Введение в HTML, CSS и JavaScript.
Промтинг

Оглавление

Ссылки, где вы найдете очень много полезной информации

http://htmlbook.ru/

https://www.w3schools.com/

https://learn.javascript.ru/

Теоретическое описание.

Язык HTML.

Язык гипертекстовой разметки (HyperText Markup Language — HTML, рис.1), основной строительный блок веб-страниц, используется для создания и визуального представления веб-страниц. Он определяет содержание страницы, но не её функциональность.

Рис.1. – HyperText Markup Language.

HTML добавляет разметку в обычный текст. Гипертекст содержит ссылки, которыми веб-страницы связываются друг с другом, делая Всемирную паутину тем, чем она является сегодня. HTML поддерживает как изображения, так и другой медиаконтент. С помощью HTML каждый может создать статический, а также динамический сайт. HTML является языком, описывающим структуру и семантику содержимого веб-документа. Контент веб-страницы размечен с помощью тегов, представляющих HTML-элементы. Примерами таких элементов являются <img>, <title>, , , <picture> и так далее. Эти элементы формируют строительные блоки для любого веб-сайта.

Большая часть элементов HTML представляет собой элементы, сформированные тегами открытия и тегами закрытия (“ text here ”) этих элементов, как и в других языках форматирования. Не каждый элемент должен сопровождаться тегом закрытия, например – есть теги переноса строки
 и тому подобные.

Базовая структура HTML документа представлена ниже:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>HTML Document</title>
   </head>
   <body>
      <p>
         <b>
            Этот текст будет полужирным, <i>а этот – ещё и курсивным</i>.
         </b>
      </p>
   </body>
</html>
```

где основными элементами являются:

- **<!DOCTYPE html>** - тип документа;
- **<html>…</html>** - сам документ;
- **<head>…</head>** - заголовок документа, в котором указывается служебная информация, подключаются скрипты, указывается заголовок страницы и др.;
- **<body>…</body>** - тело документа, в котором содержатся непосредственно элементы отображаемой страницы.

Основные элементы HTML, их описание и назначение представлено в таблице 1.

Табл.1.- Основные элементы HTML.

Основные теги	
<html></html>	Указывает программе просмотра страниц, что это HTML документ.
<head></head>	Определяет место, где помещается различная информация не отображаемая в теле документа. Здесь располагается тег названия документа и теги для поисковых машин.
<body></body>	Определяет видимую часть документа
Теги оглавления	
<title></title>	Помещает название документа в оглавление программы просмотра страниц
Атрибуты тела документа	
<body bgcolor=?>	Устанавливает цвет фона документа, используя значение цвета в стандарте RGB - пример: FFFF00 - желтый цвет.
<body text=?>	Устанавливает цвет текста документа, используя значение цвета в стандарте RGB - пример: 00ff00 - зеленый цвет.
<body link=?>	Устанавливает цвет гиперссылок, используя значение цвета в стандарте RGB - пример: 00FF00 - зеленый цвет.
<body vlink=?>	Устанавливает цвет гиперссылок, на которыхх вы уже побывали, используя значение цвета в виде стандарте RGB - пример: 333333 - темно-серый цвет.
<body alink=?>	Устанавливает цвет гиперссылок при нажатии.
Теги форматирования текста	
<pre></pre>	Обрамляет предварительно отформатированный текст. (как есть!)
	Создает заголовок со шрифтом набольшего размера (как отдельный абзац)
	Создает заголовок со шрифтом наименьшего размера (как отдельный абзац)
	Создает жирный текст
	Создает наклонный текст
<tt></tt>	Создает текст - имитирующий стиль печатной машинки.
<cite></cite>	Используется для цитат, обычно наклонный текст.
	Используется для выделения из текста слова (наклонный или жирный текст)
	Устанавливает размер текста в пределах от 1 до 7.
	Устанавливает цвет текста, используя значение цвета в виде

	RRGGBB.
Гиперссылки	
ТЕКСТ	Создает гиперссылку на другие документы или часть текущего документа. Здесь URL адрес ссылки, ТЕКСТ - текст ссылки.
 < img src="imgURL" > 	Создает гиперссылку на рисунок, находящийся по адресу imgURL.
"URL" = "links/main.htm"	Адрес документа main.htm, находящегося в локальной папке links данного компьютера.
"URL" = "http://www.rambler.ru"	Ссылка на ресурс, находящийся на удаленном компьютере. В адресе присутствуют: программа связи с удаленным компьютером http (HyperText Transfer Protocol, разделители :// и интернет (IP) адрес искомого ресурса (в данном случае поискового сервера Rambler).
 	Создает гиперссылку вызова почтовой программы для написания письма по указанному адресу.
	Отмечает часть текста как место перехода по гипперссылке в документе.
	Создает гиперссылку на помечанную часть текущего документа.
Форматирование	
<p>	Создает новый параграф
<p align=?>	Выравнивает параграф относительно одной из сторон документа, значения: left, right, justify или center
 	Вставляет перевод строки.
<blockquote> </blockquote>	Создает отступы с обеих сторон текста.
<dl></dl>	Создает список определений. Штанга <dt>Основной снаряд в тяжелой атлетике. Состоит из стального грифа (стержня) со втулками на концах и свободно надевающихся металлических дисков. <dd>
<dt>	Определяет каждый из терминов списка
<dd>	Описывает каждое определение
	Создает нумерованный список
	Определяет каждый элемент списка и присваивает номер
	Создает ненумерованный список
	Предваряет каждый элемент списка и добавляет кружок или квадратик.
	Тег, используемый для форматирования больших блоков текста HTML документа. Часто используется в таблицах стилей

Использование кодов символов и специальных знаков в документе HTML

Код в документе HTML	**Выводится браузером на экран**
	неразрывный пробел
£	£
€	€
¶	¶
§	§

&copy;	©
&reg;	®
&trade;	™
&deg;	°
&plusmn;	±
&frac14;	¼
&frac12;	½
&frac34;	¾
&times;	×
&divide;	÷
&fnof;	*f*

Графические элементы	
<img src="name">	Добавляет изображение в HTML документ
<img src="name" align=?>	Выравнивает изображение к одной из сторон документа, принимает значения: left, right, center; bottom, top, middle
<img src="name" border=?>	Устанавливает толщину рамки вокруг изображения
<hr>	Добавляет в HTML документ горизонтальную линию.
<hr size=?>	Устанавливает высоту(толщину) линии
<hr width=?>	Устанавливает ширину линии, можно указать ширину в пикселях или процентах.
<hr noshade>	Создает линию без тени.
<hr color=?>	Задает линии определенный цвет. Значение RRGGBB.
Таблицы	
<table></table>	Создает таблицу.
<tr></tr>	Определяет строку в таблице.
<td></td>	Определяет отдельную ячейку в таблице.
<th></th>	Определяет заголовок таблицы (нормальная ячейка с отцентрованным жирным текстом)
Атрибуты таблицы	
<table border=#>	Задает толщину рамки таблицы.
<table cellspacing=#>	Задает расстояние между ячейками таблицы.
<table cellpadding=#>	Задает расстояние между содержимым ячейки и ее рамкой.
<table width=#>	Устанавливает ширину таблицы в пикселах или процентах от ширины документа.
<tr align=?> или <td align=?>	Устанавливает выравнивание ячеек в таблице, принимает значения: left, center, или right. (здесь right)
<tr valign=?> или <td valign=?>	Устанавливает вертикальное выравнивание для ячеек таблицы, принимает значения : top, middle, или bottom.
<td colspan=#>	Указывает кол-во столбцев которое объединено в одной ячейке. (по умолчанию=1)
<td rowspan=#>	Указывает кол-во строк которое объединено в одной ячейке. (по умолчанию=1)
<td nowrap>	Не позволяет программе просмотра делать перевод строки в ячейке таблицы.
Фреймы	
<frameset></frameset>	Предваряет тег <body> в документе, содержащем фреймы;
<frameset rows="value,value">	Определяет строки в таблице фреймов, высота которых определена кол-вом пикселов или в процентном

	соотношении к высоте таблицы фреймов.
<frameset cols="value,value">	Определяет столбцы в таблице фреймов, ширина которых определена кол-вом пикселов или в процентном соотношении к ширине таблицы фреймов.
<frame>	Определяет единичный фрейм или область в таблице фреймов.
<noframes></noframes>	Определяет, что будет показано в окне браузера если он не поддерживает фреймы.
Атрибуты фреймов	
<frame src="URL">	Определяет какой из HTML документов будет показан во фрейме.
<frame name="name">	Указывает Имя фрейма или области, что позволяет перенаправлять информацию в этот фрейм, или область из других фреймов.
<frame marginwidth=#>	Определяет величину отступов по левому и правому краям фрема; должно быть равно или больше 1.
<frame marginheight=#>	Определяет величину отступов по верхнему и нижнему краям фрейма; должно быть равно или больше 1.
<frame scrolling=VALUE>	Указывает будет ли выводится линейка прокрутки во фрейме; значение value может быть "yes," "no," или "auto". Значение по умолчанию для обычных документов - auto.
<frame noresize>	Препятствует изменению размеров фрейма.
Формы[1]	
[1] Для форм, выполняющих какие-то функции должны быть запущены соответствующие CGI скрипты на сервере. HTML создает только внешний интерфейс формы.	
<form></form>	Создает формы
<select multiple name="NAME" size=?></select>	Создает скролируемое меню. Size устанавливает кол-во пунктов меню, которое будет показано на экране, остальные будут доступны при использовании прокрутки.
<option>	Указывает каждый отдельный элемент меню
<select name="NAME"></select>	Создает ниспадающее меню
<option>	Указывает каждый отдельный элемент меню
<textarea name="NAME" cols=40 rows=8></textarea>	Создает окно для ввода текста. Columns указывает ширину окна; rows указывает его высоту.
<input type="checkbox" name="NAME">	Создает checkbox. За тегом следует текст.
<input type="radio" name="NAME" value="x">	Создает radio кнопку. За тегом следует текст.
<input type=text name="foo" size=20>	Создает строку для ввода текста. Параметром Size указывается длина в символах.
<input type="submit" value="NAME">	Создает кнопку "Принять"
<input type="image" border=0 name="NAME" src="name.gif">	Создает кнопку "Принять" - для этого используется изображение
<input type="reset">	Создает кнопку "Отмена"

Объектная Модель Документа (DOM)

Объектная Модель Документа (DOM, рис.2) – это программный интерфейс (API) для HTML и XML документов. DOM предоставляет структурированное представление документа и определяет то, как эта структура может быть доступна из программ, которые могут изменять содержимое, стиль и структуру документа. Представление DOM состоит из структурированной группы узлов и объектов, которые имееют свойства и методы. По существу, DOM соединяет веб-страницу с языками описания сценариев либо языками программирования.

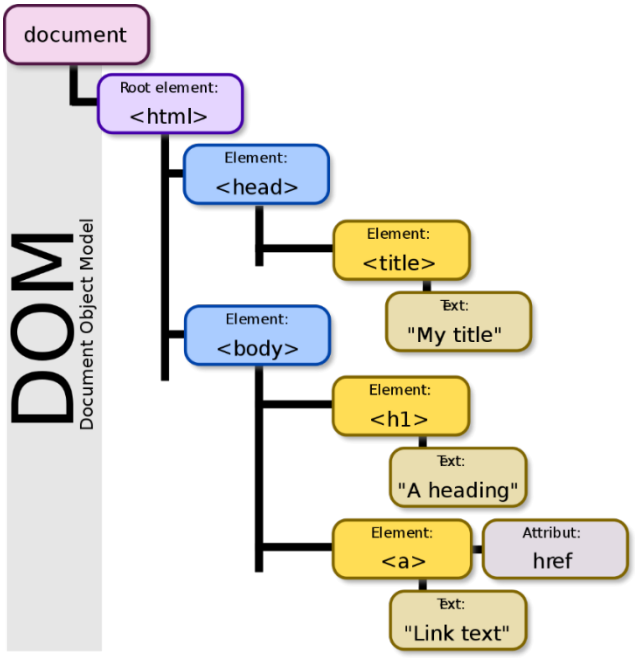


Рис.2. – Модель DOM.

Веб-страница – это документ. Документ может быть представлен как в окне браузера, так и в самом HTML-коде. В любом случае, это один и тот же документ. DOM предоставляет другой способ представления, хранения и управления этого документа. DOM полностью поддерживает объектно-ориентированнное представление веб-страницы, делая возможным её изменение при помощи языка описания сценариев наподобие JavaScript.

Стандарты W3C DOM и WHATWG DOM формируют основы DOM, реализованные в большинстве современных браузеров. Многие браузеры предлагают расширения за пределами данного стандарта, поэтому необходимо проверять работоспособность тех или иных возможностей DOM для каждого конкретного браузера.

Например: стандарт DOM описывает, что метод getElementsByTagName в коде, указанном ниже, должен возвращать список всех элементов <p> в документе.

```
paragraphs = document.getElementsByTagName("P");
// paragraphs[0] это первый <p> элемент
// paragraphs[1] это второй <p> элемент и т.д.
alert(paragraphs[0].nodeName);
```

Все свойства, методы и события, доступные для управления и создания новых страниц, организованы в виде объектов. Например, объект document, который представляет сам документ.

JavaScript

JavaScript изначально создавался для того, чтобы сделать web-странички «живыми». Программы на этом языке называются скриптами. В браузере они подключаются напрямую к HTML и, как только загружается страничка – тут же выполняются.

Программы на JavaScript – обычный текст. Они не требуют какой-то специальной подготовки. В этом плане JavaScript сильно отличается от другого языка, который называется Java.

Для выполнения программ, не важно на каком языке, существуют два способа: «компиляция» и «интерпретация»:

- *Компиляция* – это когда исходный код программы, при помощи специального инструмента, другой программы, которая называется «компилятор», преобразуется в другой язык, как правило – в машинный код. Этот машинный код затем распространяется и запускается. При этом исходный код программы остаётся у разработчика.
- *Интерпретация* – это когда исходный код программы получает другой инструмент, который называют «интерпретатор», и выполняет его «как есть». При этом распространяется именно сам исходный код (скрипт). Этот подход применяется в браузерах для JavaScript.

Современные интерпретаторы перед выполнением преобразуют JavaScript в машинный код или близко к нему, оптимизируют, а уже затем выполняют. И даже во время выполнения стараются оптимизировать. Поэтому JavaScript работает очень быстро.

Во все основные браузеры встроен интерпретатор JavaScript, именно поэтому они могут выполнять скрипты на странице. Но, разумеется, JavaScript можно использовать не только в браузере. Это полноценный язык, программы на котором можно запускать и на сервере, и даже в стиральной машинке, если в ней установлен соответствующий интерпретатор.

Современный JavaScript – это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется. Что же касается остальных возможностей – они зависят от окружения, в котором запущен JavaScript. В браузере JavaScript умеет делать всё, что относится к манипуляции со страницей, взаимодействию с посетителем и, в какой-то мере, с сервером:

- Создавать новые HTML-теги, удалять существующие, менять стили элементов, прятать, показывать элементы и т.п.
- Реагировать на действия посетителя, обрабатывать клики мыши, перемещения курсора, нажатия на клавиатуру и т.п.
- Посылать запросы на сервер и загружать данные без перезагрузки страницы (эта технология называется "AJAX").
- Получать и устанавливать cookie, запрашивать данные, выводить сообщения и др.

JavaScript – быстрый и мощный язык, но браузер накладывает на его исполнение некоторые ограничения. Это сделано для безопасности пользователей, чтобы злоумышленник не мог с помощью JavaScript получить личные данные или как-то навредить компьютеру пользователя.

Этих ограничений нет там, где JavaScript используется вне браузера, например на сервере. Кроме того, современные браузеры предоставляют свои механизмы по установке

плагинов и расширений, которые обладают расширенными возможностями, но требуют специальных действий по установке от пользователя.

Большинство возможностей JavaScript в браузере ограничено текущим окном и страницей. Список ограничений JavaScript представлен ниже:

- JavaScript не может читать/записывать произвольные файлы на жесткий диск, копировать их или вызывать программы. Он не имеет прямого доступа к операционной системе (рис.3).
- Современные браузеры могут работать с файлами, но эта возможность ограничена специально выделенной директорией – «песочницей». Возможности по доступу к устройствам также прорабатываются в современных стандартах и частично доступны в некоторых браузерах.
- JavaScript, работающий в одной вкладке, не может общаться с другими вкладками и окнами, за исключением случая, когда он сам открыл это окно или несколько вкладок из одного источника (одинаковый домен, порт, протокол).
- Есть способы это обойти, и они раскрыты в учебнике, но они требуют специального кода на оба документа, которые находятся в разных вкладках или окнах. Без него, из соображений безопасности, залезть из одной вкладки в другую при помощи JavaScript нельзя.
- Из JavaScript можно легко посылать запросы на сервер, с которого пришла страница. Запрос на другой домен тоже возможен, но менее удобен, т. к. и здесь есть ограничения безопасности.

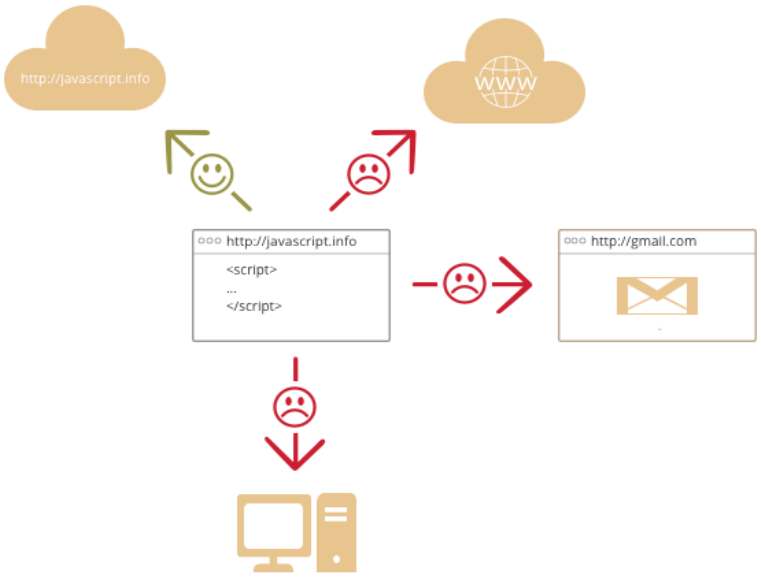


Рис.3. – Ограничения JavaScript.

Браузерные события

Для реакции на действия посетителя и внутреннего взаимодействия скриптов существуют события.

Событие – это сигнал от браузера о том, что что-то произошло. Существует много видов событий. Посмотрим список самых часто используемых, пока просто для ознакомления:

События мыши:

- click – происходит, когда кликнули на элемент левой кнопкой мыши;
- contextmenu – происходит, когда кликнули на элемент правой кнопкой мыши;
- mouseover – возникает, когда на элемент наводится мышь;
- mousedown и mouseup – когда кнопку мыши нажали или отжали;
- mousemove – при движении мыши.

События на элементах управления:

- submit – посетитель отправил форму <form>;
- focus – посетитель фокусируется на элементе, например нажимает на <input>.

Клавиатурные события:

- keydown – когда посетитель нажимает клавишу;
- keyup – когда посетитель отпускает клавишу.

События документа:

- DOMContentLoaded – когда HTML загружен и обработан, DOM документа полностью построен и доступен.

События CSS:

- transitionend – когда CSS-анимация завершена;
- многие другие.

Обработчик может быть назначен прямо в разметке, в атрибуте, который называется on<событие>. Например, чтобы прикрепить click-событие к input кнопке, можно присвоить обработчик onclick, вот так:

```
<input value="Нажми меня" onclick="alert('Клик!')" type="button"></input>
```

При клике мышкой на кнопке выполнится код, указанный в атрибуте onclick.

Фундаментальный недостаток описанных выше способов назначения обработчика – невозможность повесить *несколько* обработчиков на одно событие. Например, одна часть кода хочет при клике на кнопку делать ее подсвеченной, а другая – выдавать сообщение. Нужно в разных местах два обработчика повесить. При этом новый обработчик будет затирать предыдущий. Например, следующий код на самом деле назначает один обработчик – последний:

```
input.onclick = function() { alert(1); }
// ...
input.onclick = function() { alert(2); } // заменит предыдущий обработчик
```

Разработчики стандартов достаточно давно это поняли и предложили альтернативный способ назначения обработчиков при помощи специальных методов, которые свободны от указанного недостатка.

Методы addEventListener и removeEventListener

Методы addEventListener и removeEventListener являются современным способом назначить или удалить обработчик, и при этом позволяют использовать сколько угодно

любых обработчиков. Назначение обработчика осуществляется вызовом addEventListener с тремя аргументами:

```
element.addEventListener(event, handler[, phase]);
```

где:

- event – имя события, например click;
- handler – ссылка на функцию, которую надо поставить обработчиком;
- Phase – необязательный аргумент, «фаза», на которой обработчик должен сработать. Этот аргумент редко нужен, мы его рассмотрим позже.

Удаление обработчика осуществляется вызовом removeEventListener:

```
// передать те же аргументы, что были у addEventListener
element.removeEventListener(event, handler[, phase]);
```

Поиск элемента на странице

Есть 6 основных методов поиска элементов DOM, которые представлены в таблице 2.

Табл.2. – Основные методы поиска в JS.

Метод	Ищет по...	Ищет внутри элемента?	Поддержка
getElementById	id	нет	везде
getElementsByName	name	нет	везде
getElementsByTagName	тег или '*'	да	везде
getElementsByClassName	классу	да	кроме IE8
querySelector	CSS-селектор	да	везде
querySelectorAll	CSS-селектор	да	везде

CSS

CSS (Каскадные Таблицы Стилей, рис.4) позволяют создавать стилизованные и отформатированные веб-страницы, состоящие из слоев, блоков и так далее.

Рассмотрим, что такое CSS и как браузеры превращают HTML в (DOM), как CSS применяется к различным частям DOM, некоторые базовые примеры синтаксиса и подключение CSS к нашей веб-странице.

Рис.4. – CSS.

Документ обычно является текстовым файлом, который структурирован при помощи языка разметки. HTML это самый распространенный язык разметки, но вы можете столкнуться и с другими(SVG или XML).

Представление документа пользователю значит его конвертацию в понятную для пользователя форму. Браузеры, такие как Firefox, Chrome или Internet Explorer, предназначены для представления документов визуально, например, на экране компьютера, проектор или принтер.

Рассмотрим влияние CSS на HTML. Браузер применяет CSS правила к документу чтобы описать, как он будет отображаться. CSS-правила формируются из:

- *Набора свойств*, которые имеют значения, устанавливающие, как будет отображаться содержимое (HTML). Например, можно сделать, чтобы ширина элемента равнялась 50% ширины родительского элемента и его фон был красным.
- *Селектор*, который выбирает (англ. selects) элемент/элементы, к нужно применить измененные значения. Например, можно применить это CSS-правило ко всем параграфам в HTML-документе.

Набор правил CSS, содержащихся в таблице стилей (stylesheet), определяет, как должна выглядеть веб-страница.

Рассмотрим простой HTML-документ, содержащий <h1>и <p> (заметьте, что таблица стилей применяется к HTML с использованием элемента <link>):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Мой эксперимент с CSS</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>Это мой первый CSS-пример</p>
  </body>
</html>
```

Теперь давайте посмотрим на очень простой пример CSS, содержащий два правила:

```
<style>
    h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}

p {
  color: red;
}
</style>
```

Первое правило начинается с селектора h1, который означает, что оно будет применено к элементу <h1>. Оно содержит три свойства и три значения (каждая пара свойство/значение называется *объявление*):

1. Первое объявление меняет цвет текста на синий.
2. Второе выставляет желтый фон тексту.
3. Третье создает границу вокруг текста шириной 1 пиксель, сплошную(не пунктирную, не штрих-пунктирную) и окрашенную в черный цвет.

Второе правило начинается с селектора p, который значит, что оно применится к элементу <p>. Оно содержит одно объявление, которое меняет цвет текста на красный.

На рисунке 5 представлено как приведенный выше код отобразится в браузере.

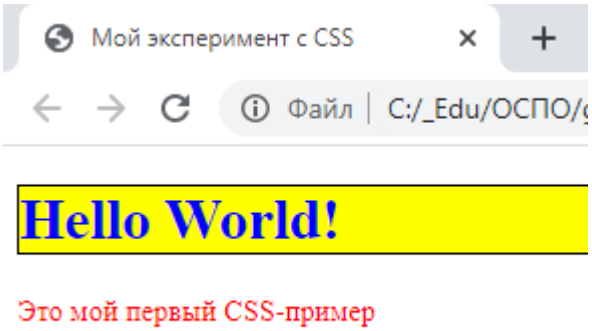


Рис.4. – CSS.

Основные виды селекторов

Основных видов селекторов всего несколько:

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

Селекторы можно комбинировать, записывая последовательно, без пробела:

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

Отношения

В CSS3 предусмотрено четыре вида отношений между элементами.

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

Есть и два более редких:

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

CSS свойства

Некоторые CSS свойства представлены в таблице 3.

Табл.3. – Свойства CSS.

Свойство	Описание	CSS
Внешний и внутренний отступы		
margin	Задает внешние отступы для элемента.	1
margin-bottom	Задает нижний внешний отступ для элемента.	1
margin-left	Задает левый внешний отступ для элемента.	1
margin-right	Задает правый внешний отступ для элемента.	1
margin-top	Задает верхний внешний отступ для элемента.	1
padding	Устанавливает внутренние отступы в элементе.	1
padding-bottom	Задает нижний отступ(внутреннее поле) для элемента.	1
padding-left	Задает левый отступ(внутреннее поле) для элемента.	1
padding-right	Задает правый отступ(внутреннее поле) для элемента.	1
padding-top	Задает верхний отступ(внутреннее поле) для элемента.	1
Позиционирование		
bottom	Указывает направление смещения позиционированного элемента от нижнего края.	2
clear	Указывает с какой стороны элемента не допускаются плавающие элементы.	1
clip	Определяет видимую часть абсолютно позиционированных элементов.	2
display	Указывает, как будет отображаться элемент в браузере.	1
float	Определяет будет ли элемент плавающим.	1
left	Указывает направление смещения позиционированного элемента от левого края.	2
position	Определяет метод позиционирования элементов.	2
right	Указывает направление смещения позиционированного элемента от правого края.	2
top	Указывает направление смещения позиционированного элемента от верхнего края.	2
visibility	Определяет, является ли элемент видимым.	2
z-index	Указывает порядок расположения элементов по оси Z.	2
Размер		
height	Устанавливает фиксированную высоту.	1
max-height	Указывает максимальную фиксированную высоту.	2
max-width	Указывает максимальную фиксированную ширину.	2

min-height	Указывает минимальную фиксированную высоту.	2
min-width	Указывает минимальную фиксированную ширину.	2
overflow	Определяет, что предпринять, если содержимое элемента превосходит размер области элемента.	2
overflow-x	Указывает, обрезать или нет левый/правый край содержимого - если оно переполняет доступную область элемента для его содержания.	3
overflow-y	Указывает, обрезать или нет верхний/нижний край содержимого - если оно переполняет доступную область элемента для его содержания.	3
resize	Указывает, может ли размер элемента изменяться пользователем.	3
width	Устанавливает фиксированную ширину.	1
Рамка и контур		
border	Позволяет использовать основные свойства границ в одном объявлении.	1
border-bottom	Позволяет использовать значения основных свойств для нижней границы рамки в одном объявлении.	1
border-bottom-color	Задает цвет для нижней границы рамки.	1
border-bottom-left-radius	Позволяет сделать округлую границу нижнего левого угла.	3
border-bottom-right-radius	Позволяет сделать округлую границу нижнего правого угла.	3
border-bottom-style	Определяет стиль для нижней границы рамки.	1
border-bottom-width	Определяет ширину для нижней границы рамки.	1
border-color	Задает цвет для границ рамки элемента.	1
border-image	Позволяет использовать изображение в качестве рамки.	3
border-left	Позволяет использовать значения основных свойств для левой границы рамки в одном объявлении.	1
border-left-color	Задает цвет для левой границы рамки.	1
border-left-style	Определяет стиль для левой границы рамки.	1
border-left-width	Определяет ширину для левой границы рамки.	1
border-radius	Позволяет изменить форму углов.	3
border-right	Меняет внешний вид правой границы рамки.	1
border-right-color	Задает цвет для правой границы рамки.	1
border-right-style	Определяет стиль для правой границы рамки.	1
border-right-width	Задает ширину для правой границы рамки.	1
border-style	Задает стиль для границ рамки элемента.	1
border-top	Меняет внешний вид верхней границы рамки.	1
border-top-color	Задает цвет для верхней границы рамки.	1
border-top-left-radius	Позволяет сделать округлую границу верхнего левого угла.	3
border-top-right-radius	Позволяет сделать округлую границу верхнего правого угла.	3
border-top-style	Определяет стиль для верхней границы рамки.	1

border-top-width	Определяет ширину для верхней границы рамки.	1
border-width	Задает ширину для границ рамки элемента.	1
outline	Создает внешнюю границу вокруг элемента.	2
outline-color	Определяет цвет внешней границы.	2
outline-offset	Сдвигает внешнюю границу на заданное расстояние от края элемента.	3
outline-style	Указывает стиль для внешней границы.	2
outline-width	Указывает ширину для внешней границы.	2
Текст		
color	Изменяет цвет текста.	1
direction	Определяет направление написания текста.	2
letter-spacing	Контролирует расстояние между символами в тексте.	1
line-height	Определяет межстрочный интервал(интерлиньяж).	1
quotes	Определяет тип кавычек для встроенных цитат.	2
text-align	Указывает способ выравнивания содержимого по горизонтали.	1
text-decoration	Добавляет некоторые элементы декорирования к тексту.	1
text-indent	Определяет отступ первой строки в тексте элемента.	1
text-overflow	Указывает, что должно произойти, когда текст переполняет содержащий элемент.	3
text-transform	Контролирует использование строчных и прописных букв в тексте.	1
vertical-align	Определяет вертикальное выравнивание в элементе.	1
white-space	Определяет способ обработки пробелов внутри элемента.	1
word-break	Определяет правила переноса для не-CJK сценариев.	3
word-spacing	Определяет ширину пробелов между словами.	1
word-wrap	Позволяет прерывать длинные слова для переноса на другую строку.	3
Тени и прозрачность		
box-shadow	Добавляет эффект отбрасывания тени к элементу.	3
opacity	Устанавливает уровень прозрачности элемента.	3
text-shadow	Создает тень для текста.	3
Фон		
background-color	Устанавливает цвет фона для элемента.	1
Шрифт		
font	Изменяет стандартный вид текста.	1
@font-face	Позволяет использование любого шрифта на странице.	3
font-family	Указывает шрифт или семейство шрифтов для текста.	1

font-size	Указывает размер для шрифта.	1
font-size-adjust	Контролирует размер неосновных шрифтов.	3
font-stretch	Регулирует ширину текста.	3
font-style	Позволяет изменять стиль текста.	1
font-variant	Конвертирует строчные буквы в прописные уменьшенного размера.	1
font-weight	Задает ширину символов текста.	1

Практическая часть

1.Переменные и функции

Функции в JS объявляются следующим образом:

```
function calcD(a, b, c) {
    return b*b - 4*a*c;
 }
```

Функция calcD считает дискриминант, попробуйте её запустить.

Индивидуальное задание 1

Реализуйте одну из следующих функций:

1. Нахождение минимума двух чисел. **min(a,b)**
2. Нахождение максимума двух чисел. **max(a,b)**
3. Проверка на равенство двух чисел. **equal(a,b)**

2.Взаимодейтсвие с пользователем

Рассмотрим три функции:

- **alert(“**текст**”)** – выводит на экран окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмёт «OK».
- **prompt(“**заголовок**”, ”**значение в строке по умолчанию”**)** – возвращает то, что ввёл посетитель – строку или специальное значение null, если ввод отменён.
- **confirm(“**вопрос**”)** – выводит окно с вопросом “вопрос” с двумя кнопками: OK и CANCEL. Результатом будет true при нажатии OK и false – при CANCEL (Esc).

Запустите следующий скрипт:

```
<script>
    if (confirm('Доброго времени суток, у этой страницы есть к вам вопрос, вы гот
овы ответить?')){
        var name = prompt('Как вас зовут?','Введите сюда ваше имя');
        alert("Ваше имя - "+name)
    }
    alert("Спасибо, что уделили время")
</script>
```

Индивидуальное задание 2

Объявите две переменные: admin и name.
Запишите в name строку свое имя, например "Алексей".
Скопируйте значение из name в admin.
Выведите admin (должно вывести «Алексей»).

Индивидуальное задание 3

Напишите скрипт, который будет спрашивать ваш возраст и после ввода возраста уточнять его, как это представлено на рисунке 5.

Если нажать **Отмена**, то скрипт попросит ввести возраст снова, если нажать **ОК**, то завершит своё выполнение.

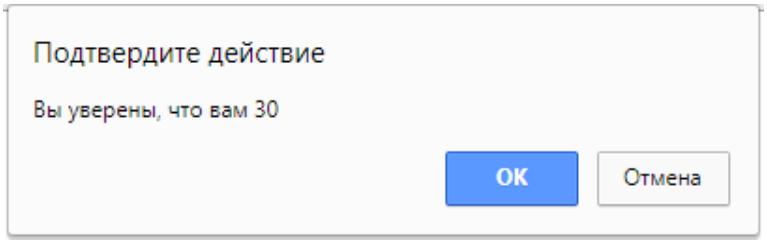


Рис.5. – Уточнение возраста.

3.Стилизация страницы

Суть задания заключается в том, что бы сделать страницу красивой. Для этого используйте CSS (CSS пишется в теге <style>…</style>). Откройте разметку и стили, представленные на следующей странице в браузере, и посмотрите, как это работает.

Используйте следующий код:

```
<body>
    <h1>Конспект курса</h1>
<p class="learned-ok">Парные теги.</p>
<p class="learned-ok">Одиночные теги.</p>
<p class="learned-ok">Атрибуты тегов.</p>
<p class="learned-ok">Инлайновые (встроенные) стили.</p>
<p class="learning-in-progress">Внешние стили.</p>
<p class="not-learned">Стилизация по классам.</p>
</body>
<style>
body {
font-family: Tahoma, serif;
}
h1 {
color: #999999;
}
.learned-ok {
color: green;
}
.learning-in-progress {
color: orange;
}
```

```
.not-learned {
color: red;
}
</style>
```

Так же стили можно указывать и в js, например, как это сделано в следующем коде (сделает текст элемента с id name зачеркнутым):

```
document.getElementsByTagId("name").style = "text-decoration:line-through"
```

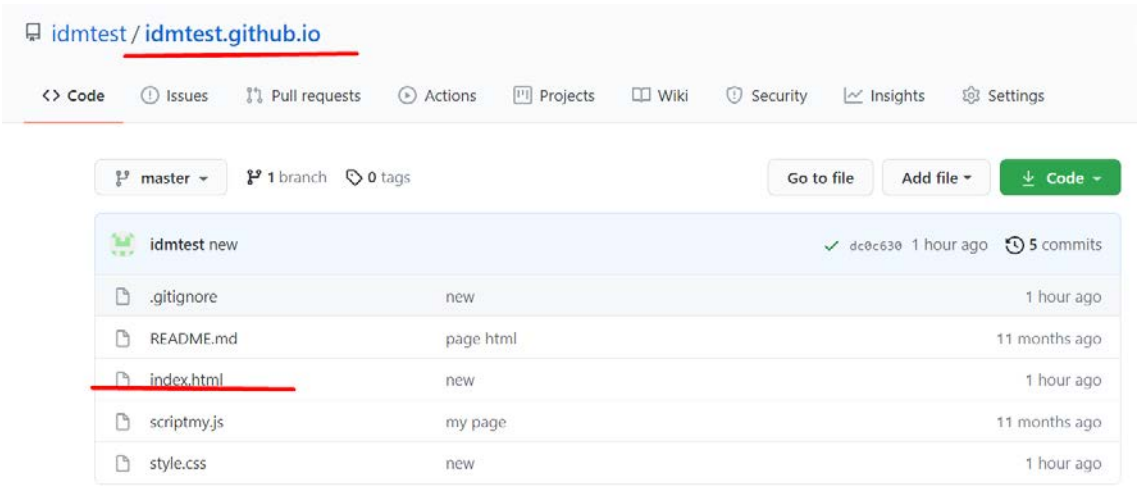
Индивидуальное задание 4

Добавьте CSS стили к странице из индивидуального задания 3, которые включают в себя следующие параметры:

- размеры шрифтов;
- тени;
- цвет текста;
- рамки;
- изменения цвета при наведении.

Дополните задание так, чтобы получилась страничка с вашим резюме, где будет имя, фамилия, фото (можно любую картинку). Также должно присутствовать минимум одно поле ввода и кнопка. По нажатию на кнопку из поля берется текст и изменяет текст фамилии. Можно добавить кнопку, которая меняет цвет фона на рандомный. Можете добавить и другие элементы на страницу.

Создайте репозиторий на github и назовите его yournick.github.io



Основной html файл должен называться index.hml. После того как вы отправите коммиты на github, ваша страница будет доступна по адресу https://yournick.github.io

Подробнее https://pages.github.com/

4.Взаимодейтсвие со страницей

Загрузите код страницы из репозитория на GithHub, расположенного по следующему адресу – https://github.com/Vchekryzhov/labs .

Тут стоит обратить внимание на:

- *document.getElementById()* с помощью этой функции мы получаем DOM элемент, который соответствует элементу на страницу, изменяя DOM в коде js он будет изменен также и на странице.
- *addEventListener(event,function(){})* с помощью этого метода можно добавить обработчик, который будет вызывать функцию при возникновении события event.

Индивидуальное задание 5 (необязательное)

Выполните любые 5 пунктов:

1. Добавить к анкете ещё два пункта: один - с чекбоксом, второй - с текстом.
2. Сделать проверку данных, при вводе, если пользователь вводит неверные данные, то повторять ввод до тех пор, пока не введет правильно. Имя – только текст, возраст – неотрицательные целые числа, пол – М или Ж.
3. Скрывать элемент «меня возьмут» пока пользователь не заполнит анкету.
4. Запретить изменение всех checkbox после того, как пользователь нажал кнопку «меня возьмут».
5. Скрыть элемент «заполнить анкету» после заполнения анкеты.
6. Изменить функцию *check()* так, что бы она показывала положительный результат, если у пользователя есть навыки: знание математики и знание программирования.
7. Изменить функцию *check()* так, что бы она показывала положительный результат, если у пользователя есть высшее образование и ещё два любых навыка.
8. Реализовать возможность редактирования имени, пола и возраста.

5.Промтинг

Когда вы выполнили базовые задания по HTML, CSS и JavaScript, важно познакомиться с современными инструментами, которые кардинально меняют процесс создания веб-проектов — интеллектуальными моделями (ИИ), такими как ChatGPT, Qwen Chat, Mistral Chat, Gemma 3 в LM Studio и другие.

ИИ-модели становятся неотъемлемой частью рабочего процесса разработчиков. Они умеют не только быстро находить решения типичных задач, но и создавать полноценные фрагменты кода, оптимизировать стили, исправлять ошибки и помогать в планировании структуры веб-приложений.

Для реализации большинства задач лабораторной отлично подойдет любая из бесплатных

https://chat.deepseek.com/

https://chat.qwen.ai/

https://chat.mistral.ai/chat

https://venice.ai/chat/

В контексте веб-разработки чаще всего используются модели **генеративного типа**:

- **ChatGPT** (от OpenAI) — мощная языковая модель, хорошо подходящая для генерации кода, объяснений, рецензирования проектов.

- **Qwen Chat** (от Alibaba) — высокоэффективная модель, ориентированная на технические задачи и помощь в программировании.
- **Mistral Chat** (от Mistral AI) — легковесная и быстрая модель для генерации кода и текстов.
- **Gemma 3** (запускаемая через LM Studio) — модель от Google DeepMind, подходящая для локального запуска и работы без подключения к интернету.

Эти модели обучены на огромных объемах данных, включая открытые коды, руководства и лучшие практики в области веб-разработки.

Какие задачи могут решать ИИ в веб-разработке?

1. **Генерация HTML-шаблонов** — создание базовых и сложных структур сайтов.
2. **Создание CSS-стилей** — оформление элементов, адаптивная верстка, анимации.
3. **Написание JavaScript-скриптов** — от простых манипуляций DOM до работы с API и создания динамических интерфейсов.
4. **Поиск и исправление ошибок в коде**.
5. **Оптимизация производительности сайта**.
6. **Создание небольших проектов под ключ** — лендинги, портфолио, интернет-магазины.

Таким образом, многие типовые задачи, которые раньше выполняли начинающие разработчики (**джуны**) и верстальщики, сегодня могут быть автоматизированы с помощью ИИ.

Почему senior-разработчики всё ещё незаменимы

1. **Понимание архитектуры проектов**.
   ИИ может написать фрагмент кода, но не способен спроектировать сложную, масштабируемую архитектуру веб-приложения.
2. **Интеграция сложных технологий**.
   Интеграция с внешними сервисами, продвинутая работа с базами данных, DevOps — всё это требует опыта и инженерного мышления.
3. **Критическое мышление** и **ревью кода**.
   Даже лучшие ИИ модели могут ошибаться. Senior-разработчики оценивают качество кода, его безопасность и эффективность.
4. **Постоянное обновление технологий**.
   Мир технологий меняется быстро. Только опытный специалист понимает, когда использовать новые инструменты, а когда стоит отказаться.
5. **Ответственность за проект**.
   ИИ не несёт ответственности за принятые решения. За успех проекта всегда отвечают люди.

Как правильно составлять промты для ИИ (ChatGPT, Qwen Chat, Mistral Chat и другие)

Промт — это текстовое задание для ИИ, которое определяет, каким будет результат. Чем яснее и структурированнее ваш промт, тем точнее и полезнее будет ответ.

Научиться правильно писать промты — значит научиться эффективно использовать ИИ в учебе и работе.

---

1. Определите цель

Перед тем как писать промт, ответьте себе на вопросы:

- Что именно вы хотите получить?
- В каком формате (код, текст, список, инструкция)?
- Какой уровень детализации вам нужен?

Пример:
❌ Плохая цель: "Сделай сайт"
✅ Хорошая цель: "Создай адаптивную страницу резюме на HTML, CSS и JS для размещения на GitHub Pages."

---

2. Будьте конкретными

Чем больше конкретики, тем лучше результат. Указывайте:

- Язык или технологии (например, HTML, CSS, JavaScript без библиотек).
- Стиль или оформление (например, минималистичный дизайн).
- Функциональные требования (например, кнопки, анимация скролла, форма обратной связи).

Пример:
"Сделай страницу с секциями: обо мне, опыт работы, образование, навыки. Добавь адаптивность и плавные анимации."

---

3. Определите ограничения

Если есть пожелания, что НЕ нужно использовать — укажите это.

- Без библиотек вроде Bootstrap.
- Без тяжелых фреймворков.
- Только чистый код.

Пример:
"Использовать только чистые HTML5, CSS3 и Vanilla JS. Без подключения сторонних библиотек."

---

4. Структурируйте промт

Разбейте задание на пункты:

- Что создать (страницу, приложение, скрипт).
- Какие секции и компоненты нужны.
- Требования к дизайну.
- Требования к интерактивности.
- Технические ограничения.
- Ожидаемый результат (например, файл index.html и комментарии в коде).

---

5. Пример хорошо составленного промта

Создай одностраничное резюме (index.html) для размещения на GitHub Pages.

Страница должна включать:

- Краткое описание пользователя (имя, должность, краткая биография).
- Блок с опытом работы (компания, должность, описание обязанностей).
- Блок образования (университет, специальность, годы).
- Список ключевых навыков.
- Кнопку "Скачать резюме" и форму обратной связи с базовой валидацией через JavaScript.

Требования:

- Адаптивная верстка с Flexbox и CSS Grid.
- Легкий минималистичный дизайн: светлый фон, четкая типографика.
- Без использования сторонних библиотек (Bootstrap и других).
- Плавные анимации появления секций при скролле.
- Чистый код с комментариями.

Ожидаемый результат: чистый index.html + встроенные стили и скрипты.

---

6. Частые ошибки при написании промтов

Ошибка	Как исправить
Слишком общее задание	Добавить конкретику, примеры и ограничения

Ошибка	Как исправить
Нет структуры	Разбить требования на пункты
Не указан формат результата	Попросить, например, "код в одном файле", "ответ в виде списка"
Забыли про ограничения	Четко написать, чего делать не нужно

---

7. Маленький чек-лист перед отправкой промта

- ✅ Я понимаю, что я хочу получить
- ✅ Я ясно изложил требования
- ✅ Я указал ограничения
- ✅ Я попросил нужный формат результата
- ✅ Промт не слишком длинный и не перегружен ненужными деталями

Итог

Хороший промт = половина успеха.
Четкие инструкции помогают ИИ давать более полезные, точные и практичные ответы.

Прокачайте навык создания промтов — и вы сможете использовать ИИ как настоящего помощника в учебе и карьере.

Индивидуальное задание 6

Создайте страничку резюме о себе. Добавьте различные интерактивные элементы: раскрывающиеся списки, выезжающие описания, таблицы. Лучше адаптивную верстку.

Сравните результаты с нескольких моделей: время генерации, размер кода, красота решения, адаптивность (перейти в режим разработчика в браузере – F12 – выбрать отображение на мобильном устройстве)

Пример промта (можете составить свой или улучшить этот):

Создай адаптивную веб-страницу index.html для личного резюме, предназначенную для размещения на GitHub Pages. Страница должна быть красивая, современный дизайн. Страница должна содержать следующие секции:

Краткая информация о пользователе (имя, должность, краткое описание).

Опыт работы (перечисление мест работы с должностью, компанией, сроками и коротким описанием обязанностей).

Образование (университет/курсы, специальность, годы обучения).

Навыки (список ключевых технических и "мягких" навыков).

Требования:

Сделай адаптивную верстку с использованием Flexbox и/или CSS Grid, чтобы страница выглядела хорошо на мобильных устройствах, планшетах и десктопах.

Добавь интерактивные элементы:

Кнопку "Скачать резюме" (псевдо-ссылка или заглушка для файла PDF).

Кнопку "Связаться со мной", которая плавно скроллит вниз к контактной форме.

Форму обратной связи (имя, email, сообщение) с базовой валидацией через JavaScript.

Используй чистые HTML5, CSS3 и JavaScript без сторонних библиотек (например, без Bootstrap).

Придерживайся современного минималистичного дизайна: светлый фон, четкая типографика, аккуратные отступы.

Добавь плавные анимации для появления секций при прокрутке страницы.

Код должен быть чистым, с комментариями, чтобы студент мог легко понять структуру и логику страницы.

Индивидуальные задания на повышенную оценку

Все должно быть красиво

1)Сделать калькулятор, выполняющий «+, -, *, /» и имеющий окно ответа, два окна ввода, на которых отображаются два числа, которые будут «+, -, *, /» между собой, кнопку «посчитать» и клавиатуру от 0 до 9. Если посчитанное число больше 15, в поле ответа калькулятора написать «число>15» и сделать фон этого поля красным. HTML+JS.

2)Сделать горизонтальный стрелочный (на 10 делений) индикатор на html-range, клавиатуру от 0 до 9. По нажатию на кнопку числа стрелочный индикатор должен показать это число. И если введенное число больше 5, цвет любого элемента должен стать красным (например фон страницы или самого стрелочного индикатора. HTML+JS.

3)Сделать форму для генерации HTML титульного листа для отчетов по лабораторным работам. На этой форме расположить следующие input поля: input кафедры, input группы, , input ФИО преподавателя, input ФИО студента; кнопку сгенерировать, которая делает редирект на получившийся титульный лист. Внешний вид должен быть как у стандартного титульного листа, но в формате HTML документа. HTML+JS.

Если вы делаете задание повышенной сложности, то либо создайте для него отдельный репозиторий, чтобы опубликовать страницу онлайн, либо объедините все задания в одно на одной странице (можно сделать пару страниц).

Список использованной литературы:

1. «Большая книга CSS3», Дэвид Макфарланд. Изд.: Питер, 2016.
2. «JavaScript», Дэвид Макфарланд. Изд.: Символ-Плюс, 2013.
3. «HTML, XHTML and CSS», Andy Harris. Изд.: Машиностроение, 2012.
4. Интернет-ресурсы.