Лекция 14
Основы Node.js (продолжение)

14.1 Запуск файлов JavaScript

```
// hello.js
console.log('Hello, app.js!');
```

При выполнении JavaScript-файла с помощью Node.js вы указываете Node.js прочитать файл, интерпретировать его содержимое и выполнить любой написанный в нем код. Процесс начинается, когда вы вводите node команду, а затем имя файла в терминале.

```
node app.js
```

Node.js загружает указанный файл, анализирует JavaScript-код и выполняет его построчно в среде Node. В приведенном выше примере Node.js находит оператор console.log и выводит сообщение в ваш терминал. Как только весь код в файле завершит выполнение, Node.js завершает процесс.

Узнайте больше.

Как Node.js обрабатывает ошибки при выполнении скриптов?

Node.js обрабатывает ошибки во время выполнения скрипта, выводя сообщения об ошибках в консоль. Если в вашем коде возникает ошибка, Node.js описывает тип ошибки, сообщение и место её возникновения в файле. Подробнее об обработке ошибок в Node.js можно узнать в официальной документации

14.2 Использование REPL Node.js

REPL в Node.js — это мощный инструмент, позволяющий интерактивно запускать код JavaScript прямо из терминала. **REPL** расшифровывается как Read-Eval-Print Loop (цикл «чтение - вычисление - вывод»), и он предоставляет среду, в которой вы можете быстро тестировать фрагменты кода, экспериментировать с возможностями JavaScript или отлаживать простую логику без создания файлов. Когда вы запускаете REPL, набрав команду node в командной строке (без указания файла), вы попадаете в приглашение, где можете вводить и выполнять операторы JavaScript по одному.

REPL особенно полезен для:

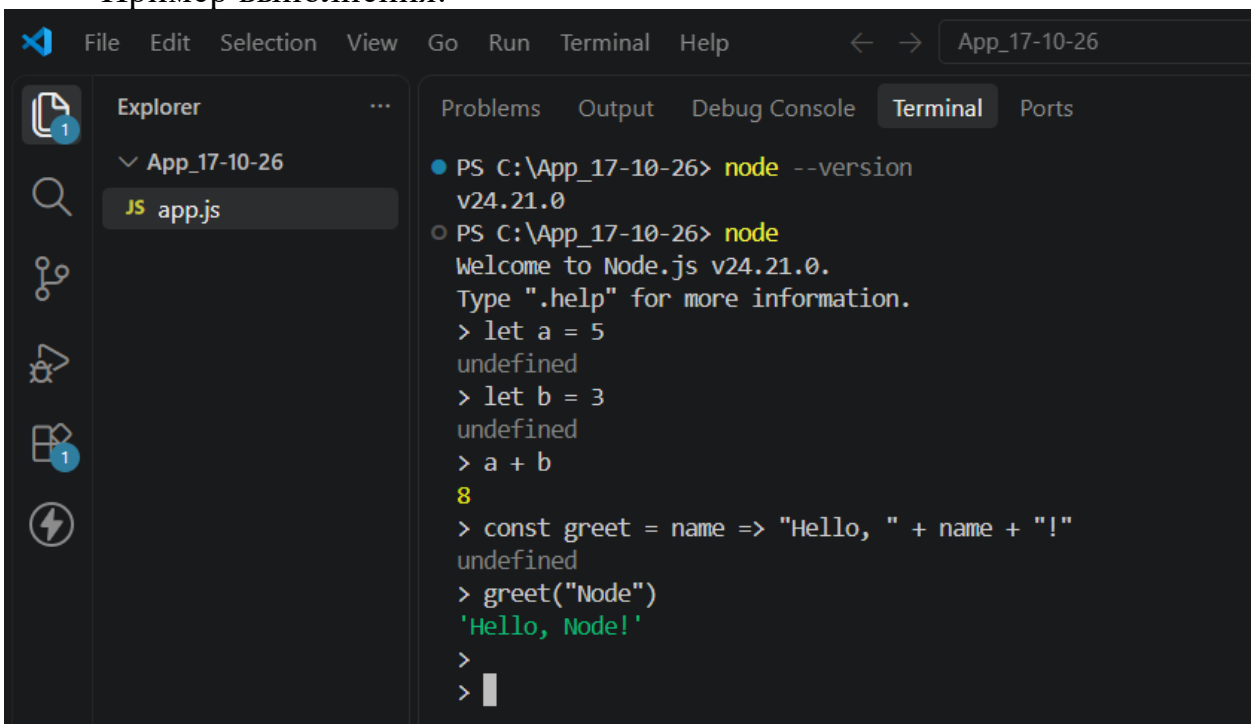
- Пробую различные выражения и операторы JavaScript;
- Тестирование функций или логики перед их добавлением в более крупную программу;
- Изучение глобальных объектов и API Node.js;
- Отладка небольших фрагментов кода в отрыве от контекста.

К основным командам REPL относятся ввод выражений, использование переменных и вызов функций. Выйти из REPL можно в

любой момент, Ctrl+C дважды нажав клавишу Enter или набрав команду `.exit. Дополнительные команды REPL начинаются с точки, например, .help для отображения доступных команд.

Определение. REPL расшифровывается как **цикл чтения-выполнения-печати** . Это описывает интерактивный цикл, в котором Node.js считывает ваши входные данные, выполняет код, выводит результат и ожидает дальнейших входных данных.

Пример выполнения:



Откройте терминал: в VS Code: **Terminal → New Terminal**;

```
> let a = 5
undefined
> let b = 3
undefined
> a + b
8
> const greet = name => "Hello, " + name + "!"
undefined
> greet("Node")
'Hello, Node!'
```

Определение. REPL расшифровывается как **цикл чтения-выполнения-печати** . Это описывает интерактивный цикл, в котором Node.js считывает ваши входные данные, выполняет код, выводит результат и ожидает дальнейших входных данных.

14.3 Работа с процессом

При запуске программ на Node.js вы получаете доступ к мощному глобальному объекту, называемому process. Этот объект предоставляет подробную информацию и контроль над текущим процессом Node.js. Объект process доступен повсюду в вашем коде без необходимости подключения или импорта чего-либо.

Два наиболее часто используемых свойства process: process.argv, process.env.

- process.argv - Это массив, содержащий аргументы командной строки, передаваемые при запуске вашей программы Node.js. Первый элемент — это путь к исполняемому файлу Node.js, второй — путь к вашему скрипту, а остальные — любые дополнительные аргументы, которые вы предоставляете;
- process.env - Это объект, который хранит переменные окружения, доступные вашему процессу. Вы можете использовать его для чтения значений конфигурации, секретов или другой информации, заданной вне вашего кода.

Понимание того, как использовать эти свойства, позволяет сделать ваши программы более гибкими и настраиваемыми.

index.js

```
// Accessing command-line arguments and environment variables

console.log('Command-line arguments:', process.argv);
console.log('Environment variable NODE_ENV:', process.env.NODE_ENV);
```

Результат:

```
PS C:\App_17-10-26> node app.js
Command-line arguments: [ 'C:\\Program Files\\nodejs\\node.exe', 'C:\\App_17-10-
26\\app.js' ]
Environment variable NODE_ENV: undefined
PS C:\App_17-10-26>
```

Объект process обладает множеством других полезных свойств и методов. Например, process.exit() он позволяет немедленно завершить программу с определенным кодом выхода, что может быть полезно для сигнализации об успехе или неудаче другим программам или скриптам.

14.4 Создание вашего первого приложения Node.js

Давайте сделаем первый шаг в мир Node.js, создав простое приложение. В этом приложении мы будем генерировать случайное число от 1 до 100 и отображать его на экране. Это упражнение поможет вам познакомиться с созданием и запуском приложения Node.js на **вашем локальном компьютере** .

Примечание. Для наилучшего обучения мы настоятельно рекомендуем работать локально на вашем компьютере.

Выполните следующие шаги, чтобы создать свое первое приложение Node.js:

1. Создайте новый каталог проекта.

Для начала создайте новую директорию для вашего проекта. Это должна быть пустая папка где-нибудь на вашем компьютере. Затем откройте эту папку в редакторе кода. Для демонстрации примеров мы будем использовать Visual Studio Code . Вы можете выбрать любой удобный для вас вариант.

2. Создайте файл app.js.

Внутри созданной директории создайте новый файл с именем **«app.js.»**. Здесь мы напишем код для нашего Node.js-приложения.

3. Напишите код приложения.

```
const randomNumber = Math.floor(Math.random() * 100) + 1;
console.log("Random Number: ", randomNumber);
```

Этот код использует встроенную Math.random() функцию JavaScript для генерации случайного числа от 0 до 1. Умножив его на 100 и прибавив 1, мы гарантируем, что результат будет находиться в диапазоне от 1 до 100.

4. Запустите приложение.

Откройте терминал или командную строку и убедитесь, что вы находитесь в директории, где расположен app.jsфайл. Выполните следующую команду, чтобы запустить app.jsфайл с помощью Node.js:

```
  node app.js
```

В результате вы должны увидеть случайно сгенерированное число:

```
PS C:\App_17-10-26> node app.js
Random Number:  2
PS C:\App_17-10-26>
```