## Учреждение образования

## Белорусский государственный технологический университет

Кафедра полиграфического оборудования и

системы обработки информации

**Отчет по лабораторной работе**

«Ознакомление со стандартами языков программирования

в части спецификаций *ECMA* 5 и *ECMA* 6»

по дисциплине «Стандартизация и сертификация

информационных систем и технологий»

Выполнил студент

ФИТ ПОИТ 5-2 Валдайцев А. Д.

Проверил

кандидат технических наук

Сулим П.Е.

Отчет по лабораторной работе

защищен с отметкой баллов

Минск 2024

# **Основные различия между ES5 и ES6**

ES5 (ECMAScript 5) и ES6 (ECMAScript 2015) представляют разные версии стандарта ECMAScript, который определяет язык JavaScript. Вот несколько основных отличий между ними:

1. **Блочная область видимости переменных:**
   * ES5: Переменные, объявленные с использованием **var**, имеют функциональную область видимости.
   * ES6: Введены ключевые слова **let** и **const**, которые позволяют объявлять переменные с блочной областью видимости, что улучшает управление областью видимости переменных и избегает проблем, связанных с поднятием переменных (**hoisting**).
2. **Стрелочные функции:**
   * ES6: Введены стрелочные функции (**=>**), которые предоставляют более краткий синтаксис для создания анонимных функций и улучшают читаемость кода.
3. **Классы:**
   * ES6: Добавлена поддержка классов, позволяющих определять объекты и их методы более удобным способом, похожим на классы в других объектно-ориентированных языках программирования.
4. **Деструктуризация:**
   * ES6: Добавлена деструктуризация, которая позволяет извлекать значения из массивов и объектов с помощью синтаксиса, подобного присваиванию.
5. **Параметры по умолчанию и оставшиеся параметры:**
   * ES6: Добавлена поддержка параметров по умолчанию и оставшихся параметров, что облегчает работу с функциями.

# **Даунгрейд кода с ES6 до ES5**

# **NPC.js**

Одним из главных отличией кода в стандартах EcmaScript является то, что в ES5 из-за отсутствия классов необходимо использовать функциональный подход, и NPC является функцией с методами. С помощью Object.defineProperty дефайнятся геттеры и сеттеры для свойств name и pos.

Статический метод create() используется для создания объекта NPC и вынесен в отдельную функцию, а методы walkForward() и walkBack() используются в прототипе NPC.prototype. Пример кода представлен на рисунке 2.1.

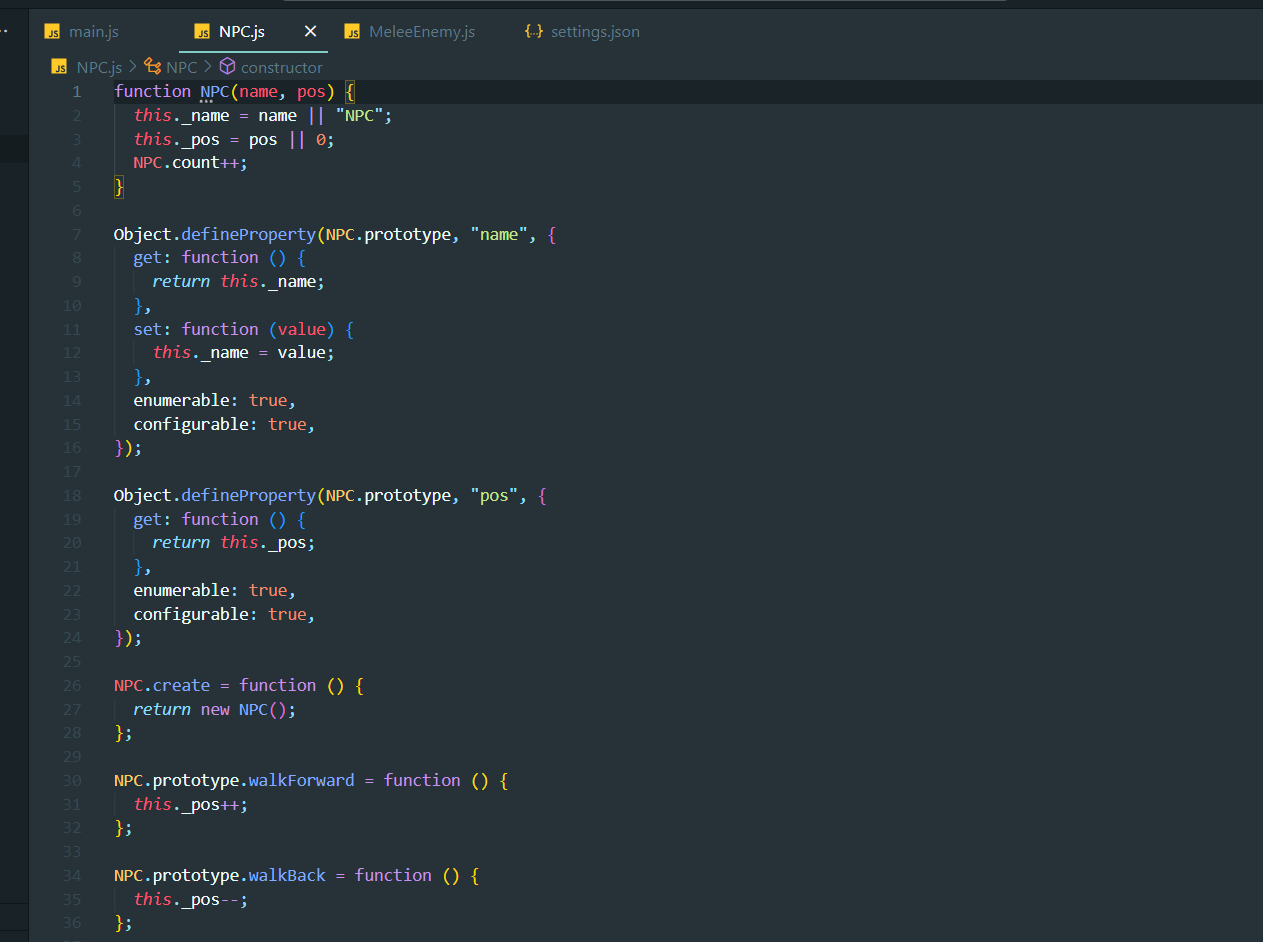


Рисунок 2.1 – код NPC.js на ES5

# **MeleeEnemy.js**

В данном файле также используется функциональный подход вместо классов. Для создания объекта используется Object.create(NPC.prototype). Так как в функциональном подходе отсутствует наследование, то вместо вызова базовой реализации с помощью ключевого слова super в ES5 используется вызов метода call из базовой функции NPC. Таким образом, можно вызвать родительский функционал перед выполнением в дочерней сущности. Это используется в конструкторе и вметодах walkForward() и walkBack(). Пример кода представлен на рисунке 2.2.

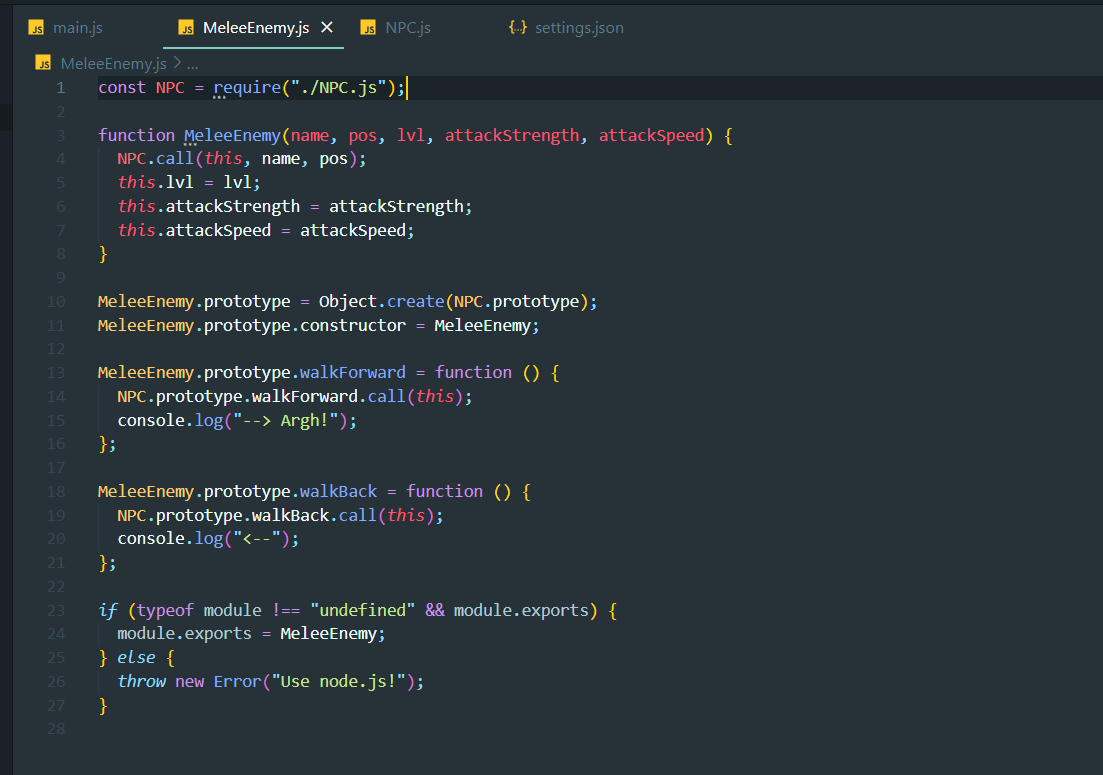


Рисунок 2.2 – Код MeleeEnemy.js на ES5

## **Main.js**

Код функции main.js аналогичен в обоих стандартах – создаются два объекта MeleeEnemy и вызываются функции walkForward() и walkBack(). Аналогичный вывод программ представлен не рисунках 2.3 и 2.4.

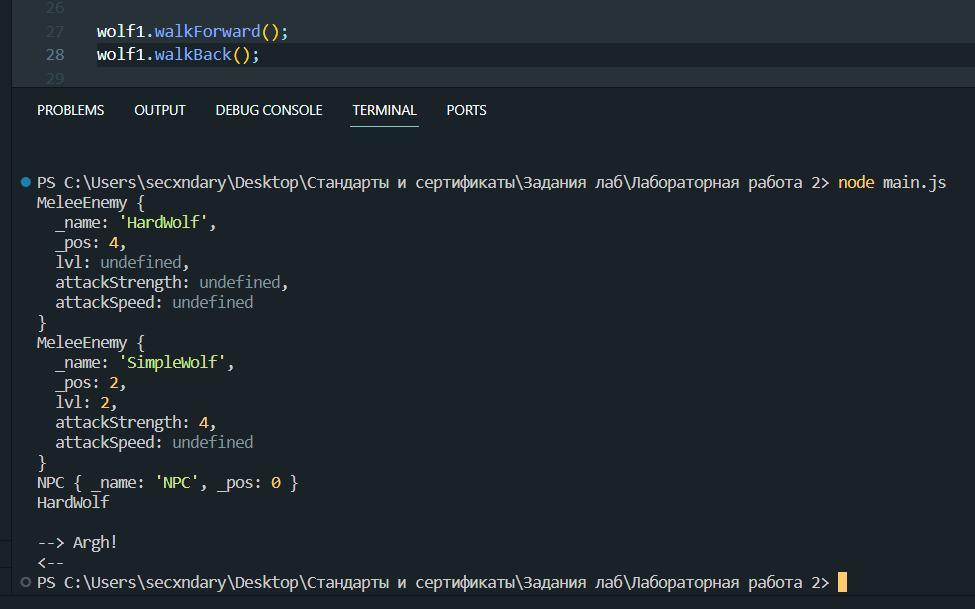
****

Рисунок 2.3 – Вывод программы на ES6

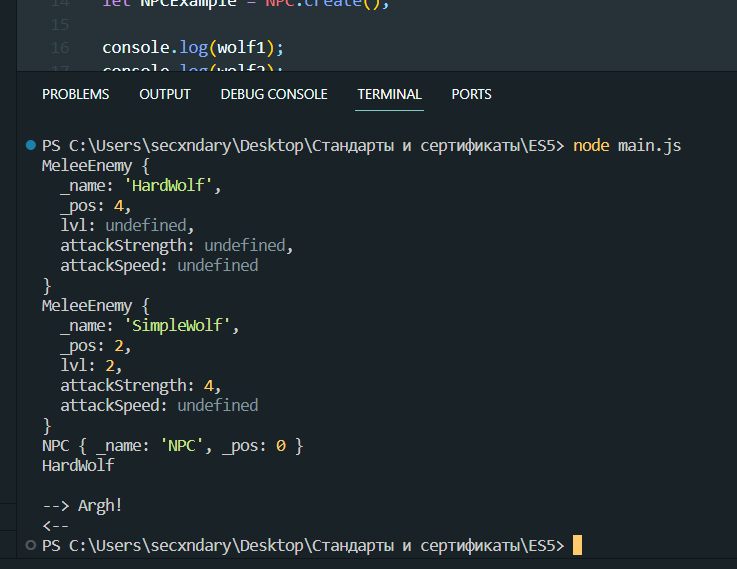


Рисунок 2.4 – Вывод программы на ES5

## **Вывод**

В ходе выполнения данной лабораторной работы был написан код, реализующий необходимый функционал на *ES*5, используя *Node.js.*

## **Источники**

[ES5 vs ES6 - Coding Ninjas](https://www.codingninjas.com/studio/library/es5-vs-es6)

[ES5 vs ES6 - javatpoint](https://www.javatpoint.com/es5-vs-es6)