**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №1

По «Технологии разработки программного обеспечения»

Студент​ Глубоков Г.В.

Родионова А.П.

Группа АИ-20

Руководитель Назаркин О.А.

Доцент , к. т. н.

Липецк 2023 г.

Задание:

1. Выбор языка программирования
2. Какие парадигмы поддерживает
3. Основное назначение
4. Экосистема
5. Начало работы с языком («Hello world»)
6. Основные фреймворки и библиотеки

**1) Возможности языка**

JavaScript— мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам, также используется для создания серверной части web-приложений.

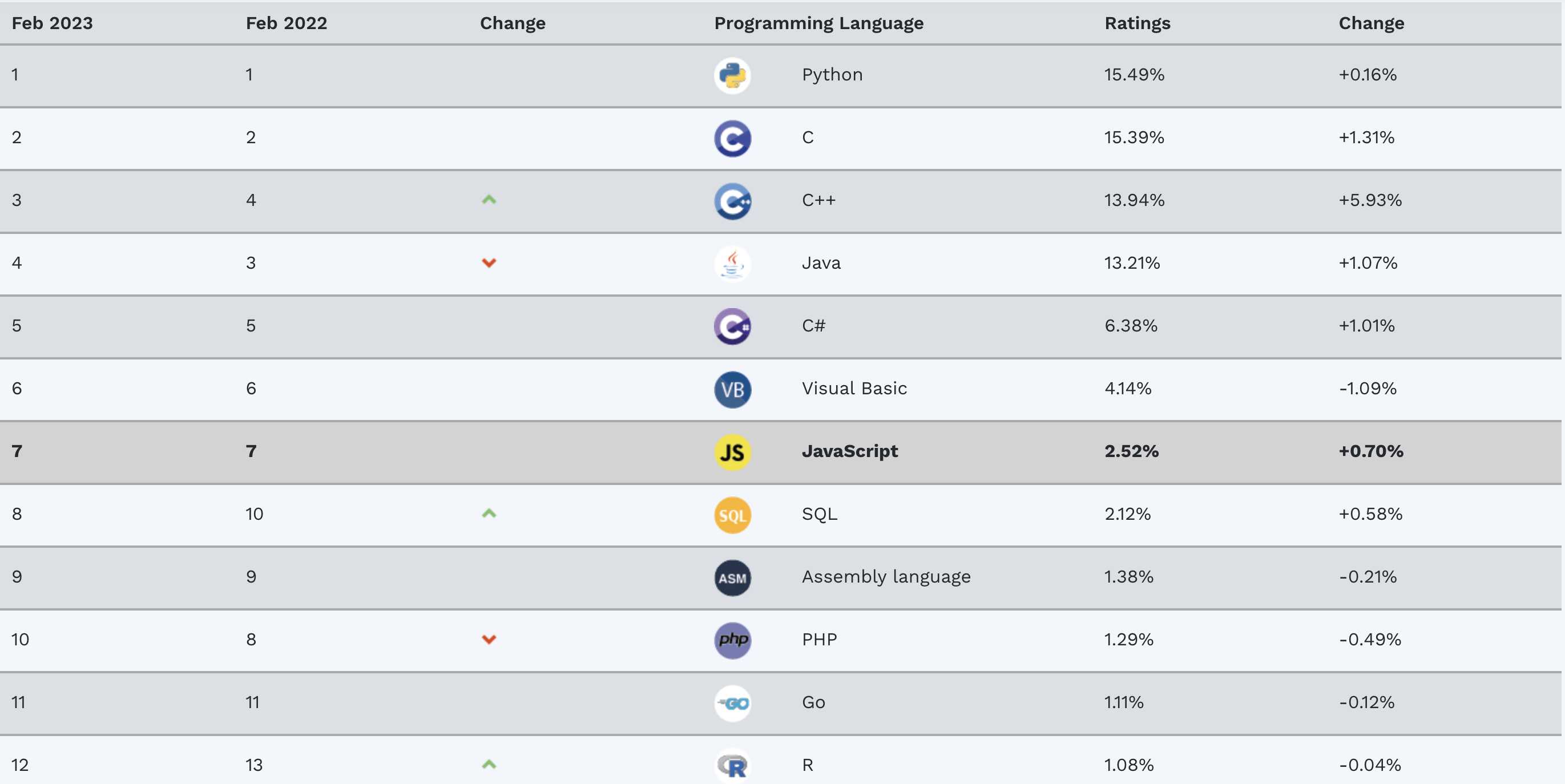


Рисунок 1 – Список популярности языков по версии tiobe.com

## Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

JavaScript обладает рядом свойств объектно-ориентированного языка, но реализованное в языке прототипирование обусловливает отличия в работе с объектами по сравнению с традиционными объектно-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания — что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

* объекты, с возможностью интроспекции;
* функции как объекты первого класса;
* автоматическое приведение типов;
* автоматическая сборка мусора;
* анонимные функции.

В языке отсутствуют такие полезные вещи как:

* модульная система: JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
* стандартная библиотека: в частности, отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода/вывода, базовых типов для бинарных данных;
* стандартные интерфейсы к веб-серверам и базам данных;
* система управления пакетами, которая бы отслеживала зависимости и автоматически устанавливала их.

Наиболее распространенными и практичными парадигмами в javascript являются **функциональное программирование** и **объектно-ориентированное программирование** , которые при освоении дают разработчикам возможность писать надежный код таким образом, который может показаться эффективным для конкретной программы;

### **Структура языка**

Структурно JavaScript можно представить в виде объединения трёх чётко различимых друг от друга частей

* ядро (ECMAScript),
* объектная модель браузера (Browser Object Model или BOM),
* объектная модель документа (Document Object Model или DOM).

Если рассматривать JavaScript в отличных от браузера окружениях, то объектная модель браузера и объектная модель документа могут не поддерживаться

Объектную модель документа иногда рассматривают как отдельную от JavaScript сущность, что согласуется с определением DOM как независимого от языка интерфейса документа. В противоположность этому ряд авторов находят BOM и DOM тесно взаимосвязанными.

#### **Ядро**

ECMAScript не является браузерным языком и на самом деле в нём не определяются методы ввода и вывода информации[28]. Это скорее основа для построения скриптовых языков. Спецификация ECMAScript описывает типы данных, инструкции, ключевые и зарезервированные слова, операторы, объекты, регулярные выражения, не ограничивая авторов производных языков в расширении их новыми составляющими

#### **Объектная модель браузера**

Объектная модель браузера — браузероспецифичная часть языка являющаяся прослойкой между ядром и объектной моделью документа Основное предназначение объектной модели браузера — управление окнами браузера и обеспечение их взаимодействия. Каждое из окон браузера представляется объектом window, центральным объектом BOM. Объектная модель браузера на данный момент не стандартизирована однако спецификация находится в разработке WHATWG и W3C.

Помимо управления окнами, в рамках объектной модели браузера, браузерами обычно обеспечивается поддержка следующих сущностей:

* управление фреймами,
* поддержка задержки в исполнении кода и зацикливания с задержкой,
* системные диалоги,
* управление адресом открытой страницы,
* управление информацией о браузере,
* управление информацией о параметрах монитора,
* ограниченное управление историей просмотра страниц,
* поддержка работы с HTTP cookie.

#### **Объектная модель документа**

Объектная модель документа — интерфейс программирования приложений для HTML и XML-документов. Согласно DOM, документу можно поставить в соответствие дерево объектов, обладающих рядом свойств, которые позволяют производить с ним различные манипуляции:

* получение узлов,
* изменение узлов,
* изменение связей между узлами,
* удаление узлов.

**Экосистема**

**Фреймворки**

**Node.js**

Программная платформа, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения и даже программировать микроконтроллеры. В основе Node.js лежит событийно-ориентированное и асинхронное программирование с неблокирующим вводом/выводом.

Одним из популярных фреймворков для Node.js является Express.js

**Express.js**

Или просто Express, фреймворк web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API. Де-факто является стандартным каркасом для Node.js

**React**

JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов на основе JSX. React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций.

**ReactNative**

Это кроссплатформенный фреймворк с открытым исходным кодом для разработки нативных мобильных и настольных приложений на JavaScript и TypeScript, созданный Facebook, Inc. React Native поддерживает такие платформы как Android, Android TV, iOS, macOS, tvOS, Web, Windows и UWP, позволяя разработчикам использовать возможности библиотеки React вне браузера для создания нативных приложений, имеющих полный доступ к системным API платформ

**Angular**

Открытая и свободная платформа для разработки веб-приложений, написанная на языке TypeScript, разрабатываемая командой из компании Google, а также сообществом разработчиков из различных компаний. Angular - полностью переписанный фреймворк от той же команды, которая написала AngularJS.

**Svelte**

Бесплатный интерфейсный компонент или язык с открытым исходным кодом , созданный Ричем Харрисом и поддерживаемый членами основной команды Svelte. Svelte не является монолитной библиотекой JavaScript , импортируемой приложениями: вместо этого Svelte компилирует HTML-шаблоны в специализированный код, который напрямую манипулирует DOM , что может уменьшить размер передаваемых файлов и повысить производительность клиента Код приложения также обрабатывается компилятором, вставляя вызовы для автоматического пересчета данных и повторного рендеринга элементов пользовательского интерфейса при изменении данных, от которых они зависят. Это также позволяет избежать накладных расходов, связанных с промежуточными представлениями времени выполнения, такими как виртуальный DOM , в отличие от традиционных фреймворков (таких как React и Vue ), которые выполняют большую часть своей работы во время выполнения, то есть в браузере.

**Пакетные менеджеры**

**Npm**

Менеджер пакетов, входящий в состав Node.js. Установка пакета производится при помощи команды: npm install Все доступные для установки пакеты и их краткое описание: npm search. Этой же командой можно производить выборочный поиск

#### **Расположение внутри страницы**

JavaScript можно использовать как и внутри HTML документа, так и отдельным файлом

Для использования внутри HTML необходимо создать файл с расширением «.html» и написать начальную структуру страницы, рис. 1

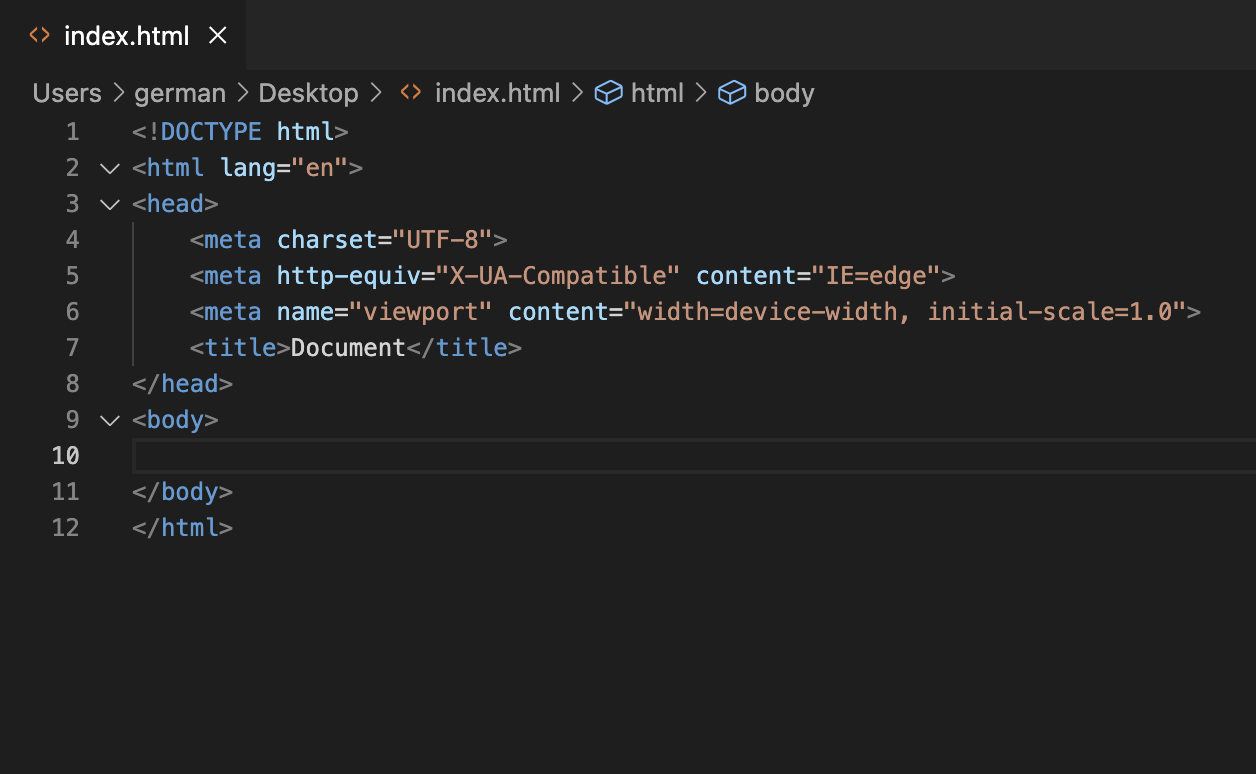


Рисунок 2 – Базовая структура HTML документа

Далее интегрируем JavaScript в наш HTML документ, для этого необходимо написать парный тег <script> внутри тега <body>

Скрипт, выводящий модальное окно с классической надписью «Hello, World!» внутри браузера:

<script>

alert('Hello, World!');

</script>

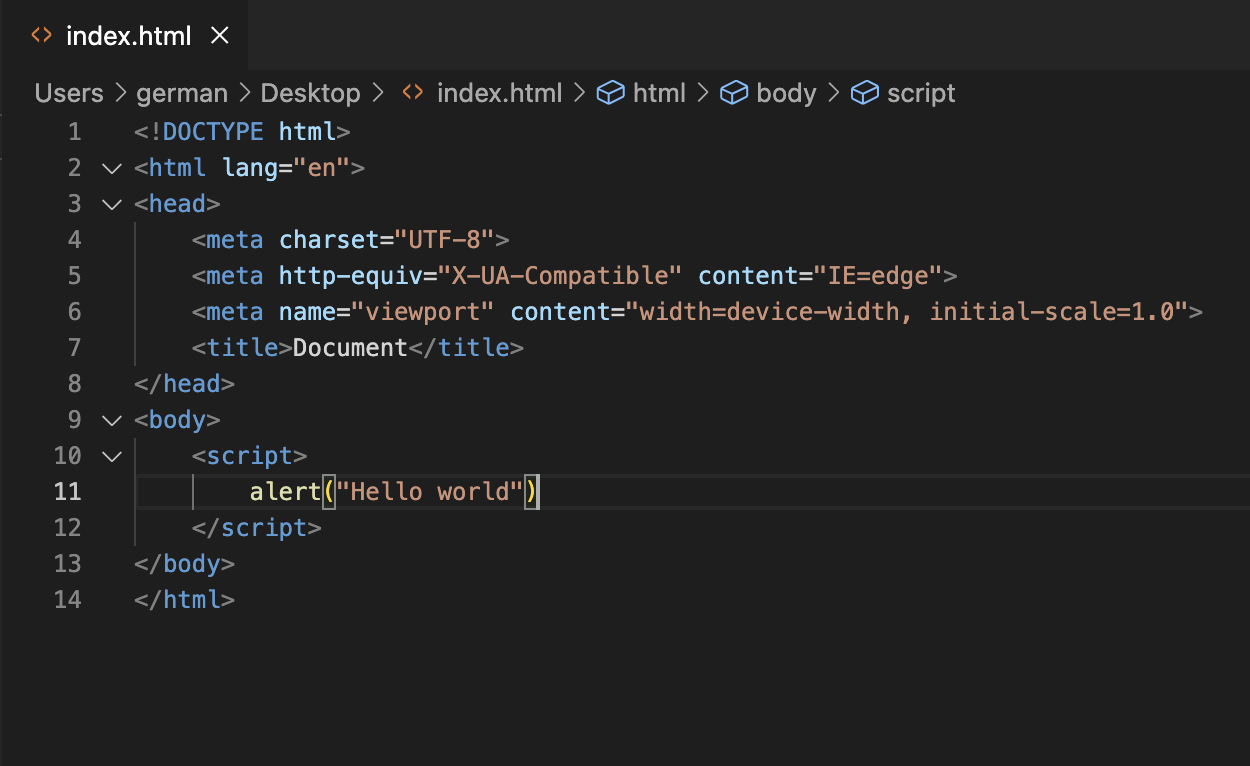


Рисунок 3 – Скрипт “Hello world” на языке JavaScript

Для вывода результата откроем файл через любой браузер, рис. 3

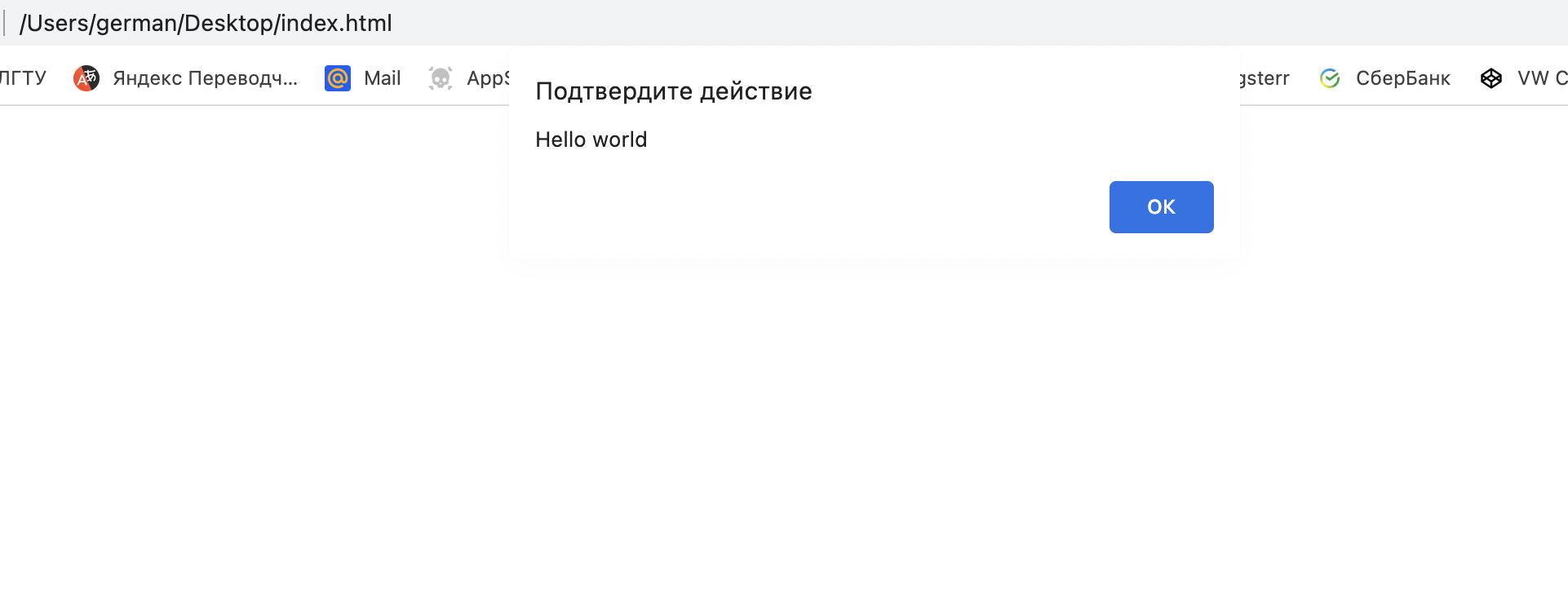


Рисунок 4 – Вывод “Hello world” на языке JavaScript

**Фреймворки**

**Vue.js**

Из большинства front-end фреймворков особую популярность получил фреймворк VUE.

Vue.js — JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле

Для начала работы создадим файл с именем index.html. Далее создадим базовую структуру HTML. Для интеграции используем следующий скрипт:

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

<script>

const app = new Vue({

el: '#app',

data:{

Message:"Hello world!",

}

})

</script>

Для работы Vue нам нужно создать экземпляр Vue и в его параметрах объявить объект с определёнными ключами (опциями), благодаря которым идёт взаимодействие с DOM.

Основные опции:

* data
* method
* computed
* watch
* хуки жизненного цикла

**Data**

Функция которая возвращает объект с данными экземпляра Vue. Vue рекурсивно преобразует его свойства в геттеры/сеттеры, чтобы сделать их «реактивными».

**Methods**

Набор методов работающих также также, как и обычные функции в JavaScript. Может принимать входные параметры.

**Computed**

Computed свойство вызывается каждый раз при изменении каких-либо реактивных свойств, описанных внутри него, и возвращает новое значение.