JavaScript - начало

Знакомство с JS, среда исполнения, node, npm, nvm

Что такое JavaScript

Изначально JavaScript был создан, чтобы «сделать веб-страницы живыми».

Программы на этом языке называются скриптами. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы.

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

Когда JavaScript создавался, у него было другое имя – «LiveScript». Однако, язык Java был очень популярен в то время, и было решено, что позиционирование JavaScript как «младшего брата» Java будет полезно.

Со временем JavaScript стал полностью независимым языком со своей собственной спецификацией, называющейся ECMAScript, и сейчас не имеет никакого отношения к Java.

Особенности JavaScript

- Интерпретируемый язык: JavaScript интерпретируется браузером, что означает, что код выполняется непосредственно во время загрузки веб-страницы.
- Динамическая типизация: Переменные в JavaScript не имеют фиксированных типов данных, и их тип может изменяться во время выполнения программы.
- Прототипное наследование: В JavaScript используется прототипное наследование, в отличие от классического наследования, присущего многим другим языкам программирования.
- Функциональное программирование: JavaScript поддерживает функции как объекты первого класса, что позволяет использовать функции как данные.
- Асинхронное программирование: JavaScript поддерживает асинхронное выполнение кода с помощью колбэков, промисов и асинхронных функций.
- Манипуляция DOM: JavaScript позволяет изменять содержимое и структуру веб-страницы с помощью DOM (Document Object Model), что делает его мощным инструментом для создания интерактивных пользовательских интерфейсов.
- Множество библиотек и фреймворков: Существует множество библиотек и фреймворков JavaScript, таких как React, Angular и Vue.js, которые облегчают разработку веб-приложений.

JavaScript в front-end-е

Веб-страницы состоят из HTML (структура), CSS (визуальное оформление) и JavaScript (интерактивность). JavaScript на стороне клиента выполняется в браузере пользователя и отвечает за динамическое изменение содержимого страницы, реакцию на действия пользователя и взаимодействие с сервером без перезагрузки страницы. На фронтенде JavaScript используется для создания интерфейсов, обработки пользовательского ввода, анимаций, валидации данных, общения с сервером посредством AJAX запросов и многого другого.

JavaScript в back-end-е

JavaScript также может быть использован на стороне сервера, благодаря среде выполнения, такой как Node.js. Node.js позволяет запускать JavaScript код на сервере, что делает его полноценной альтернативой другим серверным технологиям, таким как PHP, Python или Ruby. JavaScript на бэкенде используется для обработки запросов от клиентов, взаимодействия с базами данных, создания API, авторизации и аутентификации пользователей, обработки файлов, выполнения вычислений и многих других задач, связанных с серверной логикой приложений.

Запуск JavaScript в браузере

Движок исполнения JavaScript - это программное обеспечение, которое интерпретирует и выполняет код JavaScript. Он является ключевой частью браузера или другой среды выполнения, поддерживающей JavaScript. Вот несколько популярных движков исполнения JavaScript:

- V8 (Google Chrome, Node.js): Разработанный Google, V8 является одним из наиболее быстрых и мощных движков JavaScript. Он используется в браузере Google Chrome и среде выполнения серверного JavaScript Node.js.
- SpiderMonkey (Mozilla Firefox): SpiderMonkey - это движок JavaScript, разработанный Mozilla Foundation. Он используется в браузере Mozilla Firefox.
- JavaScriptCore (Safari): JavaScriptCore, также известный как Nitro, является движком JavaScript, разработанным Apple. Он используется в браузере Safari и других продуктах Apple.
- Chakra (Microsoft Edge, Internet Explorer): Chakra - это движок JavaScript, разработанный Microsoft. Он используется в браузерах Microsoft Edge и Internet Explorer.

Есть несколько способов подключения JavaScript к HTML:

- Встроенный (Inline)
- Внутренний (Internal)
- Внешний (External)
- Асинхронная загрузка (Async)
- Отложенная загрузка (Defer)

Inline & Internal

Встроенный (Inline):

В этом случае JavaScript-код напрямую встраивается в тег <script> в HTML-документе.

Внутренний (Internal):

JavaScript-код размещается в теге <script> внутри секции <head> или <body> HTML-документа.

```
<script>
  console.log("Привет, мир!");
</script>
```

External

Внешний (External):

JavaScript-код выносится в отдельный файл с расширением ".js", а затем подключается к HTML-документу с помощью тега <script> с атрибутом src.

html

```
<script src="script.js"></script>
```

JavaScript (в файле "script.js"):

javascript

```
console.log("Привет, мир!");
```

Async & Defer

Асинхронная загрузка (Async):

Атрибут async позволяет браузеру загружать и выполнять скрипты асинхронно, не блокируя загрузку остального содержимого страницы.

```
<script src="script.js" async></script>
```

Отложенная загрузка (Defer):

Атрибут defer указывает браузеру, что скрипт должен быть загружен, но выполнение должно быть отложено до тех пор, пока весь HTML-документ не будет полностью разобран.

```
<script src="script.js" defer></script>
```

Запуск JavaScript в терминале

Для запуска JavaScript-а вне браузере требуется движок который понимает Javascript

- node.js
- deno
- bun.js

Node.js

Node.js - это среда выполнения JavaScript, построенная на движке V8 от Google Chrome, которая позволяет запускать JavaScript на сервере. Вот несколько ключевых особенностей Node.js:

- Асинхронное и событийно-ориентированное программирование: Node.js использует неблокирующий ввод/вывод и асинхронные операции, что делает его эффективным для обработки множества одновременных запросов и событий.
- Модульность: Node.js основан на модульной архитектуре, которая позволяет разработчикам использовать множество встроенных модулей и устанавливать сторонние модули с помощью менеджера пакетов npm.
- Высокая производительность: Благодаря движку V8 Node.js обеспечивает высокую производительность выполнения JavaScript-кода.
- Широкое применение: Node.js используется для создания веб-серверов, API, приложений реального времени, микросервисов, средств автоматизации и многого другого.

Deno

Deno - это среда выполнения JavaScript, разработанная одним из создателей Node.js, Райаном Далем. Она представляет собой альтернативу Node.js, но с некоторыми ключевыми отличиями и усовершенствованиями. Вот некоторые из особенностей и характеристик Deno:

- Улучшенная безопасность: Deno предоставляет улучшенную модель безопасности по сравнению с Node.js. По умолчанию Deno не предоставляет доступ к файловой системе, сетевым ресурсам и другим чувствительным операциям. Доступ к таким операциям нужно явно разрешать.
- Модульность и стандартизация: В Deno модули загружаются из Интернета по URL, а не через локальный файловый путь, как в Node.js. Кроме того, в Deno используется стандартный набор модулей (Standard Library), что способствует единообразию в разработке.
- Встроенная поддержка TypeScript: TypeScript - это язык программирования, который представляет собой надмножество JavaScript с добавлением статической типизации. Deno поддерживает TypeScript из коробки, что упрощает разработку и обеспечивает более надежный код.
- Неблокирующая асинхронная операции по умолчанию: Deno ставит в центр неблокирующий ввод/вывод и асинхронное программирование, что делает его подходящим для работы с множеством одновременных запросов и событий.
- Встроенные инструменты: Deno включает в себя несколько встроенных инструментов, таких как инструменты для тестирования, форматирования кода и документации.
- Отсутствие пакетного менеджера: В отличие от Node.js, Deno не имеет пакетного менеджера типа npm. Вместо этого, Deno использует импорт модулей непосредственно из Интернета по URL.

bun.js

Bun.js - это фреймворк для разработки веб-приложений на JavaScript, основанный на идее "frontend-as-a-service". Он предоставляет средства для создания полноценных веб-приложений, включая клиентскую и серверную части, а также инструменты для разработки, тестирования и развертывания.

- Универсальные приложения: Bun.js позволяет создавать универсальные веб-приложения, которые могут работать как на стороне клиента (в браузере), так и на стороне сервера.
- Компонентный подход: Bun.js поддерживает компонентную архитектуру, что позволяет разбивать приложение на небольшие и переиспользуемые компоненты для упрощения разработки и обеспечения модульности.
- Реактивное программирование: Фреймворк предлагает реактивное программирование как способ управления состоянием приложения и реагирования на изменения данных.
- Инструменты разработки: Bun.js включает в себя набор инструментов для разработки, включая сборку и управление зависимостями.
- Средства тестирования и отладки: Фреймворк предоставляет средства для тестирования и отладки приложений, что упрощает процесс разработки и обеспечивает надежность приложения.

NMP & NVM

NPM - пакетный менеджер

NVM - версионный менеджер

На данный момент NPM (Node Package Manager) является одним из наиболее популярных и широко используемых менеджеров пакетов для JavaScript, особенно в контексте среды выполнения Node.js. Однако существуют некоторые альтернативы, которые предлагают свои собственные подходы к управлению пакетами и зависимостями JavaScript. Вот несколько из них:

- Yarn
- pnpm
- npm5+

npm (Node Package Manager):

Что это: npm - это менеджер пакетов для языка JavaScript, который используется в среде Node.js. Он предоставляет доступ к более чем миллиону пакетов JavaScript, которые могут быть установлены в проекты для добавления функциональности.

Зачем: npm позволяет управлять зависимостями проекта, устанавливать и обновлять пакеты, а также публиковать свои собственные пакеты для общего доступа.

Какие цели выполняет: npm упрощает процесс разработки, делая доступными множество готовых решений для расширения функциональности приложения. Он также помогает в поддержании актуальности зависимостей и управлении версиями пакетов.

Примеры работы:

Установка пакета: npm install <package-name>

Обновление пакета: npm update <package-name>

Публикация пакета: npm publish

nvm (Node Version Manager):

Что это: nvm - это инструмент для управления версиями Node.js на вашей системе. Он позволяет устанавливать несколько версий Node.js на одной машине и переключаться между ними по необходимости.

Зачем: nvm полезен, когда вам нужно работать с несколькими проектами, использующими разные версии Node.js, или когда вам нужно протестировать приложение на разных версиях Node.js.

Какие цели выполняет: nvm облегчает управление версиями Node.js и изолирует различные проекты друг от друга, обеспечивая надежную и консистентную разработку.

Примеры работы:

Установка версии Node.js: nvm install <version>

Переключение на другую версию Node.js: nvm use <version>

Просмотр установленных версий Node.js: nvm ls

Node.js

Вот как мы можем использовать Node.js для запуска JavaScript-кода в терминале:

Установка Node.js: Сначала нужно установить Node.js на вашу операционную систему. Вы можете загрузить установщик с официального сайта Node.js и следовать инструкциям по установке.

Создание JavaScript-файла: Создайте файл с расширением ".js", в котором будет находиться ваш JavaScript-код. Например, "hello.js".

Написание кода: Откройте созданный файл в текстовом редакторе и напишите в нем свой JavaScript-код

Запуск кода в терминале: Откройте терминал и перейдите в каталог, где находится ваш файл JavaScript. Затем выполните следующую команду: *node имя_файла.js*

Ресурсы

node js - ТЫК (сразу с NPM)

NVM - ТЫК (скачивает связку npm + node нужной версии)

Введение в JavaScript - ТЫК

Лучший ресурс для изучения JS - ТЫК