**Введение**

Со временем технологии проникают в разные сферы деятельности человека. Еще полвека назад на заводах было множество людей, работающих в цехах, занимающиеся тем, что сейчас делает один или несколько компьютеров. Раньше различным предприятиям и фирмам необходимо было вкладывать огромные средства, покупая рекламные места, подготавливая различные баннеры, информационные стенды и другие вещи, направленные на распространение товаров или услуг.

Огромные средства расходовали на поиски покупателей, чаще всего в своем же регионе, и только после успеха на местном рынке уже задумывались о распространении известности компании дальше, за пределы области. Нанимался огромный штат сотрудников, занимающийся продвижением и распространением продукции компании в каждом городе, а это огромные расходы. Тратились средства на распечатку рекламных материалов, буклетов, журналов и других средств привлечения клиентов.

С приходом в нашу жизнь интернета появилось много новых способов получать и передавать информацию. Сейчас практически любой человек в любое время может получить доступ к совершенно любой информации, надо лишь сформировать запрос.

Конечно, этим и воспользовались маркетологи различных компаний в разных сферах. С помощью всемирной паутины, именуемой интернетом, можно распространять любую информацию на огромные расстояния, тратя в разы меньше денег и других ресурсов, но получая ощутимый результат намного быстрей!

Любая компания стремится как можно громче рассказать о себе, быть услышанной и уведенной как можно большей публикой. Обычно с помощью сети Интернет компании рассказывают о себе и своих товарах, оставляя свои контакты. Такие сайты именуются сайтами-визитками. Они достаточно просты, содержат лишь несколько страниц информации. Обычно это история фирмы, особенно если она существует не первый год, а пару десятилетий, как многие предприятия нашей необъятной страны, краткая информация о текущем положении, какая продукция выпускается, как популярна она, кто ей пользуется из известных брендов или личностей, и контакты, по которым можно связаться для дополнительных вопросов. Часто такие сайты заказывают и обычные люди, специалисты в той или иной области, желающие получить себе резюме в такой форме. Это очень удобно, при должном профессионализме разработчиков такой сайт клиент может дополнять новой информацией по мере необходимости. Какая же польза от этого сайта? Другие люди, которые каким-либо образом узнали, например, имя и фамилию человека, выходят на его сайт и могут узнать о нем подробней. Если это специалист в какой-либо области .на сайте обычно представлены примеры работ, сроки, цены, контрактная информация. Ознакомившись с этим, потенциальные клиенты могут стать реальными, а владелец сайта получит дополнительную прибыль.

Практически любой сайт, если он направлен на увеличение прибыли компании или человека, при должном умении, быстро окупается. В наше время очень важно любой компании иметь собственный сайт, на котором представлена вся необходимая информация, ведь намного проще клиентам искать через интернет, чем ходить по различным офисам в поисках выгодного предложения. Экономия времени огромная, как для отдельного человека, так и для компании, наравне с издержками.

Помимо простых информационных сайтов существуют и такие, на которых можно онлайн, т.е. прямо на сайте, купить товары или услуги компании. Такие сайты называются интернет-магазинами. Они сложней в разработке, но для компаний, занимающихся продажей товаров или услуг, они очень выгодны и актуальны. Типичный функционал таких сайтов обычно следующих – каталог товаров, где представлена вся продукция компании, корзина, куда помещаются выбранные пользователем товары, функция выбора адреса и способа доставки, оплаты товара. На данный момент не каждый человек готов так легко и просто оплатить какую-то сумму онлайн, не удостоверившись в надежности, поэтому такие сайты часто предоставляют возможность оплатить покупку при получении на почте или в пункте выдачи, но все зависит от самой компании, которая владеет сайтом. Постепенно появляются все новые и новые способы доказать, что сайт надежен, а покупки защищены. Часто это шифрование, дабы было невозможно перехватить траффик и узнать платежные данные, перенаправление на страницу банка, где и происходит оплата покупки.

На данный момент каждая компания стремится обзавестись собственным сайтом различной сложности и наполненности. Благодаря этому другие компании, занимающиеся разработкой и поддержкой сайтов, будут получать за это деньги и опыт.

В рамках моей дипломной работы будут рассмотрены различные способы создания сайтов, разнообразные фреймворки, такие как Laravel 5.2, Yii 2, ZendFramework, Angular, Backbone, шаблонизаторBlade, способы размещения сайта в сети Интернет, настройка сервера для удобного администрирования и обновления сайта по мере необходимости.

**Анализ методов решения проблемы**

Современный рынок технологий насыщен огромным выбором различных технологий, помогающих создавать сайты любой сложности за короткий промежуток времени. Есть варианты использования готовых решений, где необходимо лишь немного изменить дизайн и наполнить сайт контентом. Так же можно найти технологии, которые предоставляют лишь небольшой набор функций, на основе которых можно быстрей построить сайт. Рассмотрим популярные решения.

**jQuery**

Данная библиотека используется очень часто, она практически незаменима в разного рода проектов, позволяя манипулировать DOM-деревом страницы достаточно легко.

Важнейшей составляющей данной библиотеки являются селекторы. С их помощью можно получить любой элемент на странице. Для того чтобы понимать как работает селектор Вам все-же необходимы базовые знания CSS, т.к. именно от принципов CSS отталкивает селектор jQuery:

* $("#header") — получение элемента с id=«header»
* $(«h3») — получить все <h3> элементы
* $(«div#content .photo») — получить все элементы с классом =«photo» которые находятся в элементе div с id=«content»
* $(«ul li») — получить все <li> элементы из списка <ul>
* $(«ul li:first») — получить только первый элемент <li> из списка <ul>

Начнем с простенького примера — слайд-панель, она у нас будет двигаться вверх/вниз по клику на ссылке

Начнем с простенького примера — слайд-панель, она у нас будет двигаться вверх/вниз по клику на ссылке. Реализуем это следующим образом, по клику на ссылку, у нас будет переключаться её класс (между «active» и «btn-slide»), а панелька с id=«panel» будет выдвигаться/прятаться. (класс «active» изменяет позицию фонового изображения, см. CSS файл во вложении).

$(document).ready(function () {

$(".btn-slide").click(function () {

$("#panel").slideToggle("slow");

$(this).toggleClass("active");

});

});

Другой пример – гармошка, популярный элемент на разных сайтах.

Теперь приступим к разбору полетов:

Первой строчкой мы добавляем класс «active» первому элементу <h3> внутри <div class=«accordion»> (класс«active» отвечает за позиционирования фонового рисунка — иконки со стрелочкой). Во второй строчке мы прячем все не первые <p> элементы внутри <div class=«accordion»>.

Когда происходит клик по заголовку <h3>, для следующего в нём элемента <p> будет применен эффект slideToggle, затем для всех остальных элементов <p> будет применен эффект slideUp. Следующие действие изменяет класс заголовка на «active», затем ищем все остальные заголовки <h3> и убираем у них класс «active».

$(document).ready(function () {

$(".accordion h3:first").addClass("active");

$(".accordion p:not(:first)").hide();

$(".accordion h3").click(function () {

$(this).next("p").slideToggle("slow")

.siblings("p:visible").slideUp("slow");

$(this).toggleClass("active");

$(this).siblings("h3").removeClass("active");

});

});

Таким образом, с помощью этой библиотеки можно сделать множество сложных эффектов таким простым способом.

**Backbone**

JavaScript-библиотека, основанная на шаблоне проектирования Model-View-Presenter (MVP), предназначена для разработки веб-приложений с поддержкой RESTful JSON интерфейса. Backbone — очень лёгкая библиотека (упакованная и gzip-сжатая 6.3 Кб), но для работы необходима библиотека Underscore.js, а для поддержки REST API и работы с DOM элементами рекомендуется подключить jQuery-подобную библиотеку: jQuery или Zepto. Backbone.js создан Джереми Ашкенасом, который известен также как создатель CoffeeScript.

Backbone.js придает структуру веб-приложениям с помощью моделей с биндингами по ключу и пользовательскими событиями, коллекций с богатым набором методов с перечислимыми сущностями, представлений с декларативной обработкой событий; и соединяет это все с вашим существующим REST-овым JSON API.

Проект размещается на GitHub, с доступным аннотированным исходным кодом, с онлайновыми тестами с примером приложения, со списком туториалов и списком реальных проектов, которые используют Backbone. Backbone доступен под лицензией MIT.

Работая с Backbone, вы представляете ваши данные как Модели (Models), которые могут быть созданы, провалидированы, удалены, и сохранены на сервере. Всякий раз, когда в интерфейсе изменяется атрибуты модели, модель вызывает событие "change"; все Представления (Views), которые отображают состояние модели, могут быть уведомлены об изменении атрибутов модели, с тем чтобы они могли отреагировать соответствующим образом — например, перерисовать себя с учетом новых данных. В готовом приложении на Backbone, вы не должны писать код, ищущий элемент с определенным id в DOM и обновлять HTML вручную. При изменении модели представление просто обновит себя самостоятельно.

**Angular**

По большому счету AngularJS является основой, которая связывает HTML-код (который генерируется для просмотра страницы в окне браузера) с JavaScript объектов /моделей. Когда изменяется один из объектов, автоматически обновляется и генерируемая страница. Верно и обратное - модели связаны с текстовым полем (контентом страницы). Когда изменяется контент, это вызывает изменения и в коде сайта.

AngularJS связывает коды в единую систему, и вам не нужно больше обновлять HTML вручную или инспектировать элементы, как в случае, если вы используете JQuery.

AngularJS включает в себя большое количество директив, которые позволяют связать HTML-элементы моделей. Они представляют из себя атрибуты, начинающиеся с префикса ng-, их можно добавлять к любому элементу. Самым важным атрибутом, который, если вы хотите использовать Angular, нужно включить во все страницы сайта является ng- приложений: <body ng-app>.

Angular также применяет контроллеры, принцип действия которых схож с ссылками на виды. Из-за двусторонней привязки данных в Angular очень легко создавать и поддерживать контроллеры, что в итоге очищает код и делает его читаемым. Можно в любой момент получить данные из DOM и отослать их обратно. Также привязка данных делает ненужным большое количество кода, связанного с манипуляцией DOM в JS, еще больше очищая код и придавая модульность контроллерам.

**Zend Framework 2**

Zend Framework 2 является фреймворком с открытым исходным кодом для разработки ВЕБ-приложений на PHP 5.3+. Использует только объектно - ориентированный код и все новшества PHP 5.3, такие как: namespaces (Пространства имен) , late static binding (Позднее статическое связывание), lambda functions and closures (Анонимные функции и замыкания).

Каждый компонент Zend Framework 2 является уникальным и разработан с минимальными зависимостями от других компонентов. ZF2 следует независимому принципу создания приложений. Такая слабосвязанная архитектура позволяет разработчикам использовать только те компоненты, которые им необходимы.Так же используется Pyrus и Composer для установки и отслеживания зависимостей как для всего проекта в целом так и для каждого из компонентов.

**Laravel 5**

Это бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC (англ. Model View Controller — модель-представление-контроллер). Laravel выпущен под лицензией MIT. Исходный код проекта размещается на GitHub.

В 2011 году фреймворк следовал, как и многие популярные opensource-проекты, мантре "релиз раньше, релиз чаще". С исторически растущей популярностью фреймворка наступило время для акцента на требованиях больших организаций и критически-важных приложений, которые нуждаются в безопасных исправлениях, без скорых обновлений. Laravel 5.1 включает 3 года безопасных исправлений.

**NodeJS**

Это программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и Mac OS) и даже программировать микроконтроллеры (например, tessel и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

В рамках дипломного проекта были выбраны следующие технологии – в качестве серверного фреймворка, организующего сохранение и вывод данных Laravel, в качестве удобного шаблонизатора Blade, который идет вместе с Laravel, являясь его частью. Так же для некоторых элементов на сайте использован jQuery. Данные инструменты были выбраны по следующим причинам: бесплатность, открытый исходных код, низкий порог вхождения, популярность.

**Описание среды разработки**

**Язык разметки HTML**

Любой сайт состоит из двух частей – серверной и клиентской. На серверной стороне обрабатывается информация, формируется ответ на запрос, хранится другая информация. В любом веб-приложении, как сейчас часто называют различные сайты, серверная сторона является очень важной составляющей и на ней происходят любые действия. Клиентская же сторона видна пользователям сайта, и они взаимодействуют именно с ней. На ней обычно оставляют логику отображения интерфейса, взаимодействия с пользователем и взаимодействия с сервером. Так же на клиенте может происходить подготовка введенных пользователем данных для последующей отправки на сервер. Такая схема построения общения клиентской и серверной стороны очень распространена и пользуется большой популярностью.

Конечно, любой сайт состоит не только из логики работы и передачи данных. Важнейшая составляющая для пользователя – интерфейс. Пользователи взаимодействуют через него с приложением и именно от него много зависит. Интерфейс создается на языке разметки HTML – HyperTextMarkupLanguage– язык гипертекстовой разметки. Он служит для создания структуры страниц, используя специальный синтаксис.

Каждая страница описывается специальным образом. Начинается страница с тега <html>, в котором уже вложены другие. Так же перед этим тегом может идти тег <!DOCTYPE ..>, в котором указывается спецификация, используемая на этой странице. Этот тег требуется браузерам, чтобы они могли правильно отобразить страницу, согласно логике, описанной в спецификации. Если данный тег не указан, браузер использует настройки по умолчанию, а это может привести к непланируемому поведению страницы.

Тег <html> является контейнером, который заключает в себе все содержимое веб-страницы, включая теги <head> и <body>. Открывающий и закрывающий теги <html> в документе необязательны, но хороший стиль диктует непременное их использование. Как правило, тег <html> идет в документе вторым, после определения типа документа (Document Type Definition, DTD), устанавливаемого через элемент <!DOCTYPE>. Закрывающий тег <html> должен всегда стоять в документе последним.

Атрибуты

* Title - добавляет всплывающую подсказку на веб-страницу.
* Manifest - указывает файл манифеста, необходимый для создания оффлайнового приложения.
* Xmlns - уазывает пространство имен для XHTML-документов.

Элемент <body> предназначен для хранения содержания веб-страницы (контента), отображаемого в окне браузера. Информацию, которую следует выводить в документе, следует располагать именно внутри контейнера <body>. К такой информации относится текст, изображения, теги, скрипты JavaScript и т.д.

Тег <body> также применяется для определения цветов ссылок и текста на веб-странице. Подобная практика в HTML 4 осуждается и взамен для указания цветовой схемы рекомендуется использовать стили, применяя их к селектору BODY. Тем не менее, большинство атрибутов до сих пор поддерживается разными браузерами.

Часто тег <body> используется для размещения обработчика событий, например, onload, которое выполняется после того, как документ завершил загрузку в текущее окно или фрейм.

Открывающий и закрывающий теги <body> на веб-странице не являются обязательными, однако хорошим стилем считается их использование, чтобы определить начало и конец HTML-документа.

Атрибуты

* Alink - устанавливает цвет активной ссылки.
* Background - задает фоновый рисунок на веб-странице.
* Bgcolor - цвет фона веб-страницы.
* Bgproperties - определяет, прокручивать фон совместно с текстом или нет.
* Bottommargin - отступ от нижнего края окна браузера до контента.
* Leftmargin - отступ по горизонтали от левого края окна браузера до контента.
* Link - цвет ссылок на веб-странице.
* Rightmargin - отступ от правого края окна браузера до контента.
* Scroll - устанавливает, отображать полосы прокрутки или нет.
* Text - цвет текста в документе.
* Topmargin - отступ от верхнего края окна браузера до контента.
* Vlink - цвет посещенных ссылок.

Элемент <div> является блочным элементом и предназначен для выделения фрагмента документа с целью изменения вида содержимого. Как правило, вид блока управляется с помощью стилей. Чтобы не описывать каждый раз стиль внутри тега, можно выделить стиль во внешнюю таблицу стилей, а для тега добавить атрибут class или id с именем селектора.

Как и при использовании других блочных элементов, содержимое тега <div> всегда начинается с новой строки. После него также добавляется перенос строки.

Атрибуты

* Align - задает выравнивание содержимого тега <div>.
* Title - добавляет всплывающую подсказку к содержимому.

Тег <span> предназначен для определения строчных элементов документа. В отличие от блочных элементов, таких как <table>, <p> или <div>, с помощью тега <span> можно выделить часть информации внутри других тегов и установить для нее свой стиль. Например, внутри абзаца (тега <p>) можно изменить цвет и размер первой буквы, если добавить начальный и конечный тег <span> и определить для него стиль текста. Чтобы не описывать каждый раз стиль внутри тега, можно выделить стиль во внешнюю таблицу стилей, а для тега добавить атрибут class или id с именем селектора.

Определяет заголовок документа. Элемент <title> не является частью документа и не показывается напрямую на веб-странице. В операционной системе Windows текст заголовка отображается в левом верхнем углу окна браузера. Допускается использовать только один тег <title> на документ и размещать его в контейнере <head>.

Тег <title> выполняет множество задач, как прямо, так и косвенно. Ниже описано какую роль выполняет заголовок страницы.

По тексту заголовка пользователь получает дополнительную информацию, что это за сайт, на котором он находится и как называется текущая страница. Не стоит думать, что достаточно в документе указать логотип сайта и проигнорировать заголовок, ведь посетитель может свернуть окно. В свернутом виде заголовок также отображается на кнопках панели задач, поэтому можно легко ориентироваться, с каким сайтом в данный момент работать, а не перелистывать их по очереди.

Большинство браузеров поддерживают возможность сохранения веб-страницы на локальный компьютер. В этом случае имя сохраненного файла совпадает с названием заголовка документа. Если в тексте заголовка содержатся символы недопустимые в имени файла (\ / : \* ? " < > |), они будут проигнорированы или автоматически заменены другими дозволенными символами.

При сохранении в разделе браузера «Избранное», адрес текущей страницы с ее заголовком помещается в список предпочитаемых ссылок. Поскольку этот список, как правило, хранится в виде отдельных файлов, к их именам также прилагается уже вышеописанное правило.

В результатах поиска по ключевым словам, поисковые системы используют заголовок страницы для указания ссылки на данный документ. Интересно написанный заголовок, содержащий ключевые слова, привлечет больше внимания посетителей и увеличит шансы на то, что сайт посетит больше людей.

Тег <input> является одним из разносторонних элементов формы и позволяет создавать разные элементы интерфейса и обеспечить взаимодействие с пользователем. Главным образом <input> предназначен для создания текстовых полей, различных кнопок, переключателей и флажков. Хотя элемент <input> не требуется помещать внутрь контейнера <form>, определяющего форму, но если введенные пользователем данные должны быть отправлены на сервер, где их обрабатывает серверная программа, то указывать <form> обязательно. То же самое обстоит и в случае обработки данных с помощью клиентских приложений, например, скриптов на языке JavaScript.

Основной атрибут тега <input>, определяющий вид элемента — type. Он позволяет задавать следующие элементы формы: текстовое поле (text), поле с паролем (password), переключатель (radio), флажок (checkbox), скрытое поле (hidden), кнопка (button), кнопка для отправки формы (submit), кнопка для очистки формы (reset), поле для отправки файла (file) и кнопка с изображением (image). Для каждого элемента существует свой список атрибутов, которые определяют его вид и характеристики. Кроме того, в HTML5 добавлено еще более десятка новых элементов.

Атрибуты

* Accept - Устанавливает фильтр на типы файлов, которые вы можете отправить через поле загрузки файлов.
* Accesskey - Переход к элементу с помощью комбинации клавиш.
* Align - Определяет выравнивание изображения.
* Alt - Альтернативный текст для кнопки с изображением.
* Autocomplete - Включает или отключает автозаполнение.
* Autofocus - Устанавливает фокус в поле формы.
* Border - Толщина рамки вокруг изображения.
* Checked - Предварительно активированный переключатель или флажок.
* Disabled - Блокирует доступ и изменение элемента.
* Form - Связывает поле с формой по её идентификатору.
* Formaction - Определяет адрес обработчика формы.
* Formenctype - Устанавливает способ кодирования данных формы при их отправке на сервер.
* Formmethod - Сообщает браузеру каким методом следует передавать данные формы на сервер.
* Formnovalidate - Отменяет встроенную проверку данных на корректность.
* Formtarget - Определяет окно или фрейм в которое будет загружаться результат, возвращаемый обработчиком формы.
* List - Указывает на список вариантов, которые можно выбирать при вводе текста.
* Max - Верхнее значение для ввода числа или даты.
* Maxlength - Максимальное количество символов разрешенных в тексте.
* Min - Нижнее значение для ввода числа или даты.
* Multiple - Позволяет загрузить несколько файлов одновременно.
* Name - Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
* Pattern - Устанавливает шаблон ввода.
* Placeholder - Выводит подсказывающий текст.
* Readonly - Устанавливает, что поле не может изменяться пользователем.
* Required - Обязательное для заполнения поле.
* Size - Ширина текстового поля.
* Src - Адрес графического файла для поля с изображением.
* Step - Шаг приращения для числовых полей.
* Tabindex - Определяет порядок перехода между элементами с помощью клавиши Tab.
* Type - Сообщает браузеру, к какому типу относится элемент формы.
* Value - значение элемента.

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

Рисунки также могут применяться в качестве карт-изображений, когда картинка содержит активные области, выступающие в качестве ссылок. Такая карта по внешнему виду ничем не отличается от обычного изображения, но при этом оно может быть разбито на невидимые зоны разной формы, где каждая из областей служит ссылкой.

Атрибуты

* Align - Определяет как рисунок будет выравниваться по краю и способ обтекания текстом.
* Alt - Альтернативный текст для изображения.
* Border - Толщина рамки вокруг изображения.
* Height - Высота изображения.
* Hspace - Горизонтальный отступ от изображения до окружающего контента.
* Ismap - Говорит браузеру, что картинка является серверной картой-изображением.
* Longdesc - Указывает адрес документа, где содержится аннотация к картинке.
* Lowsrc - Адрес изображения низкого качества.
* Src - Путь к графическому файлу.
* Vspace - Вертикальный отступ от изображения до окружающего контента.
* Width - Ширина изображения.
* Usemap - Ссылка на тег <map>, содержащий координаты для клиентской карты-изображения.

Тег <select> позволяет создать элемент интерфейса в виде раскрывающегося списка, а также список с одним или множественным выбором, как показано далее. Конечный вид зависит от использования атрибута size тега <select>, который устанавливает высоту списка. Ширина списка определяется самым широким текстом, указанным в теге <option>, а также может изменяться с помощью стилей. Каждый пункт создается с помощью тега <option>, который должен быть вложен в контейнер <select>. Если планируется отправлять данные списка на сервер, то требуется поместить элемент <select> внутрь формы. Это также необходимо, когда к данным списка идет обращение через скрипты.

Атрибуты

* Accesskey - Позволяет перейти к списку с помощью некоторого сочетания клавиш.
* Autofocus - Устанавливает, что список получает фокус после загрузки страницы.
* Disabled - Блокирует доступ и изменение элемента.
* Form - Связывает список с формой.
* Multiple - Позволяет одновременно выбирать сразу несколько элементов списка.
* Name - Имя элемента для отправки на сервер или обращения через скрипты.
* Required - Список обязателен для выбора перед отправкой формы.
* Size - Количество отображаемых строк списка.
* Tabindex - Определяет последовательность перехода между элементами при нажатии на клавишу Tab.

Тег <button> создает на веб-странице кнопки и по своему действию напоминает результат, получаемый с помощью тега <input> (с атрибутом type="button | reset | submit"). В отличие от этого тега, <button> предлагает расширенные возможности по созданию кнопок. Например, на подобной кнопке можно размещать любые элементы HTML, в том числе изображения. Используя стили можно определить вид кнопки путем изменения шрифта, цвета фона, размеров и других параметров.

Теоретически, тег <button> должен располагаться внутри формы, устанавливаемой элементом <form>. Тем не менее, браузеры не выводят сообщение об ошибке и корректно работают с тегом <button>, если он встречается самостоятельно. Однако, если необходимо результат нажатия на кнопку отправить на сервер, помещать <button> в контейнер <form> обязательно.

Атрибуты

* Accesskey - Доступ к элементам формы с помощью горячих клавиш.
* Autofocus - Устанавливает, что кнопка получает фокус после загрузки страницы.
* Disabled - Блокирует доступ и изменение элемента.
* Form - Связывает между собой форму и кнопку.
* Formaction - Задаёт адрес, на который пересылаются данные формы при нажатии на кнопку.
* Formenctype - Способ кодирования данных формы.
* Formmethod - Указывает метод пересылки данных формы.
* Formnovalidate - Отменяет проверку формы на корректность.
* Formtarget - Открывает результат отправки формы в новом окне или фрейме.
* Name - Определяет уникальное имя кнопки.
* Type - Тип кнопки — обычная, для отправки данных формы на сервер или для очистки формы.
* Value - Значение кнопки, которое будет отправлено на сервер или прочитано с помощью скриптов.

Тег <a> является одним из важных элементов HTML и предназначен для создания ссылок. В зависимости от присутствия атрибутов name или href тег <a> устанавливает ссылку или якорь. Якорем называется закладка внутри страницы, которую можно указать в качестве цели ссылки. При использовании ссылки, которая указывает на якорь, происходит переход к закладке внутри веб-страницы.

Для создания ссылки необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку. В качестве значения атрибута href используется адрес документа (URL, Universal Resource Locator, универсальный указатель ресурсов), на который происходит переход. Адрес ссылки может быть абсолютным и относительным. Абсолютные адреса работают везде и всюду независимо от имени сайта или веб-страницы, где прописана ссылка. Относительные ссылки, как следует из их названия, построены относительно текущего документа или корня сайта.

Атрибуты

* Accesskey - Активация ссылки с помощью комбинации клавиш.
* Cords - Устанавливает координаты активной области.
* Download - Предлагает скачать указанный по ссылке файл.
* Href - Задает адрес документа, на который следует перейти.
* Hreflang - Идентифицирует язык текста по ссылке.
* Name - Устанавливает имя якоря внутри документа.
* Rel - Отношения между ссылаемым и текущим документами.
* Rev - Отношения между текущим и ссылаемым документами.
* Shape - Задает форму активной области ссылки для изображений.
* Tabindex - Определяет последовательность перехода между ссылками при нажатии на кнопку Tab.
* Target - Имя окна или фрейма, куда браузер будет загружать документ.
* Title - Добавляет всплывающую подсказку к тексту ссылки.
* Type - Указывает MIME-тип документа, на который ведёт ссылка.

Элемент <table> служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью тегов <tr> и <td>. Внутри <table> допустимо использовать следующие элементы: <caption>, <col>, <colgroup>, <tbody>, <td>, <tfoot>, <th>, <thead> и <tr>.

Таблицы с невидимой границей долгое время использовались для верстки веб-страниц, позволяя разделять документ на модульные блоки. Подобный способ применения таблиц нашел воплощение на многих сайтах, пока ему на смену не пришел более современный способ верстки с помощью слоев.

Атрибуты

* Align - Определяет выравнивание таблицы.
* Background - Задает фоновый рисунок в таблице.
* Bgcolor - Цвет фона таблицы.
* Border - Толщина рамки в пикселах.
* Bordercolor - Цвет рамки.
* Cellpadding - Отступ от рамки до содержимого ячейки.
* Cellspacing - Расстояние между ячейками.
* Cols - Число колонок в таблице.
* Frame - Сообщает браузеру, как отображать границы вокруг таблицы.
* Height - Высота таблицы.
* Rules - Сообщает браузеру, где отображать границы между ячейками.
* Summary - Краткое описание таблицы.
* Width - Ширина таблицы.

Элемент <thead> предназначен для хранения одной или нескольких строк, которые представлены вверху таблицы. Допустимо использовать не более одного элемента <thead> в пределах одной таблицы, и он должен идти в исходном коде сразу после тега <table>.

Атрибуты

* Align - Определяет выравнивание текста в строке.
* Char - Выравнивание содержимого ячеек относительно заданного символа.
* Charoff - Смещение содержимого ячеек относительно указанного символа.
* Bgcolor - Цвет фона ячеек, которые расположены внутри контейнера <thead>.
* Valign - Выравнивание содержимого ячеек по вертикали.

Тег <tr> служит контейнером для создания строки таблицы. Каждая ячейка в пределах такой строки может задаваться с помощью тега <th> или <td>.

Атрибуты

* Align - Определяет выравнивание содержимого ячеек по горизонтали.
* Bgcolor - Цвет фона ячеек.
* Bordercolor - Цвет рамки.
* Char - Выравнивание содержимого ячеек относительно заданного символа.
* Charoff - Смещение содержимого ячеек относительно указанного символа.
* Valign - Выравнивание содержимого ячеек по вертикали.

Элемент <tbody> предназначен для хранения одной или нескольких строк таблицы. Это позволяет создавать структурные блоки, к которым можно применять единое оформление через стили, а также управлять их видом через скрипты.

Допускается применять несколько тегов <tbody> внутри контейнера <table>. Доступны и другие виды группировок строк — <tfoot> и <thead>, ни один из них не должен перекрываться с элементом <tbody>.

Атрибуты

* Align - Определяет выравнивание текста в строке.
* Char - Выравнивание содержимого ячеек относительно заданного символа.
* Charoff - Смещение содержимого ячеек относительно указанного символа.
* Bgcolor - Цвет фона ячеек, которые расположены внутри контейнера <tbody>.
* Valign - Выравнивание содержимого ячеек по вертикали.

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

Атрибуты

* Abbr - Краткое описание содержимого ячейки.
* Align - Определяет выравнивание содержимого ячейки по горизонтали.
* Axis - Группирует ячейки связанные между собой похожей информацией.
* Background - Задает фоновый рисунок в ячейке.
* Bgcolor - Цвет фона ячейки.
* Bordercolor - Цвет рамки.
* Char - Выравнивает содержимое ячейки по заданному символу.
* Charoff - Смещает содержимое ячейки относительно заданного символа.
* Colspan - Объединяет горизонтальные ячейки.
* Headers - Позволяет связать ячейки с заголовком.
* Height - Высота ячейки.
* Nowrap - Запрещает перенос строк.
* Rowspan - Объединяет вертикальные ячейки.
* Scope - Позволяет связать ячейки простой таблицы с заголовками в речевых браузерах.
* Valign - Выравнивание содержимого ячейки по вертикали.
* Width - Ширина ячейки.

Тег <th> предназначен для создания одной ячейки таблицы, которая обозначается как заголовочная. Текст в такой ячейке отображается браузером обычно жирным шрифтом и выравнивается по центру. Тег <th> должен размещаться внутри контейнера <tr>, который в свою очередь располагается внутри тега <table>.

Атрибуты

* Abbr - Краткое описание содержимого ячейки.
* Align - Определяет выравнивание содержимого ячейки по горизонтали.
* Axis - Группирует ячейки связанные между собой похожей информацией.
* Background - Задает фоновый рисунок в ячейке.
* Bgcolor - Цвет фона ячейки.
* Bordercolor - Цвет рамки.
* Char - Выравнивает содержимое ячейки по заданному символу.
* Charoff - Смещает содержимое ячейки относительно заданного символа.
* Colspan - Объединяет горизонтальные ячейки.
* Headers - Позволяет связать ячейки таблицы с заголовками в речевых браузерах.
* Height - Высота ячейки.
* Nowrap - Запрещает перенос строк.
* Rowspan - Объединяет вертикальные ячейки.
* Scope - Позволяет связать ячейки простой таблицы с заголовками в речевых браузерах.
* Valign - Выравнивание содержимого ячейки по вертикали.
* Width - Ширина ячейки.

**Стили CSS**

Каким же образом можно сделать пустую и неказистую страницу приятой для глаза и взаимодействия? Для этого была придумана каскадная таблица стилей. Она очень удобна тем, что разделяет описание структуры страницы и ее дизайна в разные файлы, два или больше. Стили удобны тем, что один и тот же стиль можно применить к нескольким элементам, как это делают с классами. Например, можно указать несколько элементов на странице с одним классом: <divclass=”class1”><divclass=”class1”></div></div><divclass=”class1”></div>. После описания в файле стилей класса .class1 {}, он применится ко всем элементам. Это очень удобно, особенно с часто повторяющимся кодом дизайна, когда одинаковых элементов на странице достаточно много и они разбросаны по всей структуре.

Так же стили позволяют описывать не только конкретный класс или идентификатор, но и вложенные элементы. Например, если необходимо описать стиль всех элементов <li>в теге <ul>с классом .list, то в стилях это будет выглядеть следующим образом: ul.list>li {}. Данное описание очевидно, не требует дополнительного времени для разбора вложенности. Трудности с такими вещами могут возникать только у новичков, но им редко приходится сталкиваться с таким. Если правильно уметь использовать селекторы, а именно так это называется, то разработка укоряется и становится легче. Так же у этого способа есть и обратная сторона – можно запутаться, если сильно увлечься этой вложенностью, а не использовать описания новых классов, которые будут добавлены в структуре.

Совместное использование HTMLи CSS, особенно их последних версий, может сильно упростить разработку, а так же поможет создать действительно красивый сайт, на котором будет приятно находиться. Правильное описание интерфейса сайта уменьшает проблемы как с отображением в различных браузерах и на разных устройствах, так и облегчает использование.

Вот некоторые примеры различных CSS-свойств, часто используемых в разных проектах:

background-color - Определяет цвет фона элемента. Хотя это свойство не наследует свойства своего родителя, из-за того, что начальное значение устанавливается прозрачным, цвет фона дочерних элементов совпадает с цветом фона родительского элемента.

background-image - Устанавливает фоновое изображение для элемента. Если одновременно для элемента задан цвет фона, он будет показан, пока фоновая картинка не загрузится полностью. То же произойдет, если изображения не доступны или их показ в браузере отключен. В случае наличия в рисунке прозрачных областей, через них будет проглядывать фоновый цвет. В CSS3 допустимо указывать несколько фоновых изображений, перечисляя их параметры через запятую.

Border - Универсальное свойство border позволяет одновременно установить толщину, стиль и цвет границы вокруг элемента. Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному свойству. Для установки границы только на определенных сторонах элемента, воспользуйтесь свойствами border-top, border-bottom, border-left, border-right.

Padding - Устанавливает значение полей вокруг содержимого элемента. Полем называется расстояние от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое.

Margin - Устанавливает величину отступа от каждого края элемента. Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента.

Если у элемента нет родителя, отступом будет расстояние от края элемента до края окна браузера с учетом того, что у самого окна по умолчанию тоже установлены отступы. Чтобы от них избавиться, следует устанавливать значение margin для селектора <body> равное нулю.

Свойство margin позволяет задать величину отступа сразу для всех сторон элемента или определить ее только для указанных сторон.

Overflow - Свойство overflow управляет отображением содержания блочного элемента, если оно целиком не помещается и выходит за область заданных размеров.

Width - Устанавливает ширину блочных или заменяемых элементов (к ним, например, относится тег <img>). Ширина не включает толщину границ вокруг элемента, значение отступов и полей.

**Шаблонизатор Blade**

Blade - простой, но мощный шаблонизатор, входящий в состав Laravel. Blade основан на концепции наследования шаблонов и секциях. Все шаблоны Blade должны иметь расширение .blade.php.

**Создание шаблона Blade:**

<html>

<head>

<title>App Name - @yield('title')</title>

</head>

<body>

@section('sidebar')

Это - главный сайдбар.

@show

<div class="container">

@yield('content')

</div>

</body>

</html>

**Использование шаблона Blade:**

@extends('layouts.master')

@section('title', 'Заголовок страницы')

@section('sidebar')

@

<p>Этот элемент будет добавлен к главному сайдбару.</p>

@stop

@section('content')

<p>Это - содержимое страницы.</p>

@stop

Заметьте, что шаблоны, которые расширяют другой Blade-шаблон с помощью extend, просто перекрывают секции последнего. Старое (перекрытое) содержимое может быть выведено директивой @.

Иногда - например, когда вы не уверены, что секция была определена - вам может понадобиться указать значение по умолчанию для директивы @yield. Вы можете передать его вторым аргументом:

@yield('section', 'Default Content')

**Вывод переменной:**

Привет, {{ $name }}.

Текущее время эпохи UNIX: {{ time() }}.

**Вывод переменной с проверкой на существование:**

Иногда вам нужно вывести переменную, которая может быть определена, а может быть нет. Например:

{{ isset($name) ? $name : 'Default' }}

Но вам не обязательно писать тернарный оператор, Blade позволяет записать это короче:

{{ $name or 'Default' }}

**Вывод текста с фигурными скобками:**

Если вам нужно вывести текст с фигурными скобками - управляющими символами Blade - вы можете поставить префикс @:

@{{ Этот текст не будет обрабатываться шаблонизатором Blade }}

По умолчанию весь выводимый контент экранируется, т.е. все элементы и сущности HTML показываются как есть, вместо того, чтобы обрабатываться браузером. Если вы не хотите, что бы данные экранировались, используйте следующий синтаксис:

Hello, {!! $name !!}.

**Директива If:**

@if (count($records) === 1)

Здесь есть одна запись!

@elseif (count($records) > 1)

Здесь есть много записей!

@else

Здесь нет записей!

@endif

@unless (Auth::check())

Вы не вошли в систему.

@endunless

**Циклы:**

@for ($i = 0; $i < 10; $i++)

Текущее значение: {{ $i }}

@endfor

@foreach ($users as $user)

<p>Это пользователь{{ $user->id }}</p>

@endforeach

@forelse($users as $user)

<li>{{ $user->name }}</li>

@empty

<p>No users</p>

@endforelse

@while (true)

<p>Это будет длиться вечно.</p>

@endwhile

**Подшаблоны:**

@include('view.name')

Также можно передать массив переменных во включаемый шаблон:

@include('view.name', ['some'=>'data'])

**Язык JS**

**Язык PHP**

**Laravel**

**Редактор SublimeText(2 и 3 версии)**

**Система контроля версий Git**