Оглавление

[Введение](#_Toc73539966) 3

[Глава 1 Теоретическая часть](#_Toc73539967) 4

[1.1 Понятия эргономичный интерфейс](#_Toc73539972) 2

1.2 Цветовая политра в web....................................................................................5

1.3 История изменения дизайна сайта и шаблоны...............................................9

1.4 История и применение «MySQL»..................................................................12

1.5 История и применение «PHP»........................................................................16

1.6. История и применение HTML ……………………………………………..18

1.7 История и применение CSS………………………………………………....21

1.8 История и применение JavaScript…………………………………………..24

1.9. История и применение фреймворка «Bootstrap» …………………………30

2. История и применение фреймворка «Jquery»…………………………….....37

2.1 История и применение AJAX.........................................................................40

2.2 История и применение фреймворка «Jquery-UI»..………………………...46

2.3 Фрейморк «Owl carousel»...............................................................................47

2.4 Шаблон проектирования MVC......................................................................49

2.5 История и применение фреймворка «Codeigniter»..……………………….51

2.6 Выбор среды разработки................................................................................54

2.6.1 MAMP..................................................................................................54

2.6.2 Vusial studio code................................................................................55

2.8 Создание собственного сайта......................................................................56

[Глава 2 Практическая часть..............................................................................58](#_Toc73539980)

1.1 Главная страница сайта...............................................................................58

Приложения...........................................................................................................59

Источники..........................................................................................................60

**1.1 Понятия эргономичный интерфейс**

Эргономика в дизайне пользовательского интерфейса основана на оптимальном и гармоничном расположении его элементов. В конечном итоге она обеспечивает отличные показатели юзабилити и пользовательской активности. Более точное представление об эргономике в дизайне даст изучение ее основных принципов.

Эргономика в дизайне пользовательского интерфейса базируется на оптимальном и гармоничном расположении его элементов. В результате достигаются высокие показатели юзабилити и пользовательской активности. Изучение основных принципов эргономики позволит глубже понять эту концепцию.

#1. Человеко-ориентированный дизайн.

При разработке интерфейса важно учитывать индивидуальные особенности пользователя, такие как физические и когнитивные способности, а также потребности и цели. Такой подход помогает решать конкретные проблемы, связанные с взаимодействием человека с программными интерфейсами, и достигать целей, актуальных для определённых групп пользователей.

Благодаря такому подходу пользователи быстрее осваивают продукт и адаптируются к работе с ним, получая при этом интуитивно понятное и удобное решение.

#2. Аффордансы

Аффордансы — это нативные визуальные подсказки, такие как кнопки, иконки и ярлыки, которые помогают пользователям понять, как использовать функционал веб-решения. Эти элементы быстро и легко передают пользователям идею о том, что нужно сделать для выполнения целевого действия без дополнительных всплывающих подсказок и изображений. Таким образом, пользователи не тратят лишнее время на достижение желаемого результата в рамках программного продукта, что положительно влияет на конверсию.

#3. Обратная связь

Интерфейс должен предоставлять пользователям чёткую и мгновенную информацию об их действиях, например, об ошибках и подтверждении операций. Благодаря этому пользователи будут понимать, правильно ли они выполняют целевые действия, и улучшат общее качество пользовательского опыта.

#4. Согласованность и предсказуемость

Интерфейс должен быть совместим с различными операционными системами и устройствами, иметь похожее оформление и вести себя одинаково. Это означает, что при смене устройства или операционной системы пользователям не придётся заново изучать работу с программным продуктом, и они смогут взаимодействовать с ним так же, как и раньше.

То же самое относится к предсказуемости: типичные элементы интерфейса должны выполнять функции, к которым привыкли пользователи. Например, если это кнопка, она должна подтверждать определённое действие. Если это всплывающее окно, оно должно уведомлять пользователя о событии и последующих действиях и т. д.

#5. Гибкость и персонализация

Гибкость заключается в следовании не прямому, а знакомому и интуитивно понятному пользователю пути. В частности, разработчик должен уметь распознавать цели своей целевой аудитории и помогать ей двигаться по веб-проекту с учётом этих целей.

Что касается настройки, интерфейс должен позволять пользователям настраивать его в соответствии со своими предпочтениями. Обычно такая адаптация означает «подгонку» отдельных функций интерфейса под каждого конкретного пользователя в соответствии с его требованиями. По сути, это позволяет разработчику веб-решений предложить представителям целевой аудитории индивидуальный подход в обслуживании и тем самым улучшить их личный опыт.

#6. Минимализм

Интерфейс должен быть простым и понятным, чтобы пользователи не терялись среди множества элементов. Необходимо тщательно проанализировать все функции на одной странице и удалить те, которые не влияют критически на выполнение основных задач.

#7. Доступность

Идеальный интерфейс программного продукта должен быть доступен для людей с ограниченными возможностями. Соответствие стандартам WCAG 2 можно проверить или создать собственные требования, которые сделают интерфейс доступным для определённой аудитории.

#8. Юзабилити

Пользовательский интерфейс должен быть простым для навигации и понятным, обеспечивая точное соответствие инструкций и обратной связи действиям пользователей. Задача разработчика — создать дизайн, в котором новые пользователи могут ориентироваться без подсказок или с минимальным количеством помощи.

#9. Изучаемость

Изучаемость интерфейса означает лёгкость его освоения. Это позволяет новым пользователям не тратить много времени на обучение и сокращает время на повторение действий для полного понимания работы функций.

#10. Эстетическая привлекательность

Ещё один важный аспект, о котором должен помнить каждый разработчик при создании эргономичного дизайна, — визуальная привлекательность интерфейса. Удобство не привлечёт и не удержит целевую аудиторию без хорошего дизайна. Это включает в себя тщательный выбор цветовой гаммы, продуманное размещение элементов и свободное пространство, использование динамических деталей и т. д.

#11. Эффективность

Интерфейс должен позволять пользователям выполнять задачи с минимальными затратами времени и удобства. Обратите внимание, что эффективность не может быть определена как бинарное противопоставление (эффективный или неэффективный), её следует оценивать как «достаточно» или «недостаточно».

**Источник** [**https://nuancesprog.ru/p/17293/**](https://nuancesprog.ru/p/17293/)

**1.2 «Цветовая палитра в web»**

Цвета окружают нас повсюду, проявляясь в разнообразных оттенках, градиентах и сочетаниях. Рассматривая окружающий мир, мы поражаемся богатству цветов и оттенков, которые он содержит. Некоторые цвета вызывают у нас приятные ощущения, однако существуют и уникальные цветовые комбинации, оказывающие значительное влияние на наши эмоции. Для глубокого понимания всех этих аспектов необходимо изучить теорию цвета.

Для дизайнеров эта теория служит основным принципом, определяющим восприятие и использование цветов в разных областях, таких как живопись, дизайн и иллюстрация. Для успешного применения цветов важно разбираться в понятиях комплементарных, сплит-комплементарных и триадических цветов, которые основаны на теории цвета. Дизайнеры часто опираются на цветовой круг и теорию цвета при выборе подходящих цветовых комбинаций, способных передать настроение и эмоции бренда или человека.

Эта теория поможет вам ответить на важные вопросы, например, какие цвета смешивать для получения нужного оттенка или какой уровень насыщенности и яркости будет наиболее подходящим.

Немного истории

История цвета уходит корнями в древние времена, когда пигменты, такие как цветная почва, уголь и животный жир, использовались для создания произведений искусства. Древние цивилизации, например египтяне, использовали эти пигменты для окрашивания предметов, и синий цвет был наиболее распространённым.

С течением времени художники продолжали вносить свой вклад в развитие цвета. Леонардо да Винчи изучал особенности и принципы создания цветов, описывая их в своих рукописях.

Развитие науки о цвете началось в XVIII веке с открытия Исаака Ньютона о том, что свет состоит из нескольких цветов. Эксперименты показали, что красный, жёлтый и синий являются основными цветами, из которых можно получить все остальные. Ньютон создал первую версию цветового круга, которая оказала значительное влияние на дальнейшее развитие принципов работы с цветом.

Альфред Манселл дополнил цветовой круг третичными цветами, такими как жёлто-оранжевый, красно-оранжевый, красно-фиолетовый, сине-фиолетовый и жёлто-зелёный. Он также определил, что двухмерный цветовой круг подходит для описания только одного измерения цвета, и впервые представил цвет в трёхмерном цветовом пространстве через три его атрибута: тон, значение (яркость) и хроматичность.

Сегодня дизайнеры используют цветовой круг для создания привлекательных работ с помощью палитр, основанных на теории цвета.

Цвет и его теория

Чтобы лучше понять цвет и его теорию, мы для начала вспомним, какие в цветовом круге существуют категории.

Основные цвета выступают основой для получения всех остальных цветов. К ним относятся красный, синий и желтый. (примечание: данное определение характерно для художественной системы цветовоспроизведения, которая расходится с аддитивной системой RGB Максвелла, где основными цветами являются красный, зеленый и синий).

Вторичные цвета получаются совмещением двух основных. К ним относятся оранжевый, фиолетовый и зеленый.

Третичные цвета создаются смешиванием основных со вторичными. Всего их насчитывается шесть: красно-оранжевый, красно-фиолетовый, сине-фиолетовый, сине-зеленый, желто-зеленый и желто-оранжевый.

Категории цветов

Цветовой круг представляет основные, вторичные, третичные цвета, а также их оттенки, тени и тональности. Он выступает в качестве ориентира, позволяя дизайнерам создавать уникальные и привлекательные цветовые палитры.

Тон указывает на цветовой пигмент и может полноправно называться соответствующим цветом.

Хроматичность описывает чистоту цвета. Высокая хроматичность тона указывает на то, что в нем отсутствует черный, серый и белый. С другой стороны, присутствие черного или серого просто снижает хроматичность. Дизайнер должен понимать, что следует избегать использования тонов со схожей хроматичностью и предпочитать брать те, чья хроматичность отличается на несколько оттенков.

Насыщенность определяет, как выглядит тон при различный условиях освещения. Для достижения цветового соответствия рекомендуется использовать степени насыщенности, максимально приближенные к природным.

Тональности представляют цвета, получающиеся после добавлении в тон серого. Смотрятся они тускло и скучно, как старые программы, в связи с чем подходят далеко не для всех дизайнерских решений. Тем не менее, если в проект нужно добавить винтажный эффект, с их помощью можно сделать это более элегантно.

Тени используются для затемнения цвета путем добавления в его тон черного.

Значение (яркость) цвета отражает его светлоту или темноту.

Оттенок противоположен тени и отражает, сколько белого было добавлено к цвету. Оттенки зачастую применяются в области женского дизайна, и их в изобилии можно встретить на сайтах, посвященных родительству или воспитанию юных детишек.

Цветовые палитры

Смешивание цветов в поиске нужных комбинаций является интересным и увлекательным занятием, особенно для дизайнеров. С помощью цветовой палитры они добиваются нужных красочных решений при проектировании UI или логотипов. В умелых руках таким образом создаются визуально привлекательные фирменные стили, которые успешно резонируют с эмоциями целевой аудитории.

Дизайнер оперирует богатым арсеналом цветовых схем, при использовании которых очень важно добиваться гармонии. В этом как раз и помогают цветовые палитры, позволяя получать согласованный и эстетически привлекательный дизайн. Придуманы они были не одно тысячелетие назад, но в современной цифровой эпохе получили еще большую популярность. Цветовые палитры представляются в форме HEX-кодов, с помощью которых мы сообщаем компьютеру о том, какой цвет нам нужен.

Психология цвета изучает воздействие различных оттенков на эмоциональное состояние человека. Яркие и насыщенные цвета обычно воспринимаются как более привлекательные и желанные, в то время как тёмные оттенки могут вызывать негативные эмоции.

Тепловое восприятие цветов также играет важную роль. Красные, оранжевые и жёлтые оттенки ассоциируются с теплом, синие — с холодом и водой, а зелёные — с природой. Это обусловлено особенностями восприятия цветов глазами и передачей сигналов в мозг.

Многие компании и корпорации используют знания о психологии цвета для создания брендов и дизайна продуктов. Удачно подобранные цвета помогают сформировать определённые ассоциации и сделать продукты более привлекательными и запоминающимися.

При создании цветовой палитры важно добиться уникальности бренда. Здесь дизайнеру нужно учесть ряд факторов, например фирменный дизайн конкурентов и их цветовые схемы. Использование схожей цветовой палитры может вызвать путаницу среди потребителей. Однако при этом не обязательно брать абсолютно иную цветовую гамму, лучше будет просто обыграть аналогичную в собственном ключе. Например, сети McDonald’s и Burger King обе работают в сфере быстрого питания и предлагают аналогичный сервис. При этом и тот, и другой бренд используют в фирменном стиле комбинацию красного и желтого.

**ТУТ написать кратко, плюсы выбора цвета типо: выбор такой палитры успокаивает и не отталкивает пользователю приятно находится и тд**

**1.3 История изменения дизайна сайта и шаблоны**

Пожалуй, лучший способ понять важность понятия « эргономичный интерфейс» –это на дизайн сайтов прошлого. Таким образом можно проследить и выцепить основные изменения и тренды в верстке.

Я нашел специальный сайт «Web Design Museum» , где собранны скриншоты множества популярных сайтов из разных временных периодов. Просмотрев страницы, которые были в начале 90-ых, а конкретнее в 1991 году, я увидел следующее: Черный текст на белом фоне. Выделялись только ссылки на другие страницы. Это абсолютно базовая вёрстка HTML. На страницах были заголовки 1 и 2 уровня тега «H», обертка текста в «<p>» тег и вкрапления ссылок через «<a>» тег.

Сайты за 1992 год, некоторые сайты уже начали использовать изображения в своих страницах. На сайтах музыкальных артистов или высших учебных заведений были использованы фото артистов или фасадов здания и прилегающей территории. На странице была базовые «h1» и «h2» теги, обертка текста в «p» тег, ссылки на другие страницы в «а» теге,но добавился «img» тег с gif изображениями.  
Сайты за 1993 год уже начали экспериментировать с цветами.У некоторых сайтов появились логотипы и элементы «декора». Логотипы были простыми и картинки все оставались «gif» изображениями,но было еще одно заметное нововведение- это появление «footer» блока, где были ссылки на странички компаний.  
С 1994 года картина уже становиться интереснее. Например сайт «Microsoft» выполнен в виде полусферы ядра, где снизу пиксельные звезды, а над «текстовым» лого, восходит солнце. Естественно все это было простой картинкой. На сайте был текст “If your browser doesn't support images, we have a text menu as well.” перевод : если ваш браузер не поддерживает изображения,то у нас есть текстовое меню.”. Тогда не все браузеры могли в поддержку изображений , а сейчас такое выглядит просто забавной деталью.

С 1994 года на любой странице были использованы изображения в основном , как логотипы.

С 1995 года сайты начинают экспериментировать с блочной версткой и цветами. Некоторые сайты имеют кислотные цвета или фоновые изображения. Начинают изменяться цвета ссылок. Иногда вместо текстовых ссылок используются картинки.

Из общих черт можно выделить явное разделение сайта на header,body и footer. В хэдере иногда находились навигационные ссылки, а «body» был основной контекст, а в футере контактная и юридическая информация.

С 1996 года сайты все больше начинаю экспериментировать.  
Сайт «GameSpot» с новостями в мире видеоигр имеет максимально кислотный и нагроможденный дизайн. Все изображения по прежнему изображения в формате «gif»,но теперь некоторые изображения анимированны. Появились строки поиска и списки новостей.

Сайт «McDonald’s» выполнен довольно оригинально: Клоун приветствует посетителя и предлагает использовать навигационное меню. Примечательно оригинальное и необычное оформление сайта,но глаза не разбегаются , ведь информации и текста на странице мало.Тут своеобразно стилизированный,но привычный «nav-bar». Навигационный блок. Ссылки на «форум», «трудоустройство», акции которые проходят сейчас и мероприятия.

По нашей теме можно выделить 3 сайта: «Lego» , «Adobe». В уже заметной тенденции создавать в хэдере навбары, эти два сайта делают главное: у посетителя не разбегаются глаза и он может удобно ориентироваться на сайте.

На сайте производителя конструктора «Lego» навбар выполнен максимально просто , понятно,но очень оригинально.Главное навигационное меню содержит ссылки на сайт с играми, информацию о бренде, обучающие страницы, карту мира в стиле «лего» и т.д. Оригинальность заключается в том,что каждую ссылку на раздел сопровождает человечек lego. Например ссылку на главную страницу , так называемую «домашнюю» страницу сопровождает картинка с домиком из конструктора лего. Ссылку на страницу с контактной информацией офисов компании сопровождает человечек с письмом. Это пример оригинального использования своей продукции для создания удобного и понятного интерфейса.

Оформление хэдера у на сайте компании «Adobe» очень близок к привычному для нас: Лого сайта, навигационное меню для пользователя, простые черные блоки с белым текстом. Дизайн и подбор цветов выдают старину ,но любой попавший на этот сайт, легко бы перемещался по нему.

Можно сделать вывод: к 1996-1997 уже формируется удобные навигационные меню.

1997 примечателен трендом на подобие «гамбургерного» навигационного меню,это меню выполненное, по принципе гамбургера, последовательно друг за другом, как табличный столбец. Разные компании помещают на разные стороны. Например у «Apple » слева , а у «Disney» , видимо решившей последовать тренду , справа. По моему мнению: дизайн у Дисней хуже из за отсутствия стилизации, это просто список из ссылок, а у «Apple» , по сути такой же список, но за счет использования оформления,оно выглядит понятный и не пугающим.

Просмотровая страницы дальше я не увидел ничего примечательного. Многие сайты имели собственный дизайн,который нельзя привязать к каком-то паттернам или шаблонам, например сайты , которые сделаны в качество промо-материала к фильмам.

Следующей ключевой датой могу обозначить 2005 год.

Начав листать каталог сайтов я увидел необычное количество сайтов с навбаром сверху. Почти все сайты известных брендом перешли к такому виду верстки, максимум- у парочки сайтов навбар был слева.

Несколько сайтов у которых навбар сверху: сайт «Apple», «Sony Pictures», «Steam», «IKEA», «LEGO», «Netflix», «Kodak» «Electronic Arts», «AE Games». «Abode» обновили внешний вид сайта,но хэдер все также находится сверху, значит он прижился.С каждым годом все большей сайтов будет переходить к такому вижу расположения навбара. Сайты, которые перешли к такому шаблону верстки до и к 2010. «Nokia», «CNN», «Skype», «IGN», «Microsoft» , «Marvel». Все сторонние сайты , которые принадлежат «Apple» имеют один и тот же хэдер. Таким образом: к 2010 самые популярные сайты выбрали для

**1.4 История и применение «MySQL»**

MySQL — это система управления базами данных (СУБД), предназначенная для хранения и обработки больших объёмов структурированных данных. Она обеспечивает надёжное хранение информации и быстрый доступ к ней, а также предоставляет широкие возможности для управления данными, включая создание, изменение и удаление таблиц, индексов и связей между ними.

MySQL также обладает высокой производительностью, масштабируемостью и гибкостью. В системе предусмотрена возможность горизонтального и вертикального масштабирования, а также широкий набор функций, которые можно настраивать и расширять в соответствии с изменяющимися требованиями.

MySQL обеспечивает безопасность данных, предоставляя инструменты для шифрования, аутентификации и аудита. В целом, эта СУБД является эффективным инструментом для работы с данными и применяется в различных областях, где требуется эффективная система управления базами данных.

**История MySQL**

Основатель MySQL — Михаил Видениус, также известный как Monty, из шведской компании TcX. В 1979 году он создал систему управления базами данных под названием UNIREG. В дальнейшем UNIREG была расширена для работы с большими базами данных и переписана на разных языках. В 1994 году компания TcX начала разрабатывать веб-приложения с использованием UNIREG. Однако из-за высоких затрат UNIREG не могла эффективно использоваться для динамического создания веб-страниц. Тогда Видениус связался с автором mSQL Дэвидом Хьюзом и предложил интегрировать mSQL с обработчиком B+ ISAM в UNIREG.

Хьюз успешно работал над проектом mSQL 2, и компания решила создать собственную систему управления базами данных. Они взяли за основу UNIREG, использовали инструменты сторонних разработчиков для mSQL и создали API, совместимое с API для mSQL. Исходный код новой базы данных был полностью оригинальным. В мае 1995 года компания TcX выпустила MySQL 1.0, полностью соответствующий их требованиям.

Название MySQL происходит от префиксов каталогов и библиотек в компании TcX, а также имени дочери основателя. С момента выпуска в интернете MySQL стала популярной базой данных благодаря множеству программистов, работающих над её развитием.

В 1995 году Дэвид Оксмарк начал кампанию по распространению MySQL через интернет. Версия 3.11.1 была выпущена в 1996 году в виде бинарного дистрибутива для операционных систем Linux и Solaris. Сегодня MySQL работает на разных платформах и доступна как в двоичном коде, так и в исходном тексте.

Одна из причин популярности MySQL среди пользователей PHP — это то, что поддержка сервера включена в поставку PHP. Благодаря своим характеристикам и простому интерфейсу, MySQL стал самым популярным инструментом для работы с базами данных в PHP.

Лицензионная политика MySQL более гибкая по сравнению с другими базами данных. MySQL распространяется бесплатно, кроме случаев продажи или предоставления услуг на основе этой базы данных.

MySQL обладает отличной переносимостью и может быть использована на разных операционных системах, аппаратных средствах и серверах. Также она способна обрабатывать большие базы данных с миллионами записей.

MySQL — это система управления базами данных (СУБД), предназначенная для хранения и обработки больших объёмов структурированных данных. Она обеспечивает надёжное хранение информации и быстрый доступ к ней, а также предоставляет широкие возможности для управления данными, включая создание, изменение и удаление таблиц, индексов и связей между ними.

MySQL также обладает высокой производительностью, масштабируемостью и гибкостью. В системе предусмотрена возможность горизонтального и вертикального масштабирования, а также широкий набор функций, которые можно настраивать и расширять в соответствии с изменяющимися требованиями.

MySQL обеспечивает безопасность данных, предоставляя инструменты для шифрования, аутентификации и аудита. В целом, эта СУБД является эффективным инструментом для работы с данными и применяется в различных областях, где требуется эффективная система управления базами данных.

Плюсы MySQL:

Безопасность: MySQL предлагает функции безопасности, такие как установка привилегий пользователя, шифрование данных, аутентификация и аудит, для обеспечения безопасности информации.

Пример: MySQL позволяет установить привилегии пользователя, определяя уровень доступа к данным для разных пользователей или групп пользователей.

Производительность: MySQL обладает хорошей производительностью и быстродействием благодаря оптимизированным алгоритмам выполнения запросов и работе с большими объёмами данных.

Пример: MySQL может обрабатывать большие объёмы данных с минимальными задержками, что полезно для веб-приложений с высокой нагрузкой.

Масштабируемость: MySQL может использоваться для различных масштабов проектов, от небольших веб-приложений до крупных корпоративных систем, предлагая различные методы масштабирования.

Пример: MySQL поддерживает горизонтальное и вертикальное масштабирование, что позволяет расширять базы данных при необходимости и обрабатывать большое количество одновременных запросов.

Гибкость: MySQL поддерживает разные типы данных, индексы, хранимые процедуры, триггеры и другие функции, обеспечивая гибкость при обработке данных.

Пример: MySQL позволяет создавать сложные схемы данных, используя различные типы данных и индексы.

Бесплатность и открытость исходного кода: MySQL имеет открытый исходный код и доступен бесплатно, что делает его доступным для широкого круга разработчиков и пользователей.

Пример: разработчики могут использовать исходный код MySQL для создания собственных модификаций и улучшений системы, а пользователи могут скачать и использовать MySQL бесплатно.

Общность и поддержка: MySQL имеет большое сообщество разработчиков и пользователей, предоставляющее помощь и ресурсы при возникновении проблем.

Пример: разработчики и пользователи могут найти ответы на свои вопросы на форумах и в блогах сообщества MySQL, а также получить поддержку от опытных специалистов.

Минусы MySQL:

Ограничения на производительность: MySQL может иметь проблемы с производительностью при обработке больших объёмов данных или когда требуется обработка сложных запросов или подключение большого количества клиентов.

Пример: при высокой нагрузке на систему или большом количестве одновременных запросов MySQL может замедлить работу или отказать в обслуживании.

Ограниченный набор типов данных: MySQL имеет ограниченный набор типов данных по сравнению с некоторыми другими СУБД.

Пример: MySQL не поддерживает все типы данных, доступные в других СУБД, что может потребовать использования дополнительных инструментов или преобразований данных для работы с определёнными типами данных.

Ограничения на масштабируемость и доступность: MySQL может столкнуться с ограничениями в масштабируемости и доступности при взаимодействии с большим количеством клиентов.

Пример: при высокой нагрузке на систему MySQL может столкнуться с проблемами доступности, такими как отказы в обслуживании или снижение производительности.

Сложность администрирования: MySQL может быть сложным для администрирования, особенно для новичков.

Пример: настройка и оптимизация параметров конфигурации MySQL может быть сложной задачей, требующей глубоких знаний и опыта.

Уязвимости в области безопасности: MySQL может иметь уязвимости в сфере безопасности, такие как атака SQL-инъекцией или недостаточная защита данных.

Пример: для обеспечения безопасности баз данных необходимо принимать дополнительные меры, такие как использование подготовленных запросов, проверка ввода данных и регулярное обновление системы.

**Источник :https://servergate.ru/articles/mysql-preimushchestva-i-nedostatki/.**

**Источник :https://inmotion.live/blog/notes/mysql-story/.**

**1.5 История и применение «PHP»**

В настоящее время PHP поддерживается большинством хостинг-провайдеров и считается лидером среди инструментов для создания динамических веб-сайтов.

История развития PHP началась в 1994 году с релиза первого релиза PHP, который был основан на CGI-скрипте на Perl и собирал статистику о просмотрах резюме разработчика Расмуса Лердорфа. В середине 1995 года основной модуль PHP был переписан на C, и появились первые теги для вставки кода в HTML. В 1996 году вышла вторая версия PHP, а к 1997 году в сети насчитывалось более 50 000 сайтов, использующих возможности языка.

В 1998 году Энди Гутманс и Зив Суразки создали PHP 3 на основе разработок Лердорфа, добавив поддержку API и объектно-ориентированного синтаксиса. В 2000 году вышла четвёртая версия PHP, которая стала ещё более производительной и поддерживала буферизацию вывода и HTTP-сессии. В 2004 году была выпущена пятая версия PHP с поддержкой XML и изменением системы программирования на похожую на Java.

Сейчас более распространены 6 и 7 версии PHP, а с 2020 года официально выпущен 8 релиз PHP. Несмотря на отказ компании Microsoft от поддержки языка, энтузиасты продолжают развивать и совершенствовать PHP.

На сегодняшний день PHP является одним из наиболее распространенных языков веб-программирования. Подавляющие большинство сайтов и веб-сервисов в интернете написано с помощью PHP. По некоторым оценкам PHP применяется более чем на 80% сайтов, среди которых такие сервисы, как facebook.com, vk.com, baidu.com и другие. И такая популярность неудивительна. Простота языка позволяет быстро и легко создавать сайты и порталы различной сложности.

PHP был создан в 1994 году датским программистом Расмусом Лердорфом и изначально представлял собой набор скриптов на другом языке Perl. Позже этот набор скриптов был переписан в интерпретатор на языке Си. И с самого возникновения PHP (сокращение от PHP: Hypertext Preprocessor - PHP: Препроцессор гипертекста) представлял удобный набор инструментов для упрощенного создания веб-сайтов и веб-приложений.

Области применения PHP

PHP — это серверный язык программирования, который имеет ряд преимуществ. Благодаря этому он стал основой для создания веб-приложений, где пользователи получают результаты вычислений, а всю нагрузку по ним берёт на себя удалённый сервер. Это привело к появлению таких популярных CMS, как WordPress, Joomla и Drupal, а также фреймворков Laravel, Yii2 и Symfony.

PHP используется в трёх основных направлениях:

Написание скриптов, исполняемых на стороне сервера. Основное применение PHP — создание сайтов, размещаемых на заранее настроенном хостинге.

Написание скриптов, исполняемых в командной строке независимо от веб-сервера и браузера. Примеры приложений — Task Scheduler под Windows и cron под Linux.

Разработка приложений GUI, исполняемых на стороне клиента. Основная сфера применения — создание кроссплатформенных приложений.

Язык PHP поддерживается практически всеми платформами, включая Microsoft Windows, macOS, GNU Linux, Unix, RISC OS, OS/2 и другие. Он также совместим с популярными web-серверами, такими как Apache, Personal Web Server, Microsoft Internet Information Server, Netscape, Oreilly Website Pro, iPlanet, OmniHTTPd, Xitami и Caudium.

Перспективы применения PHP

Сегодня активно используются аналоги PHP, такие как Angular, React, Node.js и Python. Некоторые разработчики даже рассматривают переход на Java и С. Однако PHP продолжает развиваться и адаптироваться к новым технологиям и требованиям.

Преимущества использования PHP:

низкий порог вхождения;

проверенные инструменты разработки;

крупное профессиональное сообщество;

большая библиотека готовых решений;

нетребовательность к рабочему окружению.

Для новичков это отличная платформа для изучения разработки веб-сайтов. Синтаксис языка входит в группу С-подобных, поэтому после PHP легко осваивать Java, C#, C++ и подобные платформы. Переход на Python или Goland может потребовать больше усилий, но это вопрос техники и желания освоить новые направления.

Платформа продолжает развиваться, внедряя лучшие практики других языков и создавая фреймворки, которые сами по себе являются «новыми языками». PHP только выигрывает от этого благодаря своей надёжности и универсальности.

**Источник:https://timeweb.com/ru/community/articles/obzor-yazyka-php-proshloe-nastoyashchee-perspektivy**

**1.6 История и применение HTML**

С момента своего возникновения Всемирная паутина прошла долгий путь. Она развивалась не только в содержании и дизайне, но и в использованных технологиях. Одной из ключевых ролей в этом развитии сыграл язык разметки гипертекста HTML. В этом посте мы отправимся в путешествие во времени, чтобы исследовать этот эволюционный процесс, начиная с ранних версий HTML и заканчивая последней версией - HTML5.

HTML 1.0: Начало интернета

HTML был впервые представлен в начале 1990-х годов, когда Тим Бернерс-Ли создал Всемирную паутину. Первая версия HTML была простым языком с ограниченными возможностями, предназначенным для структурирования текста. Хотя эта версия заложила основы интернета, в ней не было мультимедийных возможностей и интерактивности, которые мы привыкли видеть сегодня.

HTML 2.0: Развитие

Первый вариант стандарта включал в себя много интересностей:

теги для создания таблиц,

разметки математических формул,

обтекание изображений текстом и др.

Авторы вдруг вспомнили, что по своей идее HTML должен только размечать структуру документа и не должен содержать параметры графических стилей отображения элементов в браузере.

При создании HTML 3.0 разработчики ломали голову, как решить нестыковку между идеологией структурной разметки и потребностями пользователей, которых гораздо больше интересовал внешний вид веб-страницы.

И чтобы это противоречие не изменило первоначальные свойства HTML, создатели третьей версии решили добавить к ней поддержку нового средства, которое служило для оформления веб-страниц.

система CSS:

формально не зависит от HTML,

у неё свой, отличный от HTML, синтаксис,

На неё не влияют идеологические ограничения HTML,

Она позволяет задавать параметры внешнего представления для любого тега HTML.

С помощью CSS создатель веб-страницы наконец мог спокойно менять шрифт и размер чего угодно, и что важно – это стало возможным не за счет внедрения кучи новых тегов, а с помощью механизма, который влиял на интерпретацию уже существующих тегов HTML.

Чтобы наглядно продемонстрировать чудеса, на которые способен CSS, канадец Дейв Ши создал сад CSS дзена – место, где сотни веб-разработчиков медитируют верстая одну и ту же HTML-страницу с абсолютно разными стилями CSS.

HTML 3.2: Стандартизация

Выпущенный в 1997 году, HTML 3.2 ввел концепцию CSS для улучшенного контроля над внешним видом страницы. Таблицы и рамки также были улучшены, делая веб-дизайн более изысканным.

HTML 4.0: Мультимедийные возможности

HTML 4.0, также выпущенный в 1997 году, был революцией в области мультимедийных возможностей интернета. Он добавил поддержку изображений, аудио и видео, а также возможность создания более динамичных страниц.

XHTML 1.0: переход к XML

XHTML 1.0, представленный в 2000 году, стал шагом к более строгой XML-синтаксике. Это сделало HTML совместимым с другими языками XML и способствовало использованию более чистого и структурированного кода. Однако существенных нововведений в нём не было.

HTML5, представленный в 2014 году, стал значительным шагом вперёд в развитии HTML. Он обладает широким набором функций, что делает его идеальной платформой для веб-разработки. Вот некоторые ключевые особенности HTML5:

Поддержка мультимедиа: встроенные аудио- и видеоэлементы позволяют интегрировать мультимедийный контент без использования плагинов.

Улучшенная семантика: новые структурные элементы, такие как <заголовок>, <навигация> и <статья>, помогают точнее определить содержимое веб-страниц, улучшая поиск и доступность.

Canvas и SVG: элемент позволяет создавать динамическую графику и анимацию, а поддержка масштабируемой векторной графики (SVG) помогает создавать высококачественные изображения.

Геолокация и автономные приложения: HTML5 поддерживает геолокацию, что позволяет создавать локационные приложения, а также создавать автономные веб-приложения с помощью сервисных работников.

Веб-API: HTML5 предоставляет разнообразные JavaScript-API, которые упрощают реализацию различных функций, таких как перетаскивание и локальное хранилище, повышая интерактивность в интернете.

Адаптивный веб-дизайн: благодаря внедрению медиа-запросов и гибких макетов, HTML5 способствует созданию адаптивных веб-сайтов, которые хорошо отображаются на разных устройствах.

Развитие HTML от первых версий до HTML5 было захватывающим. Каждая новая версия вносила важные изменения в развитие веб-технологий, делая интернет более интерактивным, визуально привлекательным и доступным. HTML5 продолжает менять ландшафт веб-разработки, предоставляя разработчикам и дизайнерам новые интересные возможности. Это свидетельствует о постоянном изменении мира технологий и стремлении к инновациям в сети.

Источник:https://codecloudcafe.com/the-evolution-of-html-from-html-to-html5/

**1.7 История и применение CSS**

HTML и CSS тесно связаны, и кажется, что они всегда были вместе. Однако до 1994 года, когда Марк Андреессен написал своё известное письмо, CSS ещё не существовал, и веб-страницы нельзя было оформить с помощью HTML.

В 1994 году Андреессен отметил, что веб-дизайнеры не могли использовать HTML для оформления веб-страниц, и извинился перед ними за неудобства. Однако уже через 10 лет CSS стал основным инструментом для оформления веб-страниц.

Путь к принятию CSS современным веб-сообществом был непростым. Разработчики предлагали различные подходы к веб-вёрстке, включая языки стилей от Пей Яун Вэй, Андрисена и Хаконвиума Ли.

В 1991 году Пэй-Юань Вэй создал графический браузер Violaww и интегрировал свой язык стилей в браузер. Однако его синтаксис стиля не стал официальным стандартом стиля в интернете.

Андреессен, в свою очередь, предложил расширить HTML-теги для оформления веб-страниц. Однако этот подход оказался неудачным, и сообщество разработало множество альтернативных языков стилей, таких как RRP, PSL96 и другие.

Окончательно был принят каскадный список стилей (CSS), предложенный Хаконом Виумом в октябре 1994 года. Этот язык стал основным инструментом для оформления веб-страниц и используется до сих пор.

Андреессен, тем временем, попробовал что-то другое со своим браузером Netscape. Вместо создания отдельного языка разметки он применил подход расширения HTML-тегов, включив нестандартизированные HTML-теги для оформления веб-страниц. К сожалению, прошло совсем немного времени, прежде чем страница утратила всю семантизацию и стала выглядеть следующим образом:

<MULTICOL COLS="3" GUTTER="25">

<P><FONT SIZE="4" COLOR="RED">Это будет какой-нибудь шрифт, разбитый на столбцы</FONT></P>

</MULTICOL>Скопируйте код

Разработчики быстро поняли, что такая попытка безнадежна. Сообщество random Web разработало множество других альтернатив, таких как RRP, язык таблиц стилей, который использует очень краткие сокращения; PSL96 - язык, поддерживающий функции и операторы состояния. Если вам интересно узнать, как именно выглядят эти языки, ознакомьтесь с отличной статьей Зака Блума о сравнении.

Язык, который был окончательно принят, - это грамматика стилей, предложенная Хаконом Виумом в октябре 1994 года. Это называется каскадный список стилей, или сокращенно CSS.

Почему мы используем CSS

Основной причиной успеха CSS стала его простота, которая делает его более понятным, чем современные конкуренты. Синтаксис раннего CSS был таким:

window.margin.left = 2cm

font.family = times

h1.font.size = 24pt

CSS — это язык описания, а не указания, как браузер должен отображать страницу. Вместо этого мы описываем правила, которые применяются к HTML-документу одно за другим, и позволяем браузеру обрабатывать рендеринг. Так как веб-разработка в основном выполняется любителями и энтузиастами, CSS имеет предсказуемый и понятный формат, что упрощает его использование. Это означает, что CSS работает, даже если синтаксис неверен, что является особенностью, а не ошибкой.

CSS также уникален благодаря поддержке каскадных стилей, как описано в его полном названии «style cascade». Каскадирование означает, что стили могут наследовать и переопределять другие стили, следуя определённым правилам, и CSS поддерживает использование нескольких таблиц стилей на одной странице.

Обратите внимание на процентное соотношение в исходном синтаксисе CSS? По мнению Ли, это очень важный момент, так как пользователи и разработчики могут определять стили по-разному, и браузер выступает посредником между ними, согласовывая различия для рендеринга страницы. Приведённый выше процент отражает вес стиля, и чем меньше вес, тем легче его переопределить. Когда Ли впервые продемонстрировал CSS, он даже добавил ползунок для переключения между пользовательскими и разработческими стилями в браузере.

На заре CSS это вызвало большие споры, и некоторые считали, что разработчики должны иметь полный контроль над стилями, а другие настаивали на том, что пользователи должны иметь некоторый контроль. В итоге этот процент был удалён, чтобы обеспечить более чёткое правило покрытия, но именно поэтому концепция веса поддерживается в современном CSS.

Вскоре после этого Ли опубликовал своё первоначальное предложение, и он нашёл партнёра в лице Берта Боса, разработчика браузера Argo, Bos, который также определил язык стилей, совместимый с его браузером, который затем был частично включён в CSS. Затем пара разработала более детальный стандарт и обратилась к W3C, новой рабочей группе по созданию HTML, для его продвижения.

После многих лет работы, к концу 1996 года синтаксис CSS стал таким:

«

html {

margin-left: 2cm;

font-family: "Times", serif;

}

h1 {

font-size: 24px;

}» **источник: https://www.mo4tech.com/a-brief-history-of-css.html**

**1.8 История и применение JavaScript**

Прошлое

В 1995 году Netscape Navigator был лидером среди браузеров, и компания Netscape решила добавить интерактивность на HTML-страницы с помощью лёгкого языка программирования. Для разработки этого языка был приглашён Брендан Айк. Через десять дней, в мае 1995 года, была создана первая версия языка, изначально называвшегося Mocha. В сентябре того же года название было изменено на LiveScript в стиле Netscape. В декабре Netscape и Sun заключили лицензионное соглашение, которое определило окончательное название языка — JavaScript. В то же время JavaScript был включён в Netscape Navigator 2.0B3.

Название «JavaScript» указывает на первоначальное предназначение языка: язык Java от компании Sun предоставлял основные строительные блоки для сетевых приложений, в то время как JavaScript должен был служить связующим звеном между этими блоками. Очевидно, что разделение ответственности произошло на стороне пользователя: сегодня JavaScript доминирует в браузерах, в то время как Java практически исчезла.

JavaScript формировался под влиянием нескольких языков программирования: диалект Scheme языка Lisp привнёс правила области видимости переменных, включая замыкания. Из языка программирования Self, потомка Smalltalk, было заимствовано наследование на основе прототипов (объектно-ориентированное в отличие от классово-ориентированного).

Так как предполагалось, что JavaScript будет поддерживать Java, руководство Netscape потребовало, чтобы синтаксис нового языка был похож на синтаксис Java. Это исключило использование синтаксиса существующих скриптовых языков, таких как Perl, Python или TCL. Однако, хотя синтаксис JavaScript похож на синтаксис Java, это достаточно разные языки.

Стандартизация\*\*

После появления JavaScript корпорация Microsoft разработала похожий язык с изменённым названием JScript для Internet Explorer 3.0 (август 1996 года). Компания Netscape решила стандартизировать язык и обратилась в международную организацию по стандартизации Ecma International для создания стандарта. Работа над спецификацией под названием ECMA-262 началась в ноябре 1996 года. Первая версия ECMA-262 вышла в июне 1997 года. Из-за того что компания Sun (ныне Oracle) владела правами на торговую марку со словом Java, язык, описываемый стандартом, не мог называться JavaScript. Поэтому было выбрано название ECMAScript — это название стандарта для языка, а его реализации официально называются JavaScript, JScript и так далее. Вне официальных документов для обозначения реализации чаще всего используют название JavaScript. Текущая версия языка ECMAScript — это 5.1, или, другими словами, ECMA-262 в редакции 5.1. Эта версия также стала стандартом ISO ISO/IEC 16262:2011.

Язык ECMA-262 развивается под руководством технического комитета №39 организации Ecma (сокращённо ТК39). Его членами являются такие компании, как Microsoft, Mozilla и Google, которые назначают своих сотрудников для участия в работе комитета. Среди прочих членов комитета есть три человека: Брендан Эйк (автор языка), Аллен Вирфс-Брок (инженер-разработчик языка JavaScript в Microsoft, один из редакторов стандарта) и Дэвид Херман. Работа ТК39 включает обсуждение дизайна следующих версий с использованием открытых источников, таких как списки рассылки.

Достижение согласия при создании стандарта не всегда бывает лёгким, но именно благодаря этому JavaScript является действительно открытым языком, который поддерживается большим количеством производителей программного обеспечения и отличается высокой степенью совместимости. Такая совместимость стала возможной благодаря подробной и чёткой спецификации. Например, для определения определённого поведения используется псевдокод. Спецификация дополняется набором тестов, который называется test262. Он проверяет реализации языка на соответствие спецификации. Интересно, что ECMAScript находится под управлением консорциума W3C. ТК39 и W3C сотрудничают во всех областях, где пересекаются JavaScript и HTML5.

Исторические вехи JavaScript

JavaScript потребовалось много времени, чтобы завоевать признание. Многие технологии существуют некоторое время, прежде чем становятся популярными. Эта часть рассказывает о том, что произошло с момента создания JavaScript до сегодняшнего дня. Обычно упоминаются только самые популярные проекты, а многие другие остаются без внимания, даже если они были созданы в то же время. Вот два примера: часто упоминается Dojo Toolkit, в то время как qooxdoo менее известен, хотя и был создан одновременно с ним. Также все знают о Node.js, но мало кто помнит о Jaxer, который существовал до него.

1997 — Dynamic HTML (http://msdn.microsoft.com/en-us/library/ms533044(v=vs.85).aspx). Dynamic HTML позволяет динамически изменять содержимое и внешний вид веб-страницы. Это достигается путём манипулирования объектной моделью документа (Document Object Model — DOM): изменение содержимого, стиля, отображения и скрытия элементов и т. д. Dynamic HTML впервые появился в Internet Explorer 4 и в Netscape Navigator 4.

1999 — XMLHttpRequest. Это API позволяло скрипту на стороне клиента отправлять HTTP или HTTPS запросы к серверу и получать обратно данные, преимущественно в текстовом формате (XML, HTML, JSON). Оно было введено в Internet Explorer 5.

2001 — JSON, основанный на JavaScript формат обмена данными. В 2001 году Дуглас Крокфорд (Douglas Crockford) дал название и задокументировал JSON (JavaScript Object Notation) — способ в стиле языка Lisp, использующий синтаксис JavaScript и позволяющий сохранять данные в текстовом формате. JSON использует литералы объектов, массивов, строк, чисел и логических значений JavaScript для представления структур данных. Например:

{

"first": "Jane",

"last": "Porter",

"married": true,

"born": 1890,

"friends": [ "Tarzan", "Cheeta" ]

}

С течением времени JSON стал популярной и лёгкой альтернативой XML, особенно когда речь идёт о представлении данных без разметки. Действительно, работать с JSON в JavaScript очень удобно.

2004 — Dojo Toolkit, фреймворк для программирования на JavaScript (?). Dojo Toolkit упрощает процесс программирования в целом, предоставляя необходимую инфраструктуру: библиотеку наследования, модульную структуру, API для реализации графических компонентов в стиле настольных приложений и так далее.

2005 — Ajax, приложения, похожие на настольные, но созданные на базе браузера. Ajax — это набор технологий, делающих веб-страницу интерактивной и способной конкурировать с настольными приложениями. В 2005 году впечатляющим примером того, чего можно достичь с помощью Ajax, стал сервис Google Maps. Вы можете панорамировать и масштабировать карту всего мира, в то время как браузер загружает только видимую её часть. После появления Google Maps Джесси Джеймс Гарретт (Jesse James Garrett) заметил, что Google Maps имеет много общих черт с другими веб-сайтами. Он назвал эти функции «Ajax», сокращённо от Asynchronous JavaScript and XML (асинхронный JavaScript и XML). Двумя основными составляющими Ajax являются: во-первых, асинхронная фоновая загрузка (через XMLHttpRequest) и, во-вторых, динамическое обновление текущей страницы на основе полученных данных (через Dynamic HTML). Это значительно улучшило удобство использования по сравнению с необходимостью постоянно перезагружать страницу.

Ajax стал значительным прорывом в области JavaScript и динамических веб-приложений. Интересно, что на разработку всех этих технологий ушло много времени: отдельные компоненты Ajax были доступны в течение нескольких лет. С появлением Ajax стали популярны другие форматы данных (JSON вместо XML), используются другие протоколы (например, Web-сокеты в дополнение к HTTP) и возможна двусторонняя связь. Однако базовые технологии остались прежними. Несмотря на это, термин Ajax используется всё реже и почти полностью заменён более комплексным понятием — HTML5 (JavaScript и браузер-API).

2005 — Apache CouchDB, база данных, ориентированная на JavaScript. Грубо говоря, CouchDB — это JSON-база данных, которая позволяет хранить объекты JSON без определения схемы базы данных и предоставляет возможность определять представления и индексы с помощью функций, выполняющих операции записи и чтения. Таким образом, CouchDB отлично подходит для JavaScript, поскольку вы можете работать напрямую с объектами языка. В сравнении с реляционными базами данных в CouchDB отсутствует потеря соответствия при отображении данных, а в сравнении с объектно-ориентированными базами данных вы избавляетесь от множества сложностей, так как сохраняются только данные, без поведения объекта. CouchDB — это одна из подобных NoSQL баз данных, но с лучшей поддержкой JavaScript.

2006 — jQuery, облегчает работу с DOM. Объектная модель документа (Document Object Model) — одна из самых сложных частей разработки клиентских приложений. jQuery упрощает процесс работы с DOM, абстрагируясь от конкретного браузера и предоставляя мощный API для выборки и изменения DOM-объектов.

2007 — WebKit положил начало разработке веб-приложений для мобильных устройств. WebKit основан на предыдущей версии KDE, это HTML-движок, анонсированный в 2003 году, а его исходный код был открыт в 2005 году. С выходом iPhone в 2007 году, он стал основой для веб-разработки мобильных устройств. Сегодня WebKit — основной движок для Android и единственный для iOS, и доминирует на мобильном рынке. Это означает, что если вы хотите создать кросс-платформенное мобильное приложение, веб-технологии будут лучшим выбором (если вам не нужны проприетарные функции).

2008 — V8, JavaScript может быть быстрее. Когда Google анонсировал веб-браузер Chrome, одной из его особенностей стал быстрый JavaScript-движок V8. Он изменил представление о JavaScript как о медленном и вызвал гонку разработчиков. V8 — это проект с открытым исходным кодом, который можно использовать как независимый компонент там, где требуется встроенный и широко используемый язык.

2009 — Node.js, JavaScript на сервере. Node.js позволяет создавать высокопроизводительный сервер, используя событийно-ориентированный и неблокирующий ввод-вывод на JavaScript. Создатель Node.js Райан Дал упоминает следующие причины, которые послужили основанием для выбора JavaScript:

Поскольку JavaScript не является изолированным и не имеет собственного API ввода, Node.js может предложить свой собственный неблокирующий API.

Web-разработчики уже активно используют JavaScript, так как это широко известный язык программирования, особенно в контексте веб-разработки.

API для работы с DOM событийно-ориентировано, и разработчики привыкли работать без потоков и циклов событий.

Дал опирался на существующие событийно-ориентированные серверы и серверный JavaScript (в основном на проект CommonJS).

Привлекательность Node.js для JavaScript-программистов заключается не только в возможности использовать знакомый язык, но и в том, что можно применять один и тот же язык как на клиенте, так и на сервере. Это позволяет выполнять сложные задачи, например, адаптацию для браузеров, которые не могут обрабатывать сложный JavaScript-код. Сборка страницы может происходить на сервере с использованием того же кода, который используется на обычном клиенте (например, web-сервер FunctionSource, Yahoo Cocktails).

Благодаря динамическим веб-приложениям JavaScript изначально приобрёл популярность. Веб стал отличной средой для разработки приложений, и было необходимо, чтобы JavaScript стал её неотъемлемой частью. С тех пор в JavaScript было добавлено множество новых возможностей для повышения его универсальности. Такое быстрое развитие стало возможным благодаря современным JavaScript-движкам, которые позволяют использовать JavaScript-ориентированные базы данных и обмениваться данными с другими системами (например, веб-сервисами) через JSON. Серверный JavaScript позволяет использовать один и тот же язык как на сервере, так и на клиентской стороне. Кроме того, Node.js позволяет использовать JavaScript для скриптов сборки и скриптов командной строки. И наконец, JavaScript, возможно, является самым открытым языком программирования из существующих: нет единой группы, контролирующей язык; полная спецификация языка (значение которой невозможно переоценить); доступно несколько хорошо совместимых реализаций.

Будущее\*\*

Будущее приносит для JavaScript множество интересных решений: ECMAScript.next исправит некоторые причуды языка и добавит новые возможности. JavaScript станет лучшим целевым языком для компилятора (?). Поддержка параллельных вычислений важна, и сейчас рассматривается несколько возможных вариантов. И в завершение, HTML5 становится отличной основой не только для web-приложений, но также для мобильных и настольных приложений.

Источник:https://github.com/vvscode/tr--The-Past--Present--and-Future-of-JavaScript?tab=readme-ov-file#%D0%B8%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5-%D0%B2%D0%B5%D1%85%D0%B8-javascript

**1.9 История и применение фреймворка «Bootstrap»**

Разработка веб-страницы с точки зрения вёрстки — это непрерывный процесс создания и доработки блоков. Чем больше проект, тем чаще происходит итерация между идеей, функционалом и вёрсткой. Важно решать задачи бизнеса быстро и качественно. Качество вёрстки зависит от верстальщика, а скорость определяется командой.

В 2010 году в компании Twitter появился проект Twitter Blueprint, целью которого было создание системы для разработки новых компонентов внутри компании. Blueprint включал набор готовых базовых блоков с дизайном, используемых в компании. Благодаря своей простоте это решение стало популярным среди сотрудников, имеющих минимальный опыт разработки. Любой мог создать новый концепт блока и предложить его для дальнейшей разработки.

В том же году на Twitter Hack Week проект стал популярным не только внутри компании, но и за её пределами. Вскоре после этого, в 2011 году, проект стал доступен для всех под названием Bootstrap. С тех пор Bootstrap занимает лидирующие позиции среди инструментов для создания сайтов. Согласно различным статистикам, фреймворк используют от 17 до 30% всех сайтов в интернете. Код Bootstrap можно найти на сайтах таких проектов, как Hexlet, GitHub, PayPal, Spotify, Twitter и Duolingo. Этот список далеко не полный. Таким образом, Bootstrap не является библиотекой для создания админок и прототипов, как о нём говорят.

Bootstrap как готовый набор элементов

Bootstrap предлагает возможность использовать его как набор готовых элементов, что позволяет быстро создавать проекты разной сложности без глубоких знаний HTML и CSS. Для этого достаточно изучить документацию и уметь создавать простые страницы. Например, для создания кнопки используются два класса: один определяет структуру кнопки, а второй — её оформление. Такой подход называется OOCSS (Object Oriented CSS) и описан в блоге Hexlet.

<button class=“btn btn-primary”>Я — синяя кнопка</button>

Другой подход, применяемый в Bootstrap, — Atomic CSS. На нём основаны утилиты фреймворка. Atomic CSS предполагает использование одного класса для одного свойства.

<section class=“d-flex bg-white”>

<!-- HTML вёрстка -->

</section>

В этом примере тег <section> получает два свойства: background-color: #fff; display: flex;. Использование утилит требует более глубоких знаний CSS, так как это похоже на применение правил, но вместо записи в CSS-файле используются классы.

Компоненты и утилиты позволяют быстро создавать проекты разной сложности, однако выбор дизайна ограничен используемыми компонентами. Если не изменять компоненты, страница будет выглядеть так же, как в документации. Это не является недостатком, так как Bootstrap — современный фреймворк с современным дизайном, который не вызывает страха при первом знакомстве. Многие небольшие компании выбирают этот подход.

Плюсы использования Bootstrap как набора готовых элементов:

Скорость. Процесс разработки сопровождается непрерывной работой по добавлению или удалению функционала на странице. Чем крупнее проект, тем чаще происходят эти процессы, и скорость реализации во многом зависит от того, как скоро функционал обрастёт внешним видом. Используя готовые компоненты и утилиты Bootstrap, можно быстро ввести новый функционал на сайт и предоставить его пользователям. Это решает одну из проблем цикла «идея — функционал — вёрстка».

Кроссбраузерность. В веб-пространстве множество различных браузеров: Chrome, Firefox, Edge, Opera, Safari и т. д. Они предоставляют пользователям комфортный веб-сёрфинг. Однако каждый из этих браузеров может обрабатывать HTML и CSS немного по-своему, поэтому верстальщикам необходимо приводить стандартные элементы к единому виду. Такая задача требует скрупулёзного подхода и знания частых проблем и особенностей браузеров. Компоненты Bootstrap учитывают различия браузеров и оптимизированы таким образом, чтобы минимизировать различия между ними.

Адаптивность. Bootstrap имеет готовую 12-колоночную сетку, по которой можно располагать компоненты. Сетка является самым популярным компонентом Bootstrap и используется даже теми, кто скептически относится к фреймворку. Сетка использует подход, при котором можно сразу настроить адаптивность. Используя готовые компоненты, можно быть уверенным в том, что они будут хорошо отображаться на разных устройствах.

Доступность. Разработчики должны учитывать использование страниц людьми с ограниченными возможностями. Документация описывает ситуации, как сделать компонент доступным. Однако и без этого компоненты имеют базовую доступность. Один из самых полезных классов, связанных с доступностью, — sr-only. Он позволяет скрыть элемент для всех устройств, кроме скринридера. Это важно для правильной организации вёрстки форм на сайте.

Низкий порог вхождения. Создание страницы по компонентам не требует экспертных знаний HTML и CSS. Это позволяет создавать новые блоки не только верстальщику, но и любому сотруднику компании, знающему базовые основы вёрстки. Таким образом, разработчики могут оснастить новое решение готовой вёрсткой с использованием фреймворка. Bootstrap также может быть полезен небольшим компаниям, которые не готовы тратить большие деньги на разработку веб-сайта. В качестве промежуточного решения Bootstrap будет незаменим. Впоследствии можно остановиться на нём, как мы сделали на Хекслете.

Минусы использования Bootstrap как готового набора элементов:

Вес проекта.

Bootstrap содержит не только набор классов, но и интерактивные компоненты, для работы которых используются JavaScript библиотеки, увеличивающие вес подключаемых файлов на странице. Вместе с библиотеками JavaScript Bootstrap 4 весит около 300 КБ, что может стать проблемой при разработке мобильных приложений с низким качеством интернет-соединения. Такие приложения могут включать материалы для оказания экстренной помощи, вызова спасательных служб и т. д.

Дизайн.

Стандартные компоненты Bootstrap имеют определённые ограничения в дизайне, которые можно расширять только вглубь, вставляя компоненты внутрь других. Модификация компонентов требует знаний вёрстки и проверки в разных браузерах и разрешениях. Однако дизайн компонентов Bootstrap лаконичен и функционален, что помогает правильно подавать информацию. Даже с такими ограничениями можно создавать удобные и привлекательные сайты благодаря грамотной работе с пользовательским интерфейсом (UI).

Функциональность.

Заранее определённый функционал компонентов может потребовать значительных доработок при изменениях, особенно если речь идёт об интерактивных элементах с использованием JavaScript кода. Разработчики добавили классы для изменения поведения таких элементов, но этого может быть недостаточно.

Bootstrap как фреймворк предоставляет разработчикам обширные возможности для создания пользовательских компонентов благодаря своей большой кодовой базе. Фреймворк можно представить как основу для будущего приложения, на которую будут опираться стены, окна и мебель.

В Bootstrap содержатся десятки функций и миксинов, позволяющих верстальщику создать индивидуальную дизайн-систему. Однако многие разработчики недооценивают этот функционал, что приводит к формированию неправильного мнения о Bootstrap.

При добавлении нового компонента на страницу с использованием Bootstrap процесс разработки часто происходит следующим образом:

Подбор похожих компонентов из документации.

Подбор новых классов для создания новых стилей.

Перезапись и добавление новых стилей для блока в дополнительном CSS-файле.

Такой подход позволяет решить ближайшую задачу, но может вызвать трудности в долгосрочной перспективе из-за отсутствия взаимодействия между компонентом и фреймворком:

Обновление кодовой базы Bootstrap не влияет на поведение компонента, так как разработчики постоянно улучшают и исправляют ошибки в своих версиях.

Изменение настроек не влияет на компонент, например, при смене цветовой схемы. В этом случае необходимо вручную настраивать цвета для каждого пользовательского компонента.

Генерация новых утилит превращается в дублирование кода без создания системы. Например, создание утилит для фона и текста.

Начиная с Bootstrap 5, создание новых утилит решается добавлением нового массива, передаваемого в обработчик и возвращающего новые классы. Как же тогда создаётся новый компонент? Чаще всего достаточно внимательно изучить файл variables.scss.

Что там можно обнаружить? Например, цвета, которые используются в качестве цветовой схемы:

$blue: #0d6efd !default;

$green: #198754 !default;

$cyan: #0dcaf0 !default;

$yellow: #ffc107 !default;

$red: #dc3545 !default;

$gray-100: #f8f9fa !default;

$gray-600: #6c757d !default;

$gray-900: #212529 !default;

$primary: $blue !default;

$secondary: $gray-600 !default;

$success: $green !default;

$info: $cyan !default;

$warning: $yellow !default;

$danger: $red !default;

$light: $gray-100 !default;

$dark: $gray-900 !default;

$theme-colors: (

"primary": $primary,

"secondary": $secondary,

"success": $success,

"info": $info,

"warning": $warning,

"danger": $danger,

"light": $light,

"dark": $dark

) !default;

Цветов в файле \_variables.scss намного больше, но именно эти формируют базовую цветовую схему проекта. Вы можете добавить новые цвета, определить их в качестве базовых и после компиляции все компоненты и утилиты получат новые значения. Именно такое поведение отличает фреймворк от простого набора готовых компонентов.

Именно такая работа с Bootstrap позволит создать нечто большее, чем просто прототип, лендинг или панель администратора. На Hexlet, в качестве третьего проекта по вёрстке Bootstrap используется для создания внешнего вида полноценного чата.

Выводы

Изучение Bootstrap может сильно развить вас как разработчика. Ведь кроме написания кода, можно изучать и чужой, который создан и поддерживается тысячами людей по всему миру. В коде можно найти интересные решения, грамотное структурирование и множество полезных функций. Благодаря этому Bootstrap можно использовать во множестве сценариев: от создания прототипов, до вёрстки сложных макетов, а возможность использование отдельных частей позволяет добавить частичку Bootstrap в проект без необходимости переделывания структуры.

Источник

<https://guides.hexlet.io/ru/bootstrap/>

**2. История и применение фреймворка «Jquery»**

Библиотека jQuery была создана Джоном Ресигом в 2006 году. По словам автора, она должна была устранить разрыв между сложным функционалом JavaScript и простотой веб-разработки. Ресиг, родившийся 8 мая 1984 года, с юных лет проявлял интерес к программированию, который укрепился благодаря его образованию. Он получил степень бакалавра в Рочестерском технологическом институте, специализируясь на компьютерных науках.

Благодаря своей простоте и адаптивности, jQuery стала популярной библиотекой, которая распространилась по всему интернету и в течение нескольких лет доминировала над голым JavaScript. Ресиг опубликовал две версии книги «Pro JavaScript Techniques» в 2006 и 2015 годах. Позже, в 2008 году, он написал ещё одну книгу «Secrets of the JavaScript Ninja», в которой описываются техники JavaScript. Кроме того, он выступал в качестве докладчика и ведущего на многих конференциях по всему миру, акцентируя внимание на инструментах и техниках JavaScript.

После создания jQuery Ресиг перешёл на работу в Khan Academy, некоммерческую образовательную организацию, в 2011 году. Там он занимался разработкой инструментов для обработки JavaScript, что позволило ему оказать влияние на миллионы учеников и разработчиков по всему миру.

jQuery — это компактная и многофункциональная библиотека JavaScript, предназначенная для упрощения работы с HTML-документами, обработки событий, создания анимации и решения других типичных задач веб-разработки. Она позволяет сократить объём кода по сравнению с использованием чистого JavaScript.

Ключевые функции jQuery включают:

Упрощённый обход DOM и манипуляция элементами: jQuery предоставляет удобный синтаксис для поиска и выбора элементов на веб-странице, а также управления ими. С его помощью можно изменять содержимое и атрибуты HTML-элементов, создавать и удалять их.

Обработка событий: jQuery предлагает простой способ прикрепления обработчиков событий к элементам HTML. Вы можете реагировать на действия пользователя, такие как клики, нажатия клавиш и отправка форм, используя лишь несколько строк кода.

Поддержка AJAX: jQuery облегчает процесс отправки AJAX-запросов на веб-сервер. Он предоставляет высокоуровневый API, позволяющий легко отправлять и получать данные в различных форматах, включая JSON и XML.

Эффекты и анимация: jQuery включает в себя мощный фреймворк анимации, облегчающий создание динамичных и привлекательных веб-страниц. Вы можете использовать его для добавления таких эффектов, как затухание, слайды и переходы к элементам HTML.

jQuery широко применяется и имеет активное сообщество разработчиков, предоставляющих плагины и расширения для библиотеки. Он совместим с большинством веб-браузеров и может использоваться совместно с другими веб-технологиями, такими как CSS и HTML5.

Плюсы или преимущества jquery

Плюсы использования jQuery:

Кроссбраузерная совместимость: jQuery работает во всех основных браузерах, что упрощает разработку и тестирование.

Простой в использовании синтаксис: jQuery предлагает упрощённый и понятный синтаксис для выполнения различных задач, что делает код более читаемым и лёгким в поддержке.

Большое сообщество и экосистема плагинов: jQuery имеет активное сообщество разработчиков, которые создают множество плагинов и расширений, облегчающих разработку сложных проектов.

Хорошая производительность: jQuery оптимизирован для быстрой работы, что ускоряет загрузку страниц и улучшает взаимодействие с пользователем.

Минусы использования jQuery:

Кривая обучения: хотя jQuery предлагает упрощённый синтаксис, изучение основ JavaScript всё равно необходимо для эффективного использования библиотеки.

Накладные расходы: включение jQuery в проект может увеличить время загрузки страницы, особенно если используются только некоторые функции библиотеки.

Управление зависимостями: использование множества плагинов и расширений с jQuery может усложнить управление зависимостями и поддержание актуальности версий.

Альтернативные библиотеки: существуют другие библиотеки и фреймворки, которые могут предложить больше возможностей и лучшую поддержку конкретных потребностей проекта.

События в jQuery.

События jQuery — это действия или изменения, происходящие на веб-странице, например, когда пользователь нажимает кнопку или прокручивает страницу вниз. jQuery предоставляет полный перечень событий, которые можно использовать для запуска действий и реакции на странице.

Типы событий jQuery:

События мыши: запускаются при взаимодействии пользователя с мышью, например, при щелчке, наведении курсора или перетаскивании.

События клавиатуры: запускаются при взаимодействии пользователя с клавиатурой, например, при вводе текста или нажатии клавиши.

События формы: запускаются при взаимодействии пользователя с элементами формы, например, при отправке формы или изменении значения поля ввода.

События документа/окна: запускаются при выполнении действий с документом или окном, например, при загрузке или выгрузке страницы.

Пользовательские события: создаются разработчиками и могут использоваться для запуска определённых действий или реакции на странице.

Для использования событий jQuery разработчикам необходимо сначала выбрать элемент или элементы, к которым они хотят присоединить событие. Это можно сделать с помощью селекторов jQuery. После выбора элемента событие может быть присоединено с помощью метода on(), который принимает название события и функцию в качестве аргументов.

<https://abdultechhub.com/blog/pros-and-cons-of-using-jquery-for-front-end-development>  
<https://dzen.ru/a/ZYBbv19pi1rAq7Xl>

**2.1 История и применение AJAX.**

Летом 2001 года американский программист Пол Бакхейт разослал своим друзьям и коллегам веб-сайт, над которым он работал. Сайт представлял собой приложение, позволяющее пользователям просматривать тысячи электронных писем одновременно и получать релевантные результаты.

Бакхейт завершил работу над программным обеспечением за один день, поэтому оно получилось немного грубым. Однако друзья попросили его добавить функцию поиска в электронной почте. Бакхейт учёл пожелания и продолжил работу.

Спустя несколько лет компания Google выпустила Gmail для широкой аудитории, хотя изначально доступ к нему можно было получить только по приглашению, что делало его довольно эксклюзивным. В нём было множество функций, а также 1 ГБ памяти (в отличие от 4 МБ у конкурентов) и эффективный поиск. Кроме того, интерфейс Gmail был быстрым и отзывчивым, а строка поиска автоматически заполнялась по мере ввода. Переписка была организована в развёртываемые и легко читаемые темы. При нажатии на электронное письмо оно загружалось практически мгновенно. Gmail был онлайн, но ощущался как десктопное приложение, потому что именно его и разрабатывал Бакхейт.

В 2004 году компания Google приобрела Where2 Technologies, и основатели Where2 Technologies Ларс и Джен Расмуссен доживали последние дни своего проекта из-за неудачной инвестиционной сделки. Покупка Google произошла как раз вовремя и спасла их. Последние несколько лет Ларс и Джен Расмуссен работали над картографическим приложением с необычным интерфейсом. Приложение было основано на концепции плиток, где каждая плитка представляла собой небольшую часть карты, и объединение их позволяло пользователям перемещаться по карте, просто проводя по ней мышкой.

Приложение было разработано для настольных компьютеров, но Ларри Пейдж, соучредитель Google, был большим поклонником интернета. Он убедил команду, занимавшуюся картографическим проектом, преобразовать своё приложение в веб-сайт. Не зная, как работает Gmail, команда перенесла свою технологию в браузер. Вскоре после этого, в феврале 2005 года, Google Maps перешла в бета-версию.

И Gmail, и Google Maps имеют нечто большее, чем просто общую компанию. Их интерфейсы динамичны, производительны и самые передовые. Они кажутся будущим интернета, и оба они разумно используют JavaScript. Это позволяет им асинхронно подключаться к серверу и получать новые данные без загрузки новой страницы. Они должны благодарить Microsoft за эту функцию.

В то же время в Microsoft происходили интересные события. Начиная с Internet Explorer 3, Microsoft добавила свою собственную реализацию виртуальной машины Java в операционную систему Windows и, в частности, в IE, для запуска апплетов Java. Но почти сразу после этого Sun (создатель Java) подала в суд на Microsoft за неполное соблюдение стандартов Java. Microsoft, уже вовлечённая в антимонопольный иск, решила просто удалить Java из своей операционной системы.

Была только одна проблема. Чтобы заставить работать некоторые из более продвинутых функций их почтового веб-приложения Outlook, они ссылались на некоторые Java-компоненты Internet Explorer. Поэтому, чтобы восполнить этот пробел, прямо перед выпуском Internet Explorer 5 в 1999 году Microsoft добавила в браузер новую функцию JavaScript под названием XMLHttpRequest. Это позволяло браузеру делать HTTP-запрос к серверу и получать обратно некоторые данные в формате XML за кулисами его работы. Это означало, что данные могли быть обновлены на странице без полной перезагрузки. Новая функция довольно хорошо справилась со своей задачей, однако вся её работа заключалась только в том, чтобы Outlook работал так, как предполагалось.

Тем не менее, это не помешало другим браузерам использовать новую функциональность. Mozilla вскоре последовала их примеру, представив собственную реализацию в Firefox. Эта функция использовалась то в одном, то в другом проекте в течение следующих нескольких лет. Самым ярким из них был Oddpost, веб-приложение для работы с электронной почтой. Oddpost содержал множество дизайнерских решений в своём приложении, которые улучшали использование электронной почты (например, окно с тремя панелями). Ещё в 2002 году они довольно широко использовали функцию XMLHttpRequest в своём приложении. Однако сайт был платным и поэтому не привлекал особого внимания, пока его не купила Yahoo в 2005 году.

Эта функция стала основой для Gmail и Google Maps. HTML не мог справиться самостоятельно, и JavaScript помог объединить всё воедино.

Бакхейт из Gmail и братья Расмуссен из Google Maps признавали, что полагаться исключительно на JavaScript было серьёзной ошибкой. В то время реализация в разных браузерах была нестабильной, и если возникали ошибки в JavaScript, это могло привести к сбоям в работе браузера.

Несмотря на это, они успешно справились с задачей и привлекли внимание пользователей. Возможность просматривать тысячи электронных писем или прокручивать карту без задержек сделала сервисы уникальными. Отношение мира к интернету изменилось, и эти сервисы стали популярными.

Позднее Джесси Джеймс Гаретт из Adaptive Path дал название этой функции в JavaScript. Он назвал её «Asynchronous Javascript and XML», что позже сократилось до Ajax. Ajax был не отдельным методом или технологией, а набором принципов, основанных на работе, проведённой в Google, которая описывала, как обрабатывать JavaScript в сложных веб-приложениях. Суть технологии заключается в использовании XMLHttpRequest для отправки запроса на сервер, получения XML-данных и размещения их на странице с помощью JavaScript и семантического HTML.

После этого Ajax стал популярным. Сейчас он лежит в основе современных веб-технологий и породил множество фреймворков и методологий. Со временем технология была усовершенствована: XML заменили на JSON, а XMLHttpRequest на fetch. Ajax показал разработчикам, что интернет — это не только документы. С его помощью можно создавать приложения. Сейчас это может показаться незначительным, но тогда это было очень важно.

AJAX — это не отдельная технология, а концепция, основанная на сочетании нескольких смежных технологий. AJAX использует два ключевых принципа:

Динамическое обращение к серверу без полной перезагрузки страницы, например, с помощью XMLHttpRequest (основного объекта).

Динамическое создание дочерних фреймов, тегов <script> и <img>, как это реализовано в Google Analytics.

Для динамического изменения содержимого страницы используются DHTML и каскадные таблицы стилей CSS. Действия с интерфейсом преобразуются в операции с элементами DOM, которые обрабатывают доступные пользователю данные и изменяют представление. Обработка перемещений, щелчков мыши и нажатий клавиш также осуществляется с помощью этих технологий. Объект XMLHttpRequest или аналогичные механизмы обеспечивают асинхронное взаимодействие с сервером и загрузку необходимых данных во время работы приложения.

## **Сравнение стандартного подхода и AJAX**

В классической модели веб-приложения:

1. Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент;
2. [Браузер](https://ru.ruwiki.ru/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) формирует и отправляет [запрос](https://ru.ruwiki.ru/wiki/HTTP) [серверу](https://ru.ruwiki.ru/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80);
3. В ответ сервер генерирует совершенно новую веб-страницу и отправляет её браузеру и т. д., после чего браузер полностью перезагружает всю страницу.

При использовании AJAX:

1. Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент;
2. [Скрипт](https://ru.ruwiki.ru/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82) (на языке [JavaScript](https://ru.ruwiki.ru/wiki/JavaScript)) определяет, какая информация необходима для обновления страницы;
3. [Браузер](https://ru.ruwiki.ru/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) отправляет соответствующий запрос на [сервер](https://ru.ruwiki.ru/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5));
4. Сервер возвращает только ту часть документа, на которую пришёл запрос;
5. Скрипт вносит изменения с учётом полученной информации (без полной перезагрузки страницы).

Преимущества

Экономия трафика, уменьшение нагрузки на сервер

При правильной реализации AJAX позволяет снизить нагрузку на сервер в несколько раз. В частности, все страницы сайта чаще всего генерируются по одному шаблону, включая неизменные элементы («шапка», «навигационная панель», «подвал» и т. д.), для генерации которых требуются обращения к разным файлам, время на обработку скриптов (а иногда и запросы к БД) — всё это можно опустить, если заменить полную загрузку страницы генерацией и передачей лишь содержательной части.

Ускорение реакции интерфейса

Поскольку загрузка изменившейся части значительно быстрее, то пользователь видит результат своих действий быстрее и без мерцания страницы (возникающего при полной перезагрузке).

Возможности для интерактивной обработки

Например, при вводе поискового запроса в Google выводится подсказка с возможными вариантами запроса. На многих сайтах при регистрации пользователь вводит имя и сразу же видит, доступно это имя или нет. AJAX удобен для программирования чатов, административных панелей и других инструментов, которые выводят меняющиеся со временем данные.

Мультимедиа не останавливается

. Страница не перезагружается, плеер продолжает работать. Потому AJAX ценен на аудио- и видеохостингах. Так, YouTube на 2021 г. позволяет играть видео в режиме «картинка в картинке».

Недостатки

Отсутствие интеграции со стандартными инструментами браузера

URL не меняется, потому не работает кнопка «Назад», невозможно сохранить закладку. Проблему можно успешно решить с помощью History.pushState[5].

Динамически загружаемое содержимое недоступно поисковикам (если не проверять запрос, обычный он или XMLHttpRequest)

Поисковые машины не могут выполнять JavaScript, поэтому разработчики должны позаботиться об альтернативных способах доступа к содержимому сайта.

Старые методы учёта статистики сайтов становятся неактуальными

Многие сервисы статистики ведут учёт просмотров новых страниц сайта. Для сайтов, страницы которых широко используют AJAX, такая статистика теряет актуальность.

Усложнение проекта

Прераспределяется логика обработки данных — происходит выделение и частичный перенос на сторону клиента процессов первичного форматирования данных. Это усложняет контроль целостности форматов и типов. Конечный эффект технологии может быть нивелирован необоснованным ростом затрат на кодирование и управление проектом, а также риском снижения доступности сервиса для конечных пользователей.

Требуется включённый JavaScript в браузере

Он может быть выключен из соображений безопасности. Кроме того, AJAX-страницы труднодоступны неполнофункциональным браузерам, роботам и веб-архивам.

Проблемы с отображением нестандартных кодировок в некоторых сценариях ajax-скриптов

Проблемам AJAX и кириллицы посвящено множество обсуждений в Интернете.

Низкая скорость при грубом программировании

Несмотря на то, что AJAX изначально предназначен именно для повышения скорости, он может сыграть злую шутку: когда AJAX-запросов на одной странице много и, например, по каждому щелчку подгружается список, AJAX-страница становится даже медленнее традиционной.

Плохое поведение на ненадёжных соединениях

Если связь часто теряется (из-за потери несущей или перегрузки канала), обычную страницу можно как минимум перезагрузить. AJAX-страницу (например, с «бесконечной» прокруткой) пользователю приходится перезагружать с самого начала и искать, где он остановился. Параллельная работа — особенность AJAX — здесь сказывается негативно, деля и без того узкий канал на множество маленьких соединений, и велика вероятность, что какое-то из них будет разорвано. Проблема частично решается API истории.

Риск фабрикации запросов другими сайтами

Результат работы AJAX-запроса может являться JavaScript-кодом (в частности, JSON). XMLHttpRequest действует только в пределах одного домена, а тег <script> — нет. Если написать

<script type="text/javascript" src="http://example.org/inbox.php"></script>

то в HTML будет вставлена папка входящих сообщений того пользователя, который на этом компьютере авторизован на example.org. Для защиты используют POST-запрос. Но GET считается идемпотентным и потому кэшируется, POST — нет, поэтому Google вставляет в начало ответа бесконечный цикл: AJAX может делать с ответом что угодно, в том числе убрать цикл, а тэг <script> подключает скрипт как есть и зацикливается.

**2.2 История и применение фреймворка «Jquery-UI»**

jQuery UI — это бесплатное и открытое программное обеспечение, включающее коллекцию виджетов, интерфейсов и эффектов, созданных на базе библиотеки jQuery. Это набор инструментов для веб-разработчиков, желающих добавить интерактивность и стиль в свои веб-приложения.

История jQuery UI началась в 2006 году, когда Джошуа Дж. Д. Чанг предложил создать набор инструментов на основе популярной библиотеки jQuery. Первый релиз jQuery UI был представлен в декабре 2007 года, и с тех пор разработка и поддержка продолжаются усилиями множества разработчиков по всему миру. Основные версии jQuery UI были выпущены в 2011, 2013 и 2020 годах.

jQuery UI состоит из нескольких ключевых компонентов и инструментов:

Эффекты: jQuery UI предлагает множество эффектов, таких как анимация, плавное появление и исчезновение элементов, а также эффекты для изменения размеров и перемещений.

Виджеты: jQuery UI включает набор виджетов для создания различных элементов интерфейса, таких как кнопки, выпадающие списки, ползунки, календари и многое другое.

Диалоговые окна: jQuery UI предлагает набор диалоговых окон, включая модальные окна, окна сообщений, окна предупреждений и окна ошибок.

Стили: jQuery UI позволяет разработчикам применять различные стили к виджетам и интерфейсам, используя CSS-файлы или встроенные стили.

События: jQuery UI обеспечивает обработку событий, таких как нажатия кнопок, перемещения мыши и нажатия клавиш на клавиатуре.

jQuery UI находит применение в создании веб-сайтов, мобильных приложений, игр и других интерактивных веб-приложений. Вот несколько примеров использования jQuery UI:

Создание веб-сайта с галереей изображений с возможностью просмотра изображений и слайд-шоу.

Создание мобильного приложения для бронирования столиков в ресторане с выбором даты и времени.

Разработка игры на HTML5 с использованием jQuery UI для анимации и обработки событий.

Создание системы управления контентом (CMS) с использованием виджетов и диалоговых окон jQuery UI.

В итоге jQuery UI является важным инструментом для создания интерактивных и стильных веб-приложений благодаря своей гибкости, простоте использования и широкому набору инструментов.

**2.3 Плагин « Owl Carousel»**

Slider картинок на веб-сайте товара – это эффективный инструмент для привлечения внимания посетителей и улучшения пользовательского опыта. Позволяя пользователям пролистывать изображения товара, слайдеры помогают им более полно и детально рассмотреть продукт со всех сторон, в разных ракурсах и освещениях.

Основное преимущество slider картинок – это возможность показать все преимущества товара и его внешний вид в полной мере. Благодаря этому, покупатель может лучше представить, как будет выглядеть товар в реальности и принять обоснованное решение о покупке. Кроме того, slider помогает украсить страницу товара, сделая ее более привлекательной и информативной.

Другое важное преимущество slider картинок – это возможность увеличить конверсию и увеличить средний чек. Просматривая различные изображения товара, пользователь может лучше его оценить и быть более склонен к совершению покупки. Кроме того, slider обеспечивает удобство и легкость взаимодействия с сайтом, что способствует улучшению пользовательского опыта.

Таким образом, применение slider картинок у товара на сайте – это современный способ оформления страницы товара, который помогает улучшить пользовательский опыт, увеличить конверсию и сделать покупки более информативными и привлекательными для потенциальных покупателей.

Owl Carousel - это библиотека для создания интерактивных каруселей на веб-сайтах. Эта библиотека основана на языке программирования JavaScript и позволяет легко добавлять анимированные слайд-шоу на страницу.

Одной из основных особенностей Owl Carousel является его простота в использовании. Для того чтобы добавить карусель на сайт, нужно всего лишь подключить библиотеку к вашему проекту и использовать несколько строк кода, чтобы настроить внешний вид и поведение карусели.

Кроме того, Owl Carousel предоставил широкий спектр возможностей для настройки карусели под мои потребности.Я смог удобно быстро настроить количество отображаемых элементов, скорость анимации, автоматическое переключение слайдов, а также добавить различные эффекты перехода, чтобы сделать карусель еще более привлекательной для пользователей.

Благодаря своей гибкости и простоте в использовании, Owl Carousel стал одним из наиболее популярных инструментов для создания каруселей на веб-сайтах. Он используется многими веб-разработчиками по всему миру для добавления интерактивных и привлекательных слайд-шоу на их сайты.

Таким образом, это простой и удобный способ добавить красивую карусель на сайт, мне очень понравилось работать с этой библиотекой.

**2.4 Шаблон проектирования MVC**

В последние годы архитектурный шаблон MVC (Model-View-Controller) становится всё более популярным в разработке. В 2009 году он стал стандартом и, скорее всего, будет сохранять свою актуальность ещё долгое время.

Давайте исследуем, что разработчики вкладывали в этот подход в 1970–1980-х годах и как он используется сейчас.

MVC — это архитектурный шаблон проектирования, разработанный в 1970-х годах в Xerox PARC для создания пользовательских интерфейсов на основе Smalltalk. Впоследствии этот шаблон был адаптирован и применён в различных областях, включая веб-разработку.

Шаблон Model-View-Controller описан в статье «Smalltalk-80: The Language and its Implementation» авторов Алана Кея Тейлора и Дэниела Гееворта Триггерса, опубликованной в 1981 году в журнале «Byte Magazine». В этой работе авторы представляют архитектурный шаблон MVC, использованный при разработке системы программирования на языке Smalltalk-80.

Основные идеи, привнесённые этой работой:

Использование объектов как основных элементов языка и программирования.

Применение MVC для создания пользовательских интерфейсов, разделяющих данные, представление и управление.

Принцип «Everything is an object», согласно которому все элементы в языке Smalltalk-80 являются объектами.

Поддержка множественного наследования и динамической типизации.

Использование исключений в обработке ошибок.

Поддержка сборки мусора для управления памятью.

Разработчики столкнулись с проблемой управления пользовательским интерфейсом в Smalltalk-80, когда при изменении данных требовалось также изменять и пользовательский интерфейс, что было сложным и многословным процессом.

Архитектура MVC решает эту проблему, разделяя приложение на три компонента: модель, представление и контроллер. Модель содержит данные и бизнес-логику приложения, представление отображает эти данные пользователю, а контроллер управляет взаимодействием между моделью и представлением.

Сегодня MVC используется в различных технологиях и фреймворках для создания веб-приложений, таких как ASP.NET MVC, Ruby on Rails, Django и других. Эта технология позволяет отделить логику приложения от пользовательского интерфейса и упрощает разработку масштабируемых и поддерживаемых приложений.

Не менее популярным стал этот шаблон и в мире PHP. Так, востребованный фреймворк CodeIgniter предоставлял поддержку архитектуры MVC. CodeIgniter позволял разработчикам создавать приложения, разделяя их на три компонента: модель, представление и контроллер. Не отстает от него и фреймворк Zend Framework 1, который также поддерживал архитектуру MVC. Кого не перечисли — CakePHP, Symfony и Yii — везде используется MVC.

**2.5 Фрэймворк CodeIgniter.**

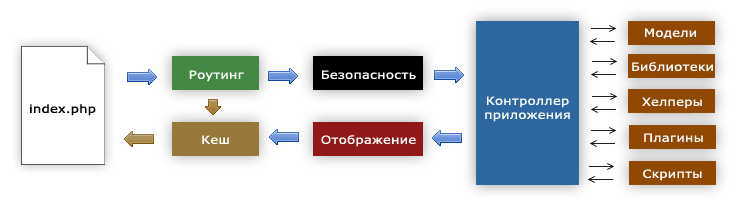
CodeIgniter — популярный PHP фреймворк с открытым исходным кодом, позволяющий создавать многофункциональные и безопасные веб-приложения с MVC архитектурой. Разработан компанией EllisLab.  
Зачем делать всё с нуля и  каждый раз переписывать по 1000 строк кода? Зачем тратить много времени на понятие собственного кода? Вам повезло если в вашем проекте мало строк кода и он хорошо документирован, но если у вас более 1000 файлов и в придачу код безобразно раскидан, то маленькое исправление может занять кучу времени. Выход есть! Рано или поздно вам придётся использовать фреймворк.

#### **Так что же такое фреймворк**.

**Фреймворк** (каркас, структура) — программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. В отличие от библиотек, которые объединяют набор подпрограмм близкой функциональности, фреймворк содержит в себе большое количество разных по назначению библиотек.

#### **Почему именно CodeIgniter**

CodeIgniter —  это фреймворк написанный на PHP. Первый публичный релиз CI состоялся в 2006 году (7 лет назад). Он быстро набирал популярность благодаря своей простоте, и быстроте. На данный момент актуальная версия **2.1.3.**CodeIgniter использует MVC архитектуру, что позволяет всё разложить по полочкам. Ниже вы можете увидеть процесс работы CI. Имеет поддержу библиотек, хелперов, хуков. Также имеется встроенная система кэширования.



#### **Плюсы CodeIgniter**

##### Отличная и понятная документация Действительно у CodeIgniter очень богатая и понятная документация, и её приятно  читать. Всё пояснено примерами кода, который можно просто скопировать и использовать у себя. Также документация уже давно переведена на русский язык, так что вам не составит большого труда, чтобы изучить CodeIgniter.

##### **Не требует большое количество ресурсов**

##### CodeIgniter может работать почти на всех хостингах с поддержкой PHP не ниже 5.1 версии. Благодаря своей грамотной структуре CI не нагружает систему и очень шустро работает. Сейчас я не буду сравнить скорость работы с другими фреймворками, так как холивары ни к чему.

##### **Очень прост в использовании** Разработка приложений  на CodeIgniter имеет  другой стиль, чем просто писать на  чистом PHP. Вам покажется, что надо надо многое учить, чтобы разрабатывать на фреймворке, но я обрадую вас, CI очень легко и быстро учится. На CodeIgniter можно разрабатывать лёгкие и средние приложения, но для более сложных проектов лучше выбрать другой фреймворк.

##### **Скорость генерации страниц.**

CodeIgniter очень быстр в работе.  И это действительно так, а если вы не верите, то можете проверить сами.

##### **Поддержка большого количества баз данных**

При помощи своих встроенных драйверов CodeIgniter может работать с разными базами данных, таким как: MySQL, PostgreSQL, MSSQL, SQLite, Oracle. Также имеется PDO драйвер, что крайне удобно. В CI реализован  шаблон проектирования ActiveRecord для работы с базами данных.

##### **Большое количество стандартных библиотек и классов** Чтобы вы не захотели бы сделать, CodeIgniter найдёт решение на большинство ваших идей. Перечислять все библиотеки, хелперы и классы я не буду, так как их очень много. Но перечислю которыми пользуюсь я: pagination, captcha\_helper, form validation, xmlrpc, email, url\_helper, security\_helper.

##### **Множество ресурсов по CI.**

##### В интернете существует огромное количество сайтов, посвящённых CodeIgniter. Так что найти библиотеки сторонних разработчиков или найти ответ на свой вопрос по CI не составит большого труда. Ниже в статье будут ссылки на различные ресурсы.

#### **Минусы CI**

##### **Слабая система кэширования**

Система кэширования в CodeIgniter работает лишь с целыми страницами, но с частями страницы, и это не очень удобно. А это значит, что вам придётся писать свою систему кэширования или прикручивать уже готовые варианты. Хотя при такой скорости, кэширование и не нужно.

##### **Плохой стиль написания**

CodeIgniter позволяет полностью отказаться от модели, что не есть хорошо. CI приучает с расхлябанному стилю программирования. Также не очень развито наследование классов, местами использует процедурный код.

**2.6 Выбор среды разработки**

**2.6.1 MAMP**

MAMP (Mac OS, Apache, MySQL, PHP) - это платформа для разработки веб-приложений, которая позволяет запускать сервер Apache, базу данных MySQL и язык программирования PHP на компьютере Mac. MAMP - это, по сути, локальный сервер, который позволяет вам разрабатывать и тестировать веб-сайты на своем компьютере, прежде чем вы разместите их на удаленном сервере. Это отличный инструмент для веб-разработчиков и дизайнеров, которые хотят создавать и тестировать свои проекты локально, без необходимости подключения к Интернету.

Основными плюсами MAMP являются удобство использования и легкость установки. Просто скачайте установочный файл с официального сайта, следуйте инструкциям по установке и в течение нескольких минут у вас будет работающий локальный сервер. MAMP включает в себя все необходимое для разработки веб-приложений, включая Apache сервер, MySQL базу данных и PHP язык программирования. Благодаря этому вы можете начать работать над своим проектом сразу после установки, не тратя время на настройку каждого компонента отдельно.

Еще одним плюсом MAMP является возможность быстро и легко создавать несколько раздельных сред для разработки. Вы можете создать несколько виртуальных хостов для различных проектов, установить разные версии PHP или MySQL для тестирования совместимости, и все это легко управляется через простой и интуитивно понятный пользовательский интерфейс.

Кроме того, MAMP обеспечивает удобство отладки ваших веб-приложений. Вы можете легко просматривать логи ошибок, настраивать параметры сервера, тестировать скрипты и многое другое, все это делается намного проще и удобнее, чем на удаленном сервере.

Однако у MAMP есть и минусы. Это производительность. Так как все компоненты запускаются на вашем компьютере, это может привести к замедлению работы вашей системы, особенно если у вас не самый мощный компьютер. Также, локальный сервер не будет иметь таких же настроек и параметров, как удаленный сервер, что может привести к непредвиденным проблемам при развертывании вашего проекта.

В целом, MAMP - отличный инструмент для тестирования и разработки веб-приложений на компьютере Он обеспечивает легкость использования, удобство настройки и эффективный процесс отладки. Однако, перед использованием MAMP важно учитывать его ограничения и недостатки, чтобы избежать проблем при развертывании проекта на удаленном сервере.

**2.6.2 Visual Studio Code**

Visual Studio Code был анонсирован 29 апреля 2015 года на конференции Build и выпущен в бета-версии 18 ноября 2015 года. Исходный код был опубликован на GitHub.

**Плюсы Visual Studio Code:**

Подсветка синтаксиса для множества языков программирования.

Автоматическое завершение кода.

Встроенный отладчик.

Поддержка системы контроля версий Git.

Обширная библиотека расширений.

Возможность создавать пользовательские расширения.

Встроенный терминал.

Анализ ошибок и подсказки по исправлению.

Возможность создавать и запускать задачи для автоматизации рутинных операций.

Возможность менять темы оформления, шрифты и другие параметры интерфейса.

**Минусы Visual Studio Code:**

Не подходит для разработки сложных проектов, требующих мощных инструментов.

Может быть менее производительным при работе с большими проектами.

Некоторые функции могут быть ограничены в бесплатной версии.

**2.7 Создание собственного сайта**

Темой для создания эргономичного интерфейса был выбран интернет магазин. Исходя и выше проведённого исследования и стандартов вёрстки у нас есть обязательные и желательные элементы для сайта.

К обязательный структурным блока относится: header,main (container), footer.В header у нас будет находится navbar. Навигационное меню в которому у нас будет:Логотип сайта , который на главной странице не является ссылка на главную страницу, а на других страницах являлся ссылкой на главную страницу сайта. Дальше будет форма в которой будет строка и кнопка для поиска товара. Эта форма будет вести на страницу , где будут отображаться результаты поиска. В правой части сайта будут ссылки на «whishlistt», корзину,профиль и кнопку входа.

Wishlist(перевод: «Список желаемого»).Это страница с товарами которые пользователь переместил в список, чтобы сохранить товары, которые он хотел бы купить.

Кнопка «Профиль» будет вести на страницу пользователя, где будет написана о пользователе.

Кнопка «корзина» будет вести на страницу корзины,где будут показаны выбранные для покупки товары.

В блоке main у нас будет находится основной контент страницы, высота страницы автоматический будет подстраиваться под содержимое. Ширина всех страниц фиксированная.

В блоке footer у нас будет коротка информация об разработчиках сайта: контактная почта и т.д.

Желательным элементом будет боковое меню для быстрого поиска по основным категориям.

Содержание контейнера будет зависеть от страницы.

На страницах всех онлайн магазинов есть блоки со скидками, акциями и другими предложениями. Даже на станицах с просмотром информации об конкретном товаре есть блок с предложениями. Это нужно будет учитывать.

На главной странице будет несколько блоков с предложениями.

На странице с отдельным товаром будет информация о товаре:

Обложка товара, карусель других фотографий, название товара,год выхода,описание и набор тегов для этого товара. Кнопки добавления товара в корзину или список желаемого.

На странице поиска будет блок с фильтрами, где можно выбрать нужные параметры и произвести поиск по ним.

На сайте обязательно будет система входа и регистрации ,ведь к ней привязана система добавления в корзину и список желаемого.

**Глава 2 Практическая часть**

**Главная страница сайта**

Приложения:

Источники:  
1. <https://abdultechhub.com/blog/pros-and-cons-of-using-jquery-for-front-end-development>  
2. <https://dzen.ru/a/ZYBbv19pi1rAq7Xl>

3. <https://guides.hexlet.io/ru/bootstrap/>

4. https://www.mo4tech.com/a-brief-history-of-css.html

5. https://timeweb.com/ru/community/articles/obzor-yazyka-php-proshloe-nastoyashchee-perspektivy

6. https://codecloudcafe.com/the-evolution-of-html-from-html-to-html5/

7. :https://timeweb.com/ru/community/articles/obzor-yazyka-php-proshloe-nastoyashchee-perspektivy

8. :https://servergate.ru/articles/mysql-preimushchestva-i-nedostatki/.

9. :https://inmotion.live/blog/notes/mysql-story/.

[10. https://nuancesprog.ru/p/17293/](https://nuancesprog.ru/p/17293/)