МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет прикладной математики, информатики и механики

Кафедра математического и прикладного анализа

**РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ ПЕРЕГОВОРНЫМИ КОМНАТАМИ**

Бакалаврская работа

Направление 01.03.02 Прикладная математика и информатика

Профиль Математические и компьютерные методы решения задач естествознания

Зав. Кафедрой\_\_\_\_\_\_ д.ф.-м.н., проф. А.И. Шашкин \_\_.\_\_.20\_г.

Обучающийся\_\_\_\_\_\_ Р.С. Бакланов

Руководитель \_\_\_\_\_\_ к.ф-м.н., М.И. Быкова

Воронеж 2022

# ВВЕДЕНИЕ

Последние несколько лет весь мир сталкивался с различными последствиями пандемии, которая оказывает огромное влияние на все отрасли экономики. Область информационных технологий не стала исключением. Влияние, оказанное пандемией, привело к изменениям и новым проблемам, в сфере ИТ, которые требуют своевременного решения.

В следствии повсеместного введения удалённого формата работы, многие компании начали нанимать специалистов из разных городов и стран, но также не приостановили найм в годах присутствия. Зачастую компании исходили из расчета, что подобный формат работы не подразумевает расширения офисного пространства и создания новых физических рабочих мест, переговорных и т.д., но с затуханием пандемии и возрастающем желанием части сотрудников вернуться в офис или переехать в город присутствия компании, появилась проблема организации оптимального использования офисного пространства.

Актуальность проблемы возрастает, когда очередная волна пандемии затухает и увеличившееся количество сотрудников компании начинает посещать офис, а компания не стала вкладывать средства в расширение рабочего пространства, или наоборот, просторный офис компании пустует в следствии начала новой волны пандемии и предприятие вынужденно оплачивать его аренду и обслуживание. В такой ситуации компании терпят издержки и использование офисного пространства не может быть оптимальным.

Проблемы возникают не только с рабочими местами, но с другими помещениями офиса. Например, с переговорными, в которых проводятся деловые встречи и видеоконференции. Не каждый сотрудник может организовать себе время и место для проведения деловых встреч у себя дома, особенно когда часовые пояса коллег сильно отличаются. Также в деловой среде очные встречи играют важную роль, а зачастую необходимы для подписания новых соглашений, контрактов, обсуждения требований к разработке ПО. Переговорные могут использоваться и командами разработки для анализа проделанной работы и обсуждения планов. В такой ситуации требуется оптимизировать процесс использования офисной инфраструктуры, чтобы сотрудники имели возможность эффективно работать, а компания развиваться и расти, без лишних трат.

# ПОСТАНОВКА ЗАДАЧИ

Для решения проблем организации использования офисного пространства было принято решение разработать веб-приложение, которое будет поддерживать возможность бронирования переговорных и закрепления за сотрудниками рабочих мест. Подобное решение применяется для контроля за использованием парковочных мест у офисов компаний, так как не все сотрудники используют собственное транспортное средство для поездок в офис, а оплачивать места на каждого сотрудника нецелесообразно.

Основная задача приложения — давать возможность пользователям авторизовываться с помощью своих учетных данных, показывать расписание броней для каждой переговорной, информацию о количестве мест, а также оборудование, которое присутствует в помещении, предоставлять возможность забронировать определённое время и добавить участников встречи.

В качестве дополнительного функционала планируется реализация роли администратора, в возможности которой входит: создание новых переговорных комнат и занесение данных о них в базу, создание, удаление и редактирование пользовательских данных, изменение ролей пользователей. Также планируется создание раздела с встречами, в которых участвует пользователь.

# АНАЛИЗ ЗАДАЧИ

Для решения задачи можно использовать технологию построения одностраничных веб-приложений (Single Page Application). Ключевой особенностью данного подхода является загрузка всего приложения на одну HTML-страницу. Благодаря динамическому обновлению с помощью языка программирования JavaScript, во время использования не нужно перезагружать или загружать дополнительные страницы. В процессе работы пользователь будет испытывать схожий опыт взаимодействия, как с полноценным настольным приложением, а не веб-сайтом, так как оно мгновенно реагирует на все его действия, без задержек. И в тоже время сохраняется, возможность запуска приложения на любом типе устройства, так как для его запуска нужен только браузер.

В качестве инструмента для реализации решения с использованием концепции одностраничного приложения будет использован современный веб-фреймворк Angular, использующий язык TypeScript, который поддерживает анализирование типов и безопасности кода. Одной из возможностей Angular является создание прогрессивных веб-приложений (Progressive Web Application). PWA является гибридным решением и позволяет открывать приложения с помощью браузера. При этом полностью сохраняется функциональность полноценного настольного или мобильного приложения. На текущий момент времени, существуют ограничения наложенные иностранными компаниями на распространение и обновление мобильных приложений на смартфонах, использование технологии прогрессивных веб приложений позволяет обойти запреты, так как такое приложение может быть установлено прямо с сайта, доступность которого может быть обеспечена, путем его размещения на собственном серверном оборудовании.

В ходе работы использовались программные продукты: среда для веб разработки WebStorm; платформа облачной базы данных Firebase; распределенная система контроля версиями программного кода Git.

# СРЕДСТВА РЕАЛИЗАЦИИ

## Framework веб разработки Angular

Angular представляет фреймворк от компании Google для создания клиентских приложений. Прежде всего он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений. В этом плане Angular является наследником другого фреймворка AngularJS. В то же время Angular это не новая версия AngularJS, а совершенно новый фреймворк.

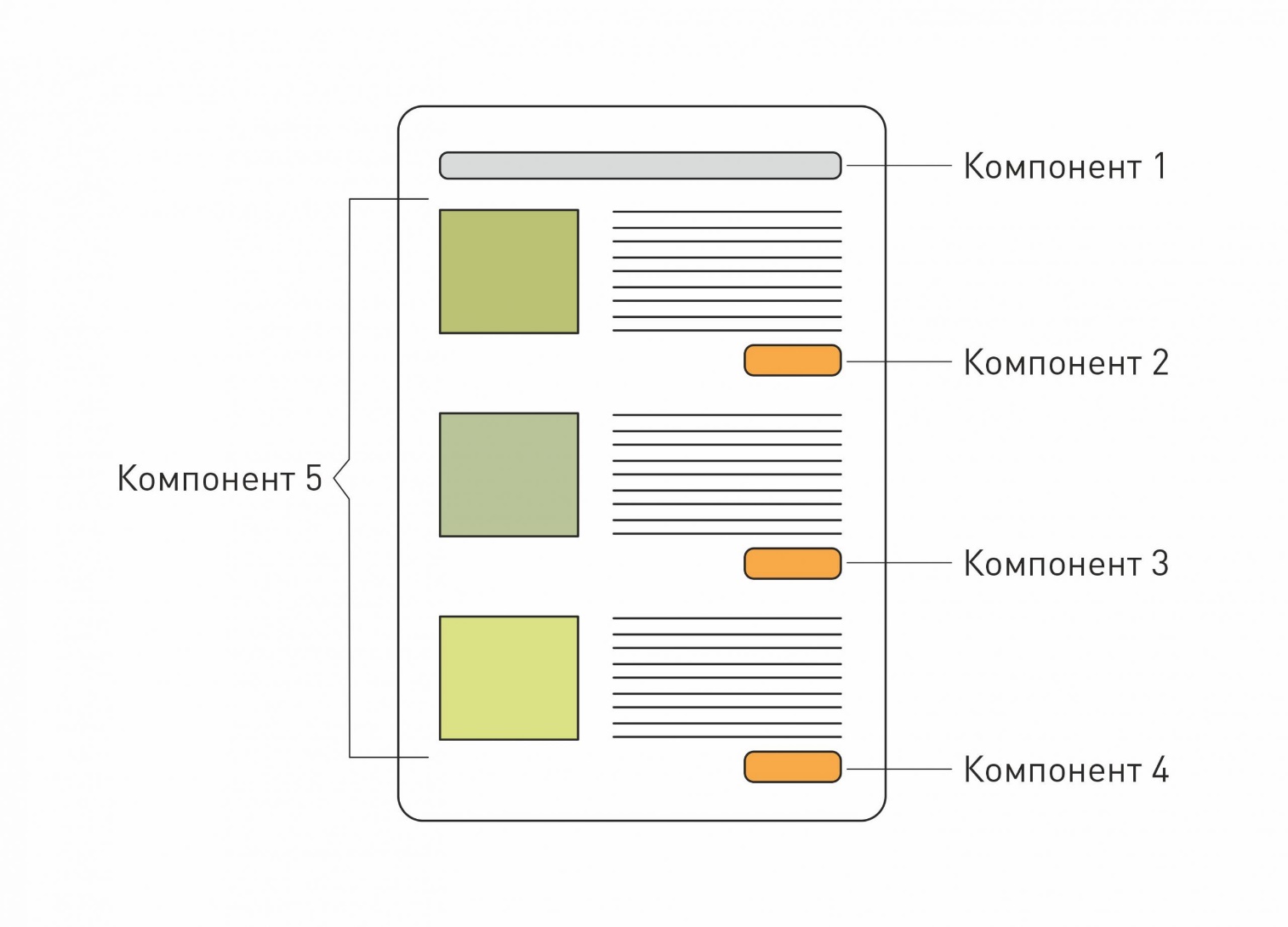
Angular предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация.

Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript. Но мы не ограничены языком TypeScript. При желании можем писать приложения на Angular с помощью таких языков как Dart или JavaScript. Однако TypeScript все-таки является основным языком для Angular.

Компоненты представляют основные строительные блоки приложения Angular. Каждое приложение Angular имеет как минимум один компонент. Для создания компонента необходимо импортировать функцию декоратора @Component из библиотеки @angular/core. Декоратор @Component позволяет идентифицировать класс как компонент. Если бы мы не применили декоратор @Component к классу AppComponent, то класс AppComponent компонентом бы не считался.

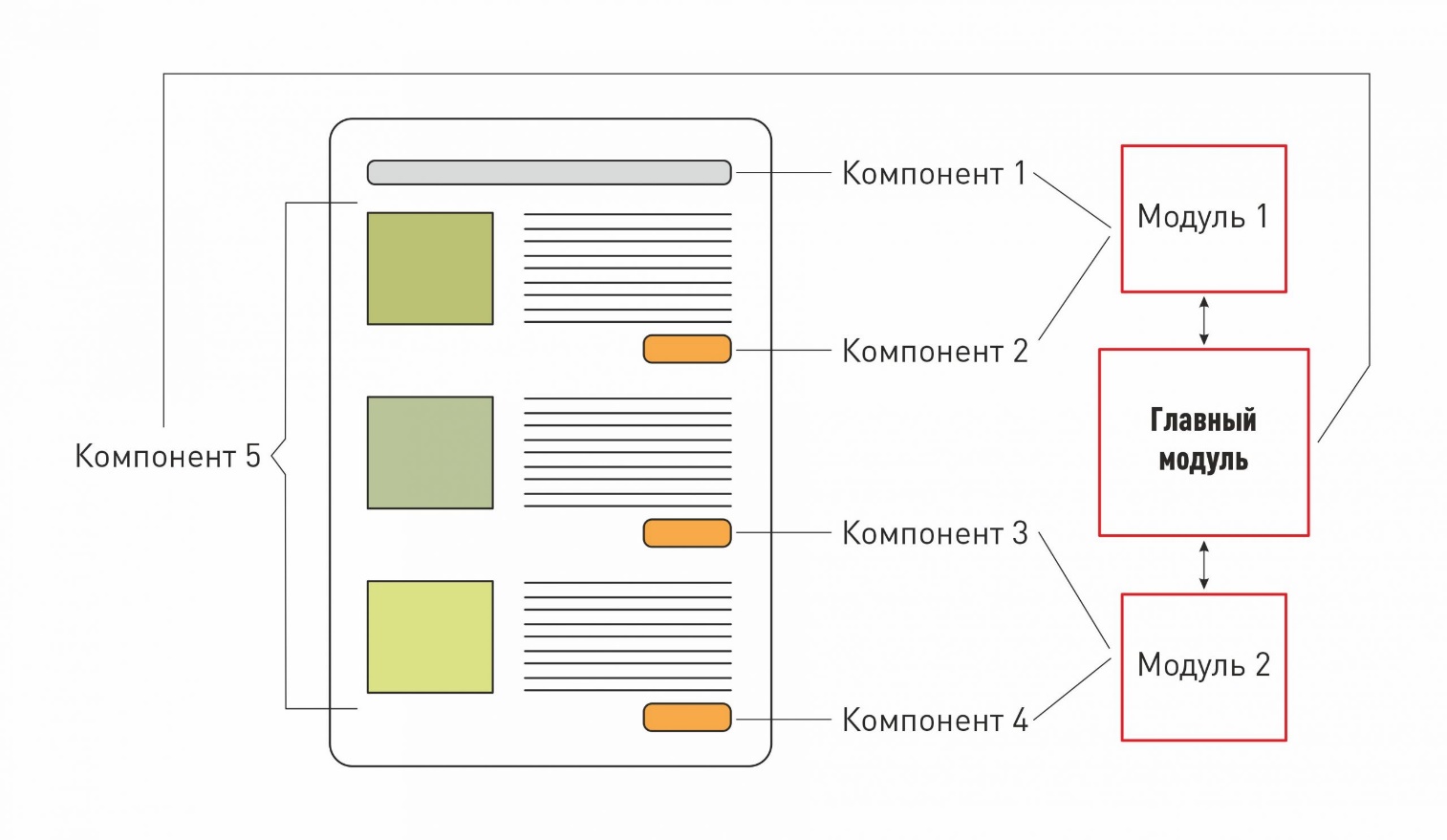
Декоратор @Component в качестве параметра принимает объект с конфигурацией, которая указывает фреймворку, как работать с компонентом и его представлением.

С помощью свойства template. Шаблон представляет кусок разметки HTML с вкраплениями кода Angular. Фактически шаблон это и есть представление, которое увидит пользователь при работе с приложением.

  
(Рис. 1. – структура шаблонов в Angular)

Каждый компонент должен иметь один шаблон. Однако необязательно определять шаблон напрямую с помощью свойства template. Можно вынести шаблон во внешний файл с разметкой html, а для его подключения использовать свойство templateUrl.

Все компоненты подключаются к главному или дополнительным модулям. Модули управляют компонентами. Главный модуль контролирует всё приложение, а дополнительные модули следят за работой отдельных элементов. Дополнительные модули выполняют узкую задачу, подчиняются основному модулю и нужны для снятия нагрузки с основного. Они отвечают за навигацию, анимацию, хранилище компонентов, взаимосвязь с браузером.

  
(Рис. 2. – взаимодействие модулей приложения)

Для сложных операций вместо компонентов используют сервисы. Они отвечают только за тот набор логических операций, для которых он предназначен. Например, мы можем подключить сервис «Таблицы». Это поможет отслеживать страницы с большим количеством данных, которые после обновлений меняют содержимое сайта. Никаких других действий сервис «Таблицы» выполнять не будет.

## Язык программирования TypeScript

Язык программирования TypeScript представляет собой расширение уже существующего языка программирования JavaScript. Разработчиком языка является корпорация Microsoft, которая представила его в 2012 году. В создании TypeScript принимал участие разработчик Андерс Хейлсберг, создавший ранее Turbo Pascal, Delphi и C#. Язык ввёл новые концепции, позволяющие выражать идеи более кратко и точно, и легко создавать безопасные и масштабируемые современные приложения.

**Компилятор**

Главной особенностью TypeScript является его компилятор, результатом работы которого является код на JavaScript, что позволяет сохранить обратную совместимость и использовать библиотеки написанные на JavaScript в TypeScript коде. В проверке типов заключается основное отличие TypeScript. С её помощью можно убедиться, что программа, написанная вами, работает, так как вы ожидаете. В общем виде процесс преобразование и выполнение кода TypeScript выглядит так (рис. 3).

(Рис. 3. – процесс выполнения кода TypeScript)

Шаги 1–3 выполняются компилятором TypeScript, а шаги 4–6 средой выполнения JavaScript, находящейся в браузере пользователя, или на сервере. По сути, компилятор переводит код на TypeScript в обычный JavaScript, но в полученном коде уже нет специфичных конструкций и типов данных, такой процесс часто называют транспиляцией.

**Система типов**

Системой типов является набор правил, которые применяются к написанному вами коду на этапе его компиляции. Системы типов можно разделить на два вида:

1. С явным указанием типов;
2. С автоматическим выводом типов.

TypeScript заимствует лучшее из этих двух видов, вы можете явно указать типы, либо позволить компиляторы самостоятельно сделать их определение за вас. Для явного вывода типов используются аннотации, которые указывают какой тип имеет значение.

let a: number = 2 // переменная a является number

let b: string = ‘string value’ // переменная b является string

let c: boolean = [true, false] // переменная с является boolean массивом

Язык TypeScript использует механизм постепенной типизации — он работает лучше, если все типы в программе были указаны заранее. В случае не типизированного кода, компилятор сможет осуществить проверку типов и обнаружить ошибки. Конечно, есть множество ошибок, которые TypeScript не сможет определить. К ним относятся разрывы сетевых соединений, переполнение стека и некорректный ввод данных пользователем. Данные ошибки будут приводить к исключениям при работе программы. Основная задача TypeScript — уберечь разработчика от несоответствия типов и обнаружить такие ошибки на этапе компиляции. Сравнение систем типов JavaScript и TypeScript показано в таблице 1.

|  |  |  |
| --- | --- | --- |
| Система типов | JavaScript | TypeScript |
| Как связываются типы | Динамически | Статически |
| Конвертируются ли типы автоматически | Да | Нет (в основном) |
| Когда проверяются типы | Во время выполнения | Во время компиляции |
| Когда выводятся ошибки | Во время выполнения | Во время компиляции |

(таблица 1. Сравнение систем типов JavaScript и TypeScript)

## Программный продукт для написания веб-приложений WebStorm

Программный продукт WebStorm от JetBrains является инструментов для веб-разработки и поддерживает большое количество возможностей по работе с программным кодом. В WebStorm есть все необходимое для работы с JavaScript, HTML, CSS и TypeScript, а также есть поддержка отдельных библиотек.

Продукт имеет встроенную поддержку системы контроля версий, которая позволяется просматривать варианты кода, внесённые изменения, автора изменений. Благодаря ведению истории в WebStorm можно восстанавливать удалённые куски кода, а также файлы.

Технология LiveEdit позволяет редактировать код на HTML, CSS, JavaScript и видеть, как результат отображается в окне браузера. Для использования этой возможности потребуется установка расширения для браузера.

WebStorm поддерживает языки автодополнение кода, дополняя написанное разработчиком. Все подсказки кода формируются с использованием машинного обучения, их выдача происходит с анализом и проверкой контекста типов. Данная технология позволяет получать подсказки CSS классов при написании кода внутри файлов JavaScript или TypeScript. Для улучшения качества кода, среда разработки проводит анализ кода и выявляет в нем ошибки и осуществляет проверку правописания и следования правилам оформления функций, классов и переменных. Проанализировав программный код WebStorm, может осуществить безопасную переработку части кода.

Как большинство сред разработки JetBrains WebStorm поддерживает функцию отладки исполняемого кода. Данная функция доступна для разных типов инструментов и технологий включая JavaScript и TypeScript. В процессе выполнения программы можно расставлять точки останова и исполнять программу пошагово.

## Технологии верстки: HTML и CSS

**Язык разметки HMTL**

Язык гипертекстовой разметки HTML является разработкой британского учёного Тима Бернерса-Ли в 1986 – 1991 годах в Женеве в Швейцарии. Изначально HTML создавался для обмена научной и технической документацией, между людьми не обладающими специальными знаниями в области верстки пользовательских интерфейсов. Получение HTML документов происходит с использованием протоколов HTTP/HTTPS, или можно открыть документ из локальной файловой системы в браузере. После открытия код, содержащийся в документе, интерпретируется и отображается на экране, в соответствии с стилями форматирования.

Основными строительными блоками HTML документа являются элементы или «теги». С помощью HTML разные конструкции, контент, веб-формы и другие объекты могут быть встроены и отформатированы на отображаемой странице. Средствами HTML можно создавать заголовки, абзацы, ссылки, списки и т.д. Элементы HTML обозначаются угловыми скобками.

HTML документы поддерживают возможность для интеграции кода JavaScript, для этого используется специальный элемент <script>. JavaScript код позволяет динамически манипулировать элементами на странице и менять её содержимое. Для расширения возможностей по стилизации и позиционированию содержимого документа, используется CSS.

В начале своего существования язык HTML был разработан, чтобы облегчить структурирование и форматирование документов без привязки к средствам их отображения. Вид документов должен был быть одинаковым на разных типах дисплеев. Однако сегодня применение HTML далеко от его изначальной задачи. Часто элементы HTML используются не по прямому назначению. Например, блок <table> предназначен для создания таблиц, долгое время использовался для позиционирования элементов на странице.

**Представление HTML документа**

В работе языка HTML используются правила описанные в файле объявления типа документа (Document Type Definition). Данный документ определяет какие элементы, атрибуты и их значения действительны для конкретного типа HTML.

DOCTYPE позволяет корректно отобразить документ в браузере, он определяет версию HTML, и файл объявления типа документа.

<!DOCTYPE html> <!-- Объявление формата документа -->

<html>

<head> <!-- Техническая информация о документе -->

<meta charset="UTF-8"> <!-- Определяем кодировку символов документа -->

<title>Заголовок</title> <!-- Задаем заголовок документа -->

<link rel="stylesheet" type="text/css" href="style.css"> <!-- Подключаем внешнюю таблицу стилей -->

<script src="script.js"></script> <!-- Подключаем JavaScript -->

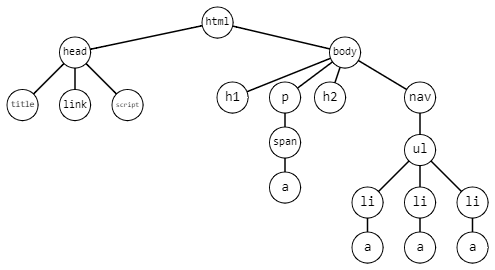
</head>

<body> <!-- Основная часть документа -->

</body>

</html>

Каждый HTML документ при интерпретации браузером образует, объектную модель документа (Document Object Model). Данная модель представляет собой дерево HTML элементов.

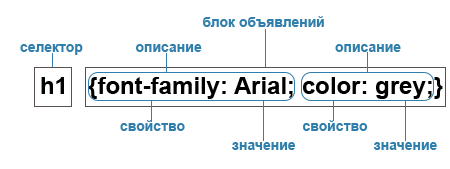
****(Рис. 4 – пример объектной модели документа)

**Язык таблиц стилей CSS**

CSS (Cascading Style Sheets) является одним из основных языков веб-разработки. Он используется для представления внешнего вида документа, описанного с помощью HTML или XML. С помощью CSS можно описывать стили веб-страниц и пользовательских интерфейсов. Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для HTML элементов можно использовать ограниченный набор свойств.

**Объявление стилей документа**

Для объявления стиля используются специальные селекторы. Селектором является специальные выражения, которые интерпретируются браузером и привязываются к HTML элементам. Именно селектор определяет то, к какому элементу будут применены заданные стили.

  
(Рис. 5 – структура объявления селектора CSS)

СSS селекторы подразделяются на следующие типы:

1. Селектор по HTML элементу, предназначен для выбора «тегов» по их имени;
2. Селектор по классу, выбирает элемент по специфичному атрибуту class у элемента;
3. Селектор по идентификатору, предназначен для выбора элемента по атрибуту id;
4. Универсальный селектор (селектор звёздочка) предназначен для выбора всех элементов на странице;
5. Селектор по атрибуту, предназначен для идентификации элемента по имени его атрибутов и значению.

Синтаксис основных селекторов CSS:

/\* селектор div выберет все элементы типа div на странице \*/

div {

margin-bottom: 10px;

}

/\* селектор .right выберет все элементы, имеющие атрибут class со значением right\*/

.right {

right: 10%;

}

/\* селектор #topbar выберет элемент, имеющий идентификатор topbar \*/

#topbar {

height: 120px;

width: 100%;

}

/\* универсальный селектор \* выберет все HTML элементы \*/

\* {

font-size: 16pt;

}

/\* селектор [href] выберет все элементы, которые имеют атрибут href \*/

[href] {

color: #fd231;

}

**Псевдоклассы**

Псевдоклассы, являются дополнение к основным видам селекторов. Они дают возможность выбрать HTML элемент в зависимости от его особого состояния. Псевдоклассы позволяют применять стили к элементам основываясь на внешних факторах, таких как история посещений, позиция курсора мыши, состояния содержимого элемента.

Синтаксис использования псевдокласса:

/\* при наведении курсора мыши на элемент h1, применится псевдокласс hover и его цвет изменится \*/

h1: hover {

background-color: red;

}

**Специфичность CSS**

Специфичность — это алгоритм, который используется браузером для определения, какие именно стили из всех файлов будут применены к конкретному элементу. Для вычисления используются CSS селекторы. В случае если одному и тому же элементу написаны несколько одинаковых стилей с разными селекторами, будут применены те стили селектор, которых обладает наибольшим весом.

Каждый CSS селектор обладает собственным весом. Порядок селекторов по убыванию их веса:

1. Селекторы по идентификатору;
2. Селекторы по классу, селекторы по атрибуту и селекторы с псевдоклассами;
3. Селекторы по тегу, селекторы с псевдоэлементами.

Однако, существуют исключения, комбинаторы +, >, ~, а также универсальный селектор \* и псевдокласс :not(), не учувствуют в расчетах специфичности и имеют вес равный нулю.

**Система расчета специфичности**

Базовое значение специфичности можно представить в виде трех цифр, тогда:

* Селекторы по идентификатору увеличивают первую цифру;
* Селекторы по классу, по атрибуту или псевдокласс увеличивают вторую цифру;
* Селекторы по тегу или псевдоэлемент увеличивают третью цифру.

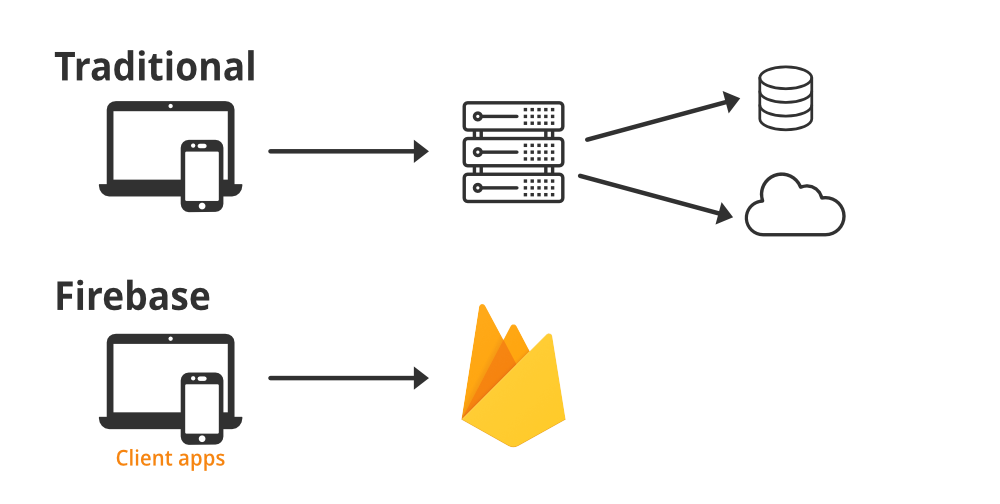
Каждый селектор добавляет единицу к специфичности. В случае комбинации селекторов на один HTML элемент можно рассчитать специфичность. Например, при наличии у элемента селектора по тегу и одного идентификатора, селектор по тегу увеличит последнюю цифру, селектор по идентификатору – первую, тогда вес селектора будет равен 1.0.1.

Существует специальный атрибут style внутри HTML, который имеет максимальную специфичность и обязательно переопределит написанные стили в других файлах своим содержимым.

CSS содержит в своем синтаксисе специальную команду !important, которая способна переопределить отдельные стили, которые были написаны для HTML элемента. В случае применения этого механизма, будут проигнорированы все правила расчета специфичности и применятся свойства стоящие рядом с ключевым словом !important. Применение этой конструкции нежелательно, так как нарушает привычные правила поведения стилей, что осложняет отладку кода.

## 1.5 Облачная платформа Firebase

Firebase — это продукт разработанный компанией Google, который помогает разработчикам легко создавать свои приложения быстрее и безопаснее. В Firebase присутствует облачная база данных, которая позволяет хранить и получать данные, а также фильтровать их с помощью встроенных функций работы с данными. Однако, прямое написание кода на стороне Firebase крайне ограничено, в отличии от классической архитектуры.

  
(Рис. 6 – структура приложений на Firebase в сравнении с классической)



Firebase

Приложение

Приложение

Облачные сервисы

База данных

Сервер

**Краткая история Firebase**

Первоначально Firebase предоставляла специальный сервис для онлайн-чата на различных веб-сайтах и называлась Envolve. Однако, в результате развития сервис начали использовать не только для передачи текстовых сообщений в чате, но и для сохранения состояния приложения и его синхронизации в режиме реального времени. В результате команда разработки решила усовершенствовать архитектуру и сосредоточиться только на управлении состоянием и данными приложений. В текущем виде Firebase была представлена в 2012 году.

**Возможности Firebase**

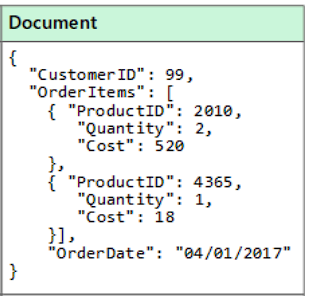
1. База данных реального времени. Представляет собой облачную NoSQL базу, которая хранит данные приложения и осуществляет доступ к ним в режиме реального времени с минимальной задержкой.
2. Облачное хранилище Firestorage, позволяет хранить медиа контент (фотографии, видео, документы). При добавлении файла в Firestorage происходит его оптимизация и сжатие. Использование добавленных файлов возможно по специальной ссылке, которая возвращается после добавления. Данное хранилище доступно между разными устройствами и его можно использовать в нескольких приложениях одновременно.
3. Аутентификация: предоставляет простые в использовании функции аутентификации пользователей в приложении. Поддерживается несколько типов аутентификации: с использованием электронной почти и пароля, с использованием сторонних сервисов (Google, Facebook, Вконтакте и т.д.). Стоит отметить, что механизмы аутентификации разработаны по строгим стандартам безопасности и предотвращают несанкционированный доступ к пользовательским данным, а также обеспечивает хранение данных приложения в зашифрованном виде.
4. Размещение приложения: Firebase обеспечивает быструю и безопасную публикацию приложений. Его можно использовать для размещения статических или динамических веб-сайтов и микросервисов, всего одной командой.
5. Удалённая консоль доступа: служба удалённой настройки помогает быстро публиковать обновления приложения и осуществлять его настройку с любого компьютера, а благодаря поддержке ролей пользователей есть возможность настройки уровня доступа к инфраструктуре проекта.

**Особенности NoSQL баз данных**

Термин NoSQL используется к системам хранения данных, в которых для выполнения манипулирования данными не используется язык запросов SQL. Вместо этого они запрашивают данные с помощью других языков и конструкций.

Хранение данных в не реляционных базах данных происходит в виде наборов значений именованных строковых полей и данных объекта в сущности, которая называется документом. Как правило данные в таких хранилищах хранятся в виде JSON (JavaScript Object Notation). Значение поля может представлять собой скалярный элемент, например число, или сложный объект, вроде списка или вложенного объекта. Поля документов управляются системой, что позволяет пользователям выполнять запросы и применять фильтры, основанные на значениях этих полей.

Большим преимуществом использования NoSQL баз данных, является способ хранения, в случае SQL базы хранение информации происходит в множестве таблиц, тогда как одна запись документно-ориентированной базы данных может содержать в себе множество разнородных данных по конкретной сущности, например пользователю. Для получения документов из базы используется специальный ключ. Ключ – это уникальный идентификатор документа, зачастую к нему применяется хэширование для равномерного распределения данных. Одним из преимуществ NoSQL баз данных является возможность изменения отдельных полей документа, без перезаписи всего документа.

  
(Рис. 7 – пример записи документно-ориентированной базы данных)