondragstart вызывает функцию drag(event), которая определяет, какие данные должны быть перетащены.

Метод dataTransfer.setData() назначает тип данных и значение перетаскиваемых данных:

function drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

В нашем примере тип данных - "text", а значение - идентификатор ID перетаскиваемого элемента (изображения "image").

Куда сбросить

Событие ondragover указывает, куда перетаскиваемый элемент должен быть сброшен. По умолчанию данные и элементы не могут быть сброшены на другие элементы. Чтобы завершить перетаскивание, нам нужно отменить поведение элемента, установленное по умолчанию.

Для этого нужно назначить событию ondragover метод event.preventDefault().

Сбрасывание

Когда пользователь отпускает перетаскиваемый элемент, происходит событие drop.

В приведённом выше примере атрибут ondrop вызывает функцию drop(event):

**37.1 УРОК (07.10.2021)**

**Рисуем фигуры**

**SVG (Scalable Vector Graphics**, или масштабируемая векторная графика) - язык разметки в стиле HTML для рисования фигур.

Есть несколько методов рисования траекторий, прямоугольников, кругов, букв и графических изображений.

Векторная графика не привязана к пикселям, поэтому такие изображения могут быть увеличены бесконечно без потери качества.

**Вставка SVG – изображений**

Векторное изображение может быть вставлено в HTML-код с помощью простого тега изображения с атрибутом, указывающий путь к исходной картинке:

<img src="image.svg" alt="" height="300" />

SVG описывает векторную графику в формате XML.

**Рисуем круг**

Чтобы нарисовать фигуру с помощью **SVG - графики**, сначала нужно написать элемент svg с двумя атрибутами: шириной width и высотой height.

Если нужно нарисовать круг, добавьте тег **<circle>:**

<svg width="2000" height="2000"><circle cx="80" cy="80" r="50" fill="green" /></svg>

- cx - центр круга передвигается вправо по экрану

- cy - центр круга передвигается вниз по экрану

- r - задаёт радиус

- fill - задаёт цвет нашему кругу

- stroke - для создания контура

Каждый элемент и каждый атрибут в файлах SVG может быть анимирован.

**<rect>** нарисует прямоугольник:

**<line>** нарисует линию:

(x1, y1) задают координаты начала, (х2, у2) задают координаты конца линии.

**<polyline>** рисует фигуры, составленные из нескольких отрезков линий:

**Points** - координаты ломанной линии.

Код внизу нарисует чёрную галочку:

Атрибуты width и height элемента <rect> определяют высоту и ширину прямоугольника.

**Элементы <ellipse> и <polygon>**

Элемент <ellipse> пишется так же, как <circle>, с той лишь разницей, что:

Вы можете отдельно задавать координаты горизонтальной и вертикальной осей радиуса эллипса - с помощью атрибутов rx и ry.

**Многоугольник**

Элемент <polygon> используется для создания фигур с не менее, чем тремя сторонами. Элемент polygon стоит особняком, потому что он может автоматически замыкать фигуру.

Слово полигон происходит от греческого. "Poly" означает "много", а "gon" означает "угол".

**38.1 УРОК (09.10.2021)**

**Анимация фигур**

Анимация в SVG-графике создаётся с помощью элемента **<animate>.**

В приведённом ниже прямоугольник будет перемещаться в течение 3 секунд, а затем дважды повторит это действие:

**attributeName** - указывает, к какому атрибуту будет применяться анимация

**from** - назначает стартовое значение атрибута

**to** - назначает конечное значение атрибута

**dur** - назначает длительность анимации

**fill** - указывает, должно ли значение атрибута вернуться к своему первоначальному значению после завершения анимации (значения: "remove" сбрасывает значение; "freeze" - сохраняет значение "to")

**repeatCount** - назначает количество повторов анимации

В приведённом выше примере атрибут прямоугольника х изменяется от 0 до 300 в течении 3 секунд.

Чтобы анимация повторялась бесконечно, атрибуту repeatCount задаётся значение **"indefinite".**

**Траектории**

Элемент <path> используется для указания траектории.

С элементом path можно использовать следующие команды:

**M**: moveto

**L:** lineto

**H**: horizontal lineto

**V:** vertical lineto

**C**: curveto

**S**: smooth curveto

**Q**: quadratic Bézier curve

**T:** smooth quadratic Bézier curveto

**A:** elliptical Arc

**Z**: closepath

Траекторию указываем атрибутом **d:**

М устанавливает нашу «виртуальную ручку» в положение (0, 0). «Ручка» затем двигается по диагонали вниз и вправо на 200 пикселей, затем вверх в положение (200, 0). Команда Z замыкает фигуру, создаётся гипотенуза:

Во всех вышеперечисленных командах также могут использоваться строчные буквы. Заглавные буквы назначают абсолютное положение, буквы в нижнем регистре назначают относительное положение.

**39.1 УРОК**

**Элемент <canvas>**

С помощью элемента canvas в HTML рисуется графика всех типов сложности: от простых линий до сложных графических объектов.

Элемент canvas оформляется следующим образом:

<canvas id="canvas1" width="200" height="100"></canvas>

Элемент <canvas> является только контейнером для графики. Чтобы рисовать графику, используются сценарии (как правило, JavaScript).

**Элемент <canvas> должен иметь атрибут id для привязки в JavaScript:**

<html>

<head>

<script>

var can = document.getElementById("canvas1");

var ctx = can.getContext("2d");

</script>

</head>

<body>

<canvas id="canvas1"

width="400" height="300"></canvas>

</body>

</html>

С помощью метода getContext() получаем контекст рисования на холсте. Вам нужно познакомиться с азами JavaScript, чтобы понимать принцип работы с холстом в HTML5.

**Координаты холста**

Холст в HTML представляет собой двумерную сетку.

В верхнем левом углу холста находится начало его координат: (0,0).

По оси X значения возрастают к правому краю холста, а по оси Y - к нижнему краю холста.

**x**

**Y**

|  |
| --- |
| Canvas |

Элемент **<canvas>** является только контейнером для графики.

**Рисуем фигуры**

Посредством метода fillRect(x, y, w, h) рисуется «залитый» прямоугольник, при этом w - его ширина и h - его высота. По умолчанию используется чёрный цвет заливки.

Здесь на холсте в позиции (20, 20) будет нарисован чёрный прямоугольник размером 100х100 пикселей:

Свойство **fillStyle** задаст цвет, градиент или узор заливки рисунка.

С помощью этого свойства Вы можете нарисовать, например, зелёный прямоугольник.

На холсте в HTML можно рисовать с помощью и других методов:

**Нарисовать линию**

moveTo(x,y) - задаёт стартовую точку линии

lineTo(x,y) - задаёт конечную точку линии

**Нарисовать круг**

beginPath() - начинает рисовать

arc(x,y,r,start,stop) - задаёт параметры круга

stroke() - заканчивает рисовать

**Градиенты**

createLinearGradient(х, у, x1, y1) - создаёт объект линейного градиента

createRadialGradient(x,y,r,x1,y1,r1) - создаёт объект кругового градиента

**Нарисовать текст на холсте**

Font - задаёт параметры шрифта

fillText(text,x,y) - рисует «залитый» текст на холсте

strokeText(text,x,y) - рисует текст на холсте без заливки

Существует много других методов рисования фигур и иллюстраций на холсте.

**40.1 УРОК**

**Сравнение: холст и SVG – графика**

**Холст:**

- объекты рисуются программным способом

- рисование на основе пикселей

- анимация не встроена

- быстрая обработка операций рисования на основе пикселей

- нужно принимать во внимание разрешение

- нет поддержки обработчиков событий

- возможность сохранить получившееся изображение в формате .png или .jpg

- хорошо подходит для игр со сложной графикой

**SVG**

- элементы являются частью модели DOM (объектной модели документа) страницы

- рисование осуществляется с помощью векторов

- встроены эффекты, такие как анимация

- на основе более понятного стандартного синтаксиса XML

- не нужно принимать во внимание разрешение

- реализована поддержка обработчиков событий

- не подходит для игровых приложений

- лучше всего подходит для приложений с большой областью отображения (например, Google Maps)

В принципе, можно использовать SVG-графику и холст вместе на одной странице, если это необходимо.

Тем не менее, нельзя рисовать векторные изображения поверх холста или наоборот.

**41.1 УРОК**

**Работаем дальше с холстом**

Элемент canvas можно трансформировать. В качестве примера мы написали некоторый текст на холсте в позиции (20, 10).

Метод translate(x,y) используется для перемещения холста. х указывает, как далеко нужно переместить сетку по горизонтали, а у — как далеко нужно переместить сетку по вертикали.

Canvas включает несколько способов рисования контуров, кругов, окон, текста и добавления изображений.

**Метод rotate()**

Метод **rotate()** используется для вращения холста в HTML5. Параметр задаётся в радианах, а не градусах.

В примере внизу показан один и тот же прямоугольник до и после применения вращения:

Поворот влияет только на рисунки, созданные после завершения поворота.

**Метод scale()**

Метод scale() масштабирует текущий рисунок. Требует два параметра:

- во сколько раз изображение должно быть масштабировано вдоль оси X

- во сколько раз изображение должно быть масштабировано вдоль оси Y

В результате холст будет масштабирован в 1,5 раза вдоль оси X и в 4 раза вдоль оси Y:

Если вы масштабируете рисунок, все будущие рисунки также будут масштабированы.

**42.1 УРОК (13.10.2021)**

**Формы в HTML5**

HTML5 привносит много новых возможностей и совершенствований в технологию веб-форм. С целью повышения удобства просмотра веб-страниц были разработаны и добавлены новые атрибуты и способы ввода данных. Синтаксис веб-форм в HTML5 такой же, как был в HTML4:

Используйте атрибут novalidate, чтобы избежать проверки формы при отправке.

В HTML5 появился новый атрибут заполнителя **placeholder.** В элементах <input> и <textarea> этот атрибут содержит подсказку для пользователя о том, какая информация может быть введена в данное поле.

Атрибут **autofocus** ставит курсор в нужное поле ввода, когда форма загрузится:

Атрибут **required** указывает браузеру, что заполнение формы необходимо.

**Формы с обязательными полями**

Атрибут **required** используется, когда нужно сделать поля ввода обязательными для заполнения. Пользователь не сможет отправить форму, если не были заполнены все обязательные поля.

Атрибут **autocomplete** используется, когда нужно указать, будет ли работать функция автозаполнения формы.

Когда автозаполнение включено, браузер автоматически заполнит поля, исходя из ранее введённых пользователем данных.

**В HTML5 добавлены новые поля ввода input type:**

- color

- date

- datetime

- datetime-local

- email

- month

- number

- range

- search

- tel

- time

- url

- week

**Новые атрибуты полей ввода в HTML5:**

- autofocus

- form

- formaction

- formenctype

- formmethod

- formnovalidate

- formtarget

- height and width

- list

- min and max

- multiple

- pattern (regexp)

- placeholder

- required

- step

Поля ввода, не поддерживаемые старыми браузерами, будут восприниматься ими как поля ввода text.

**43.1 УРОК**

**Создаём окно поиска**

Новый тип поля ввода **search** служит для создания окна поиска:

Не забудьте указать атрибут name для <input>; в противном случае ничего не будет отправлено.

**Параметры поиска**

Тег **<datalist>** может быть использован, чтобы задать список предопределённых параметров поиска:

С помощью **<option>** создаются варианты выбора, которые будут предложены пользователю в выпадающем списке.

Идентификатор ID элемента datalist должен совпадать с атрибутом list в элементе input.

**Создаем другие поля**

Среди других новых типов полей ввода - **email, url** и **tel:**

Они будут особенно полезны при просмотре страницы на современном мобильном устройстве: устройство распознает тип поля ввода и откроет нужную клавиатуру.

Новые типы полей ввода облегчают сортировку и проверку данных, отправляемых в HTML-формах.

**44.1 УРОК (тесты)**