Приднестровский государственный университет им. Т.Г. Шевченко
Инженерно-технический институт

РАЗРАБОТКА АДАПТИВНОЙ ВЁРСТКИ ВЕБ-САЙТА ПРИ ПОМОЩИ ТЕХНОЛОГИЙ HTML5 И CSS3

Лабораторный практикум

Разработал:
ст. преподаватель
кафедры ИТиАУПП
Бричаг Д.В.

г. Тирасполь
2020 г.

Лабораторная работа №4

Формы и её элементы в HTML5. Подключение JavaScript к странице

Цель работы: ознакомиться с расширенными элементами формы в новом стандарте. Создание формы с полями разных типов и проверкой их заполнения без скриптов. Подключение файлов скриптов и инициализация функций на языке JavaScript по стандарту ES6.

Теоретическая справка

Кроме тегов, классов и идентификаторов, в предыдущих работах мы сталкивались с обращением к псевдоклассам. В курсе лабораторных работ мы рассмотрим лишь самые популярные из них. Но на сложных страницах зачастую необходимы дополнительные средства гибкой настройки стилей к элементам. Важно знать о существовании целого ряда таких псевдоклассов и уметь искать среди них те, что необходимы для реализации текущей задачи.

Селектор псевдокласса

Псевдоклассы — это классы, фактически не прикрепленные к HTML-тегам. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу. Псевдоклассы характеризуют элементы со следующими свойствами:

:link — не посещенная ссылка;

:visited — посещенная ссылка;

:hover — любой элемент, по которому проводят курсором мыши;

:focus — интерактивный элемент, к которому перешли с помощью клавиатуры или активировали посредством мыши;

:active — элемент, который был активизирован пользователем;

:valid — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;

:invalid — поля формы, содержимое которых не соответствует указанному типу данных;

:enabled — все активные поля форм;

:disabled — заблокированные поля форм, т.е., находящиеся в неактивном состоянии;

:in-range — поля формы, значения которых находятся в заданном диапазоне;

:out-of-range — поля формы, значения которых не входят в установленный диапазон;

:lang() — элементы с текстом на указанном языке;

:not(селектор) — элементы, которые не содержат указанный селектор — класс, идентификатор, название или тип поля формы — :not([type="submit"]);

:target — элемент с символом #, на который ссылаются в документе;

:checked — выделенные (выбранные пользователем) элементы формы.

Селектор структурных псевдоклассов

Структурные псевдоклассы отбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках:

:nth-child(odd) — нечётные дочерние элементы;

:nth-child(even) — чётные дочерние элементы;

:nth-child(3n) — каждый третий элемент среди дочерних;

:nth-child(3n+2) — выбирает каждый третий элемент, начиная со второго дочернего элемента (+2);

:nth-child(n+2) — выбирает все элементы, начиная со второго;

:nth-child(3) — выбирает третий дочерний элемент;

:nth-last-child() — в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с :nth-child(), но начиная с последнего, в обратную сторону;

:first-child — позволяет оформить только самый первый дочерний элемент тега;

:last-child — позволяет форматировать последний дочерний элемент тега;

:only-child — выбирает элемент, являющийся единственным дочерним элементом;

:empty — выбирает элементы, у которых нет дочерних элементов;

:root — выбирает элемент, являющийся корневым в документе — элемент html.

Расчёт значения специфичности

Вы вдоволь наигрались со специфичностью, а теперь пришло время изучить полные правила её вычисления.

Специфичность селектора разбивается на 4 группы — a, b, c, d:

- если стиль встроенный, то есть определён как style="...", то a=1, иначе a=0;
- значение b равно количеству идентификаторов (тех, которые начинаются с #) в селекторе;
- значение c равно количеству классов, псевдоклассов и селекторов атрибутов;
- значение d равно количеству селекторов типов элементов и псевдо-элементов.

После этого полученное значение приводится к числу (обычно в десятичной системе счисления). Селектор, обладающий большим значением специфичности, обладает и большим приоритетом.

Посчитаем специфичность в нашем примере:

Селектор	a, b, c, d	Число
span	0, 0, 0, 1	1
ul.menu	0, 0, 1, 1	11
#myTopnav.topnav	**0, 1, 1, 0**	**110**
ul li	0, 0, 0, 2	2
.menu	0, 0, 1, 0	10
#myTopnav li	0, 1, 0, 1	101

Таблица 1 – Расчёт специфичности CSS селектора

Отсюда сразу видно, что в нашем примере самым приоритетным является селектор #myTopnav.topnav. Если два CSS-правила применяются к одному и тому же элементу и имеют одинаковую специфичность, то более приоритетным будет то правило, которое появится в коде позже другого.

Ещё один подход к взвешиванию селекторов и важные особенности поведения CSS можно прочитать в статье на Хабре: https://habr.com/ru/post/137588/

Формы в HTML5

HTML-формы являются элементами управления, которые применяются для сбора информации от посетителей веб-сайта.

Веб-формы состоят из набора текстовых полей, кнопок, списков и других элементов управления, которые активизируются щелчком мыши. Технически формы передают данные от пользователя удаленному серверу.

Для получения и обработки данных форм используются языки веб-программирования, такие как PHP, Perl.

До появления HTML5 веб-формы представляли собой набор нескольких элементов <input type="text">, <input type="password">, завершающихся кнопкой <input type="submit">. Для стилизации форм в разных браузерах приходилось прилагать немало усилий. Кроме того, формы требовали применения JavaScript для проверки введенных данных, а также были лишены специфических типов полей ввода для указания повседневной информации типа дат, адресов электронной почты и URL-адресов.

HTML5-формы решили большинство этих распространенных проблем благодаря наличию новых атрибутов, предоставив возможность изменять внешний вид элементов форм за счет CSS3.

Элемент <form>

Основу любой формы составляет элемент <form>...</form>. Он не предусматривает ввод данных, так как является контейнером, удерживая вместе все

элементы управления формы – поля. Атрибуты этого элемента содержат информацию, общую для всех полей формы, поэтому в одну форму нужно включать поля, объединенные логически.

Атрибут	Значение / описание
action	**Обязательный атрибут**, который указывает url обработчика формы на сервере, которому передаются данные. Представляет из себя файл (например, action.php), в котором описано, что нужно делать с данными формы. Если значение атрибута не будет указано, то после перезагрузки страницы элементы формы примут значения по умолчанию. В случае, если вся работа будет выполняться на стороне клиента сценариями JavaScript, то для атрибута action можно указать значение #. Также можно сделать так, чтобы заполненная посетителем форма приходила вам на почту. Для этого нужно внести следующую запись: <form action="mailto:адрес вашей электронной почты" enctype="text/plain"></form>
method	Задает способ передачи данных формы. Метод get передает данные на сервер через адресную строку браузера. При формировании запроса к серверу все переменные и их значения формируют последовательность вида www.any-site.ru/form.php?var1=1&var2=2. Имена и значения переменных присоединяются к адресу сервера после знака ? и разделяются между собой знаком &. Все специальные символы и буквы, отличные от латинских, кодируются в формате %nn, пробел заменяется на +. Этот метод нужно использовать, если вы не передаете больших объемов информации. Если вместе с формой предполагается отправка какого-либо файла, этот метод не подойдет. Метод post применяется для пересылки данных больших объемов, а также конфиденциальной информации и паролей. Данные, отправляемые с помощью этого метода, не видны в заголовке URL, так как они содержатся в теле сообщения. <form action="action.php" enctype="multipart/form-data" method="post"></form>
name	Задает **имя формы**, которое будет использоваться для доступа к элементам формы через сценарии, например, name="opros".

Таблица 2 – Атрибуты тега <FORM>

Создание полей формы

Элемент <input> создает большинство полей формы. Атрибуты элемента отличаются в зависимости от типа поля, для создания которого используется этот элемент.

С помощью css-стилей можно изменить размер шрифта, тип шрифта, цвет и другие свойства текста, а также добавить границы, цвет фона и фоновое изображение. Ширина поля задается свойством width.

Атрибут	Значение / описание
autofocus	Позволяет сделать так, чтобы в загружаемой форме то или иное поле ввода уже имело фокус (было выбрано), являясь готовым к вводу значения.
checked	Атрибут проверяет, установлен ли флажок по умолчанию при загрузке страницы для полей типа type="checkbox" и type="radio".
disabled	Отключает возможность редактирования и копирования содержимого поля.
form	Значение атрибута должно быть равно атрибуту id элемента <form> в этом же документе. Определяет одну или несколько форм, которым принадлежит данное поле формы.
list	Является ссылкой на элемент <datalist>, содержит его id.Позволяет предоставить пользователю несколько вариантов на выбор, когда он начинает вводить значение в соответствующем поле.
max	Позволяет ограничить допустимый ввод числовых данных максимальным значением, значение атрибута может содержать целое или дробное число. Рекомендуется использовать этот атрибут вместе с атрибутом min. Работает со следующими типами полей: number, range, date, datetime, datetime-local, month, time и week.
maxlength	Атрибут задает максимальное количество символов, вводимых в поле. Значение по умолчанию 524288 символов.
min	Позволяет ограничить допустимый ввод числовых данных минимальным значением.
multiple	Позволяет пользователю ввести несколько значений атрибутов, разделяя их запятой. Применяется в отношении файлов и адресов электронной почты. Указывается без значения атрибута.
name	Определяет имя, которое будет использоваться для доступа к элементу <form>, к примеру, в таблицах стилей css. Является аналогом атрибута id.

Таблица 3.1 – Атрибуты тега <INPUT>

Атрибут	Значение / описание
placeholder	Содержит текст, который отображается в поле ввода до заполнения (чаще всего это подсказка).
readonly	Не позволяет пользователю изменять значения элементов формы, выделение и копирование текста при этом доступно. Указывается без значения атрибута.
required	Выводит сообщение о том, что данное поле является обязательным для заполнения. Если пользователь попытается отправить форму, не введя в это поле требуемое значение, то на экране отобразится предупреждающее сообщение. Указывается без значения атрибута.
size	Задает видимую ширину поля в символах. Значение по умолчанию — 20. Работает со следующими типами полей: text, search, tel, url, email и password.
src	Задает url изображения, используемого в качестве кнопки отправки данных формы. Указывается только для поля <input type="image">.
step	Используется для элементов, предполагающих ввод числовых значений, указывает величину увеличения или уменьшения значений в процессе регулировки диапазона (шаг).
value	Определяет текст, отображаемый на кнопке, в поле или связанный текст. Не указывается для полей типа file.

Таблица 3.2 – Атрибуты тега <INPUT>. Продолжение.

Флажки и переключатели в формах

Флажки в формах задаются с помощью конструкции <input type="checkbox">, а переключатель — при помощи <input type="radio">.

Флажков, в отличие от переключателей, в одной форме может быть установлено несколько. Если для флажков указан атрибут checked, то при загрузке станицы на соответствующих полях формы флажки уже будут установлены.

Элемент <label> применяется при реализации выбора с помощью переключателей и флажков. Можно выбрать нужный пункт, просто щелкая кнопкой мыши на тексте, связанном с ним. Для этого нужно поместить <input> внутрь элемента <label>.

Атрибут	Значение / описание
type	button — создает кнопку.
	checkbox – превращает поле ввода во флажок, который можно установить или очистить
	color — генерирует палитры цветов в поддерживающих браузерах, давая пользователям возможность выбирать значения цветов в шестнадцатеричном формате.
	date — позволяет вводить дату в формате дд.мм.гггг.
	email — браузеры, поддерживающие данный атрибут, будут ожидать, что пользователь введет данные, соответствующие синтаксису адресов электронной почты.
	file — позволяет загружать файлы с компьютера пользователя.
	hidden — скрывает элемент управления, который не отображается браузером и не дает пользователю изменять значения по умолчанию.
	image — создает кнопку, позволяя вместо текста на кнопке вставить изображение.
	number — предназначено для ввода целочисленных значений. Его атрибуты min, max и step задают верхний, нижний пределы и шаг между значениями соответственно. Эти атрибуты предполагаются у всех элементов, имеющих численные показатели. Их значения по умолчанию зависят от типа элемента.
	password — создает текстовые поля в форме, при этом вводимые пользователем символы заменяются на звездочки, маркеры, либо другие, установленные браузером значки.
	radio — создает переключатель — элемент управления в виде небольшого кружка, который можно включить или выключить.
	range — позволит создать такой элемент интерфейса, как ползунок, min / max — позволят установить диапазон выбора
	reset — создает кнопку, которая очищает поля формы от введенных пользователем данных.
	search — обозначает поле поиска, по умолчанию поле ввода имеет прямоугольную форму.
	submit — создает стандартную кнопку, активизируемую щелчком мыши. Кнопка собирает информацию с формы и отправляет ее для обработки.
	text — создает текстовые поля в форме, выводя однострочное текстовое поле для ввода текста.
	time — позволяет вводить время в 24-часовом формате по шаблону чч:мм. В поддерживающих браузерах оно отображается как элемент управления в виде числового поля ввода со значением, изменяемым с помощью мыши, и допускает ввод только значений времени.
	url — поле предназначено для указания URL-адресов.

Таблица 3.3 – Наиболее часто используемые значения атрибута type тега <INPUT>.

Изначально JavaScript был создан, чтобы «сделать веб-страницы живыми». Программы на этом языке называются скриптами. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы.

Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска.

Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называющуюся «движком» JavaScript. У браузера есть собственный движок, который иногда называют «виртуальная машина JavaScript».

Современный JavaScript – это «безопасный» язык программирования. Он не предоставляет низкоуровневый доступ к памяти или процессору, потому что изначально был создан для браузеров, не требующих этого.

Возможности JavaScript сильно зависят от окружения, в котором он работает. Например, Node.JS поддерживает функции чтения/записи произвольных файлов, выполнения сетевых запросов и т.д.

В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Например, в браузере JavaScript может:

1. Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.

2. Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.

3. Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии AJAX и COMET).

4. Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.

Запоминать данные на стороне клиента («local storage»).

Практическая часть

Здесь мы продолжим наполнять нашу страницу интерактивными элементами. Одним из самых популярных на сегодня блоков является форма обратной связи. Их разновидности бывают разными:

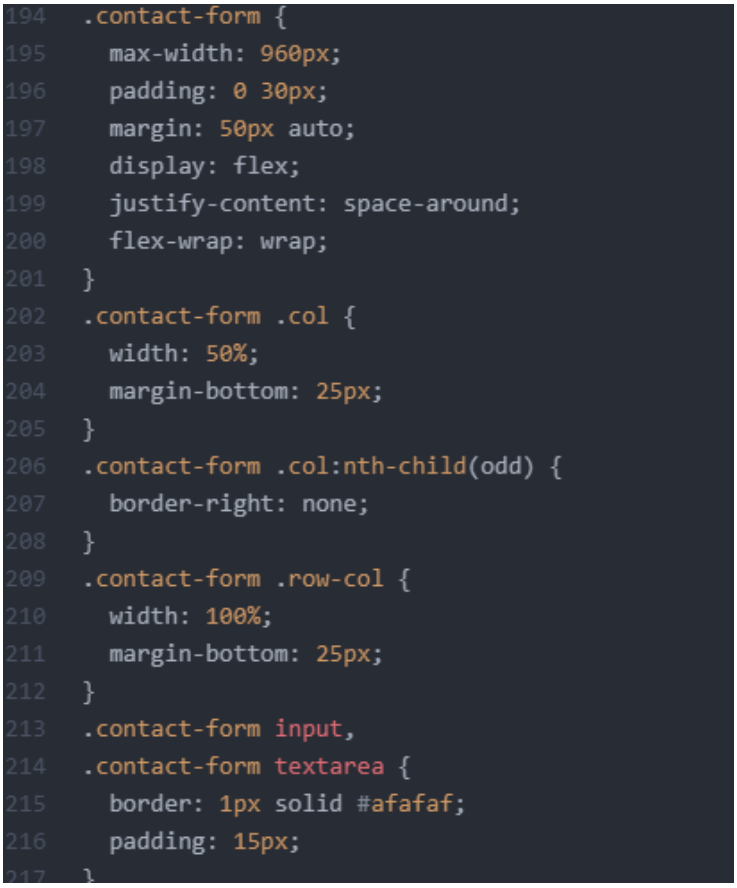
1. форма регистрации;
2. форма отправки сообщения;
3. форма быстрого заказа

Их объединяет одна задача, максимально удобно удержать клиента сайта на странице. Потому они являются важной части вашей страницы. Добавим такую секцию с контактной формой, где пользователь может отправить сообщение через сайт.

```
<section class="contact">
  <h2 class="main-title">Контактная форма</h1>
  <div class="delimiter"></div>
  <p class="main-title-description">С радостью ответим на любой вопрос</p>
  <form id="contactform" method="post" action="contact.php" class="contact-form">
    <input name="name" type="text" class="col" placeholder="Ваше имя *" required>
    <input name="email" type="email" class="col" placeholder="E-mail адрес *" required>
    <input name="theme" type="text" class="col" placeholder="Тема сообщения">
    <input name="phone" type="tel" class="col" placeholder="Номер телефона">
    <textarea name="comment" class="row-col" placeholder="Сообщение *" required></textarea>
    <input type="submit" id="submit" class="submit btn" value="Отправить сообщение">
  </form>
</section>
```

Рис. 43 – Создание контактной формы

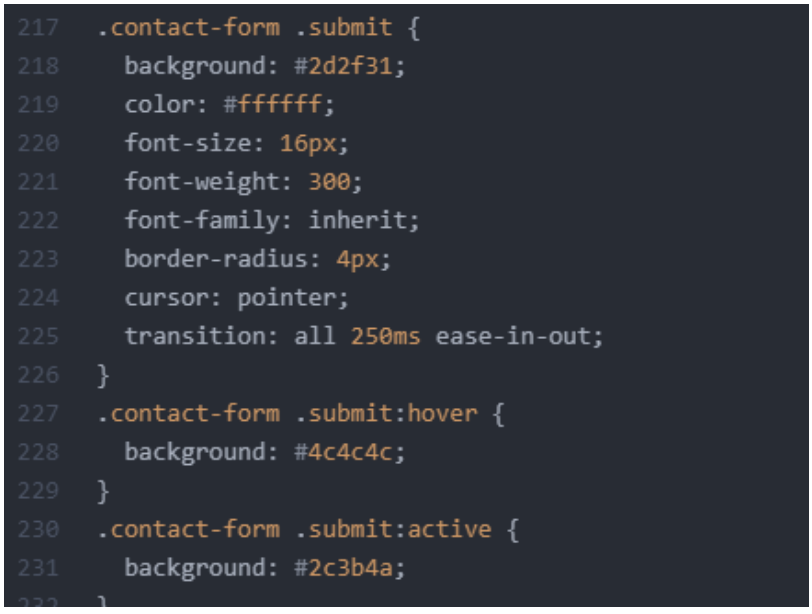
По умолчанию браузер применил системный внешний вид элементов формы. Обратите внимание на атрибуты **type** и **required**. Атрибут type означает какого типа поле, в то время как required объявляет поле обязательным к заполнению. Изменим внешний, добавив полям ввода стилей:



```
.contact-form {
  max-width: 960px;
  padding: 0 30px;
  margin: 50px auto;
  display: flex;
  justify-content: space-around;
  flex-wrap: wrap;
}
.contact-form .col {
  width: 50%;
  margin-bottom: 25px;
}
.contact-form .col:nth-child(odd) {
  border-right: none;
}
.contact-form .row-col {
  width: 100%;
  margin-bottom: 25px;
}
.contact-form input,
.contact-form textarea {
  border: 1px solid #afafaf;
  padding: 15px;
}
```

Рис. 44 – Стили контактной формы

А также сделаем кнопку более отзывчивой, добавив стили для событий наведения и нажатия на неё:



```
.contact-form .submit {
  background: #2d2f31;
  color: #ffffff;
  font-size: 16px;
  font-weight: 300;
  font-family: inherit;
  border-radius: 4px;
  cursor: pointer;
  transition: all 250ms ease-in-out;
}
.contact-form .submit:hover {
  background: #4c4c4c;
}
.contact-form .submit:active {
  background: #2c3b4a;
}
```

Рис. 45 – Стили кнопки и её событий

Для сохранения адаптивности для мобильных телефонов с диагональю с шириной экрана меньше 576px создадим следующие стили:

```
208   .contact-form .col:nth-child(odd) {
209     border: 1px solid #afafaf;
210   }
211   .contact-form .col {
212     width: 100%;
213     margin-bottom: 15px;
214   }
215   .contact-form .submit {
216     font-size: 14px;
217   }
```

Рис. 46 – Стили формы для мобильных устройств

В результате стилизованная форма будет лучше сочетаться с общим дизайном страницы. Результат внешнего вида формы изображён на рисунке 47.

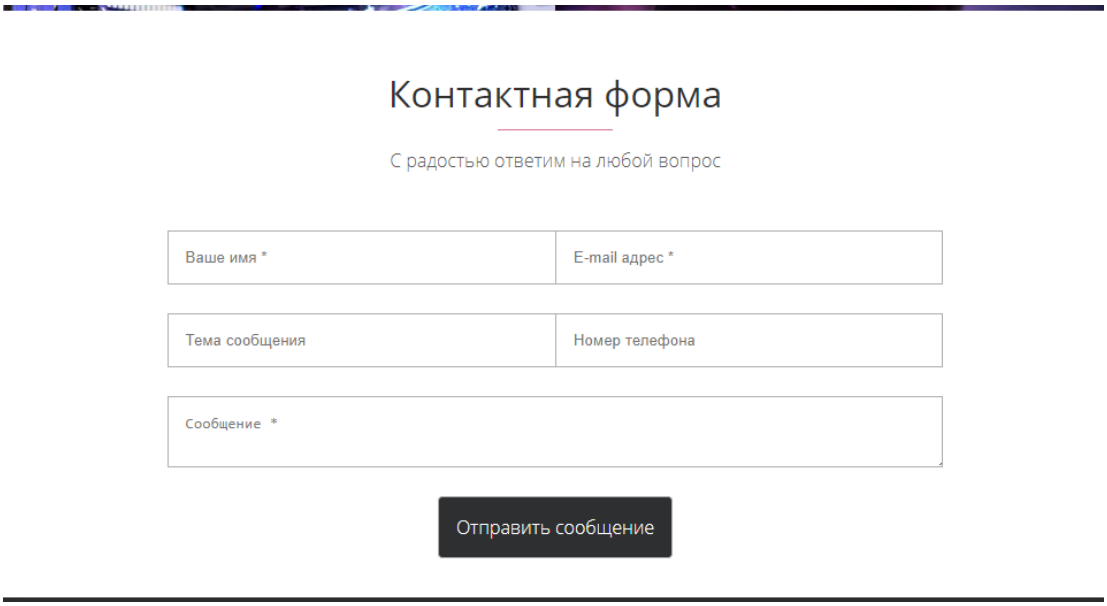


Рис. 47 – Стилизованная контактная форма

Так как мы задали атрибут **required** нескольким полям, то по нажатию кнопки форма будет требовать заполнения полей, а поскольку мы использовали

разные атрибуты для этих полей, то страница будет просить заполнить их корректным образом. Всплывающие подсказки отображает движок браузера при помощи JavaScript. Мы не можем их изменить, но при желании можем создать свои скрипты, которые будут себя вести схожим образом. Однако, сейчас не требуется писать скрипты, чтобы создавать простую валидацию. Достаточно лишь следовать стандарту HTML5.

Элементы input бывают не только для ввода текста и кнопки. В HTML реализовано множество интерактивных элементов, например группы переключателей (радиокнопки), флажки(чекбоксы), поле загрузки файла, поле ввода даты, сброса формы, поиска и так далее.

Для примера рассмотрим добавление чекбокса. Популярное решение: подтвердить принятие пользовательского соглашения или лицензии.

Добавим перед кнопкой отдельный блок, в нём два тега: label и input.

```
157             <span class="row-col check">
158               <input id="accept" type="checkbox" class="" required>
159               <label for="accept">Я прочитал(а) и принимаю условия
160                 <a href="#">Пользовательского соглашения</a></label>
161             </span>
```

Рис. 48 – Поле с флажком и подписью

Тут использован важный способ привязки подписи к полю. Через атрибут id задаётся уникальное имя поля, а для тега label посредством атрибута for указывается к какому полю относится эта подпись. В такой связке подпись становится активной и активирует то поле, к которому указано. Попробуйте кликнуть по надписи, флажок чекбокса должен переключаться.

```
218  .contact-form .check {
219    color: #616161;
220    text-align: center;
221  }
222  .contact-form .check a {
223    color: #ce416e;
```

Рис. 48 – Стилизация подписи поля

Стоит заметить, что, открыв страницу со смартфона, при тапе внутрь поля будет появляться разная раскладка клавиатуры: для поля с номером телефона – с набором цифр, для текстового – с текстом, для адреса – со значком @.

В заключении создадим подвал сайта, или футер. Контейнер для него мы создали в первой работе, добавим туда несколько блоков.

Во-первых, продублируем меню как показано на рисунке 48.

Во-вторых, добавим бренд сайта под меню. Картинка с логотипом уже есть, продублируем её здесь:

```
<div class="brand">
  <img src="img/fort-icon.png" alt="Логотип сайта">
  <span>AWESOME</span>
</div>
```

```
160      <footer>
161        <nav>
162          <ul>
163            <li><a href="#" title="На главную">Главная</a></li>
164            <li><a href="#" title="Общая информация">Общее</a></li>
165            <li><a href="#" title="Фотогалерея">Галерея</a></li>
166            <li><a href="#" title="Контактная информация">Контакты</a></li>
167          </ul>
168        </nav>
```

Рис. 48 – Меню для подвала сайта

В-третьих, добавим ссылки на социальные сети в виде брендовых иконок. Такой элемент сегодня является обязательным атрибутом любого сайта. Добавьте в католог img каталог social из методических материалов. А затем вёрстку с иконками, например:

```
<div class="social">
  <a href="#" title="Facebook" target="_blank">
    <img class="social-img" src="img/social/001-facebook.png" alt="Facebook">
  </a>
  <a href="#" title="LinkedIn" target="_blank">
    <img class="social-img" src="img/social/004-linkedin.png" alt="LinkedIn">
  </a>
  <a href="#" title="Instagram" target="_blank">
    <img class="social-img" src="img/social/005-instagram.png" alt="Instagram">
  </a>
  <a href="#" title="Twitter" target="_blank">
    <img class="social-img" src="img/social/007-twitter.png" alt="Twitter">
  </a>
  <a href="#" title="Tumblr" target="_blank">
    <img class="social-img" src="img/social/010-tumblr.png" alt="Tumblr">
  </a>
</div>
```

Рис. 49 – Социальные иконки в подвале сайта

И напоследок добавим строку копирайта:

© Very First Page 2020

После добавления элементов приступим к стилизации. Для футера и блока с брендом добавим:

```
footer {
  background: #2d2f31;
  padding-bottom: 10px;
}
.brand {
  display: flex;
  justify-content: center;
  align-items: center;
}
.brand span {
  font-size: 42px;
  font-weight: 400;
  color: white;
  font-family: "Roboto", "OpenSans-Light", Helvetica, Arial, sans-serif;
  line-height: 1;
  margin-left: 15px;
}
.brand img {
  width: 35px;
}
```

Рис. 50 – Стили подвала сайта

Для социальных кнопок и копирайта:

```
.social {
  display: flex;
  justify-content: center;
}
.social a {
  margin: 10px;
}
.social img {
  width: 30px;
}
.copyright {
  font-size: 14px;
  margin-top: 10px;
  text-align: center;
  color: rgba(255, 255, 255, 0.5);
}
```

Рис. 51 – Стили социальных кнопок в подвале

После сохранения, обновим страницу, а результат должен быть похож на тот, что изображён на рисунке 52.

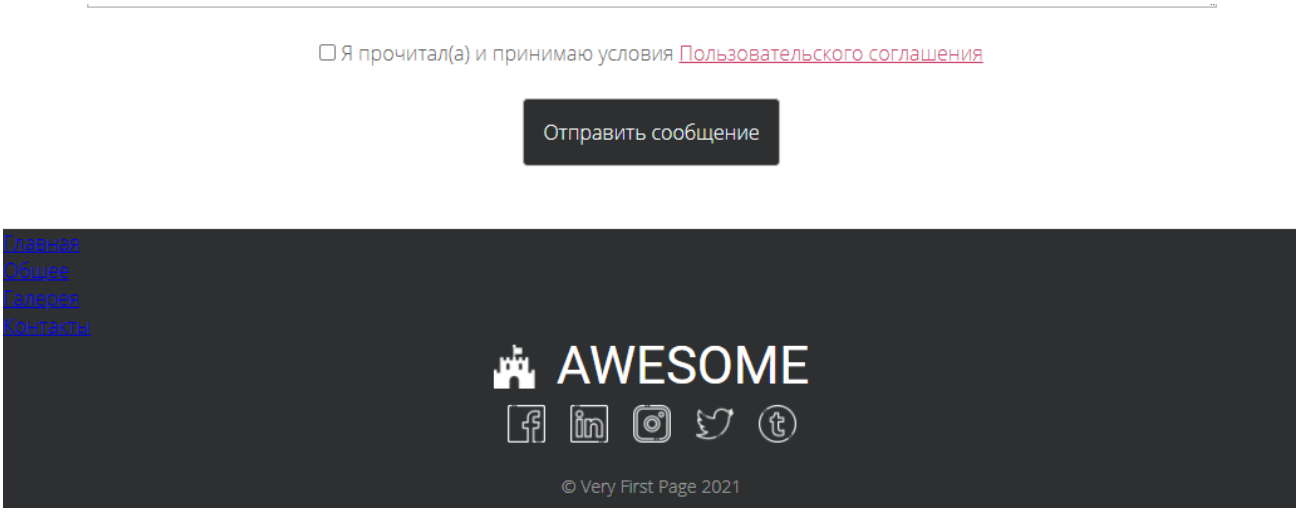


Рис. 52 – Внешний вид подвала страницы

Видно, что меню выглядит как стандартный список с ссылками, то есть необходимо повторить то, что мы делали ранее. Но сейчас вместо того, чтобы

скопировать стили для подвала, мы можем указать через запятую для каких тегов (или классов) мы хотим применить стили ещё.

Ниже на рисунке 53 показано, как указать множественные селекторы в CSS.

Рис. 53 – Множественный селектор для набора стилей

Теперь нас устраивает как выглядит меню в подвале, за исключением положения элементов. Давайте их расположим по центру, для этого добавим **ниже** стиль, который перезапишет свойство выравнивания:

```
footer ul {
  justify-content: center;
}
```

Так как оба меню находятся в уникальных блоках, то можно переназначить стиль даже без использования классов.

За хороший тон использования селекторов принято считать цепочку их 3 элементов. Если для применения свойства текущего селектора не хватает, а он уже состоит из трёх звеньев (например, ul li a), то стоит перейти на классы.

Важно не создавать избыточную цепочку селекторов (например, .menu .element .link) если для применения стиля достаточно использовать селектор только с одним классом. Это упростит в дальнейшем повторное использование классов и точную настройку вложенных стилей.

Проверим вёрстку для узких экранов, но там меню не отображается. Всё потому, что в предыдущей работе мы задали правила отображения для меню, но не указали, что только для меню в шапке. Поправим:

```
307  @media screen and (max-width: 992px) {
308    header nav li {
309      display: none;
310      padding: 15px 0;
311    }
```

Рис. 54 – Правки таблицы стилей

Отлично. Мы создали адаптивный лендинг с типовыми элементами. Но на сайте по-прежнему не работает мобильное меню. И тут средствами HTML5 CSS3 уже не обойтись. Тут нам приходит на помощь JavaScript. С его помощью мы можем оживить нашу страницу и сделать её более отзывчивой.

Но прежде мы начнём с ссылок самого меню, ведь они уже добавляют интерактив. Но не мешало бы это немного «освежить». Например, сейчас в нашем меню ссылки установлены с заглушками при помощи знака решётки. Допишем после знака решётки якоря:

```
<ul class="menu">
  <li><a href="#top" class="menu-element" title="На главную">Главная</a></li>
  <li><a href="#services" class="menu-element" title="Общая информация">Общее</a></li>
  <li><a href="#portfolio" class="menu-element" title="Фотогалерея">Галерея</a></li>
  <li><a href="#contact" class="menu-element" title="Контактная информация">Контакты</a></li>
</ul>
```

Рис. 55 – Ссылки в меню

А для наших секций установим соответствующие идентификаторы-якоря:

```
37 ▾    <main>
38 ›      <section id="top">…
41 ›      <section class="services" id="services">…
89 ›      <section class="portfolio" id="portfolio">…
145 ›     <section class="contact" id="contact">…
158     </main>
```

Рис. 56 – Идентификаторы разделов главной страницы сайта

После сохранения изменений и обновления страницы кликните по разделам меню: страница будет «перескакивать» прямо к разделам. Поправьте ссылки для меню из подвала сайта.

Идентификаторы, как мы знаем, могут использоваться не только как якоря, но и как селекторы для стилизации. Кроме того, к ним можно легко обращаться в JavaScript. Однако, стоит запомнить, что имя идентификатора должно быть уникальным на странице. В случае дублей, страница не сломается, но ссылку, скрипты и селекторы будут всегда обращаться к верхнему идентификатору из нескольких с одинаковым именем, поэтому вся их суть не возымеет никакого эффекта.

Также наше меню можно закрепить вверху страницы при прокрутке без скриптов. Для этого воспользуемся свойством position: sticky. В отличии от свойства position: fixed блок до прокрутки будет вести себя также, как если бы это свойство не было задано. А после прокрутки, будет закреплён к тому месту, куда укажем. Без указания позиции закрепления, это свойство работать не будет.

```
10  header {
11    background: #2d2f31;
12    position: sticky;
13    top: 0;
14    z-index: 1;
15  }
```

Рис. 57 – Закрепляем header страницы

Прекрасно, наше меню закреплено и можно быстро переходить к произвольным разделам страницы из любого места. Но на мобильных диагоналях списка меню до сих пор нет. Чтобы это исправить, подключим файл со скриптами. В head сайта уже прописан путь к файлу с расширением *.js. Если в вашем проекте его ещё нет – создайте его.

Добавим в него анонимную функцию к событию window.onload:

window.onload = function() {

}

Событие load на объекте window наступает, когда загрузилась вся страница, включая стили, картинки и другие ресурсы.

Добавим переменную:

const hamburger = document.getElementById('hamburger');

Она принимает значение элемента в дереве DOM с заданным идентификатором. Теперь можно присвоить к событию нажатия на этот элемент новую функцию, которая будет добавлять или удалять класс к меню:

index.html script.js

```
window.onload = function() {
  const hamburger = document.getElementById('hamburger');

  hamburger.onclick = function() {
    let topNav = document.getElementById('myTopnav');
    if (topNav.className === 'responsive') {
      topNav.className = '';
    } else {
      topNav.className = 'responsive';
    }
  }
}
```

Рис. 58 – Функция переключения меню

Следующим шагом добавим для мобильного меню на дисплеях до 992px с классом responsive стили:

```
header ul {
  position: absolute;
  left: 0;
  right: 0;
  background: #2d2f31;
  height: 0px;
  top: 57px;
  transition: all 250ms ease-in-out;
  flex-direction: column;
}
.responsive ul {
  height: 200px;
}
.responsive li {
  display: inline-block;
}
```

Рис. 59 – Стили меню для мобильных диагоналей

А к меню с тегом nav добавим идентификатор id="myTopnav" (к нему будем добавлять адаптивный класс). Проверяем работу функции отображения мобильного меню: оно должно «выезжать» сверху и «прятаться» обратно.

При помощи JavaScript мы можем сделать ещё один популярный метод плавного перехода к элементу на странице. Сейчас же, когда мы нажимаем на ссылки в меню, переход осуществляется мгновенно.

Для этого улучшим меню, добавив ссылкам вспомогательный атрибут data с параметром link и значением, соответствующим названию якоря.

```
<ul class="menu">
  <li>
    <a href="#top" class="menu-element" data-link="top" title="На главную">Главная</a>
  </li>
  <li>
    <a href="#services" class="menu-element" data-link="services" title="Общая информация">Общее</a>
  </li>
  <li>
    <a href="#portfolio" class="menu-element" data-link="portfolio" title="Фотогалерея">Галерея</a>
  </li>
  <li>
    <a href="#contact" class="menu-element" data-link="contact" title="Контактная информация">Контакты</a>
  </li>
</ul>
```

Рис. 60 – Изменения для меню

После этого в файле скриптов в уже существующую функцию, вызываемую событием загрузки страницы, добавим новую переменную:

const menuList = document.querySelectorAll('.menu-element');

В отличии от переменной hamburger menuList принимает значение всех элементов, удовлетворяющих условию селектора. К каждому событию клика всех элементов зададим функцию через цикл forEach.

```
13  const menuList = document.querySelectorAll('.menu-element');
14  menuList.forEach(function(element) {
15    element.addEventListener('click', function(event) {
16      event.preventDefault();
17      const elementLink = element.dataset.link;
18      document.getElementById(elementLink).scrollIntoView({ behavior: 'smooth'});
19    });
20  });
```

Рис. 61 – Закрепляем header страницы

В функции, вызываемой по клику на элемент сначала идёт обращение к объекту события event и вызывается его метод preventDefault(). Этот метод запрещает выполнение стандартного поведения нажатого элемента. В нашем случае он запрещает ссылке меню переходить к своему якорю, указанному в атрибуте href. Этот переход резкий, а мы ходим добиться плавной прокрутки.

Затем создаётся константа, принимающая значение атрибута data со значением link, который мы прописали в вёрстке. Таким образом можно хранить и передавать множество параметров из DOM в скрипты.

Полученное значение используем, чтобы найти указанный элемент в дереве, а метод scrollIntoView() с параметром behavior: ‘smooth’ выполняет плавную прокрутку к нему.

Попробуйте прописать элементам меню из подвала такие же классы и атрибуты, и они будут выполнять такие же действия, благодаря универсальному

селектору в функции JavaScript. В результате работы ваших скриптов клики по ссылкам будут плавно спускать страницу к выбранному разделу.

Однако, для работы с DOM деревом есть специальные инструменты, которые помогают быстро обращаться к нужным узлам. Рассмотрим это подробнее в следующей работе.

Задание на контроль

1. Повторить пример их практической части.

2. Для страницы примера должен быть доработано мобильное меню и прокрутка по ссылкам.

3. Объяснить, как были добавлены скрипты к странице.

4. Добавить медиазапросы для мобильного меню.

5. Лабораторная работа считается защищенной, если студент показал в окне браузера страницу из практической части со стилизованным меню и ответил правильно на все заданные контрольные вопросы (следующая лабораторная работа не подлежит защите до тех пор, пока не будет защищена предыдущая).