## Учреждение образования

## Белорусский государственный технологический университет

Кафедра полиграфического оборудования и

системы обработки информации

**ЛАБОРАТОРНАЯ РАБОТА № 2**

по дисциплине «Стандартизация и сертификация

информационных систем и технологий»

Тема

«Ознакомление со стандартами языков программирования

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

Выполнил студент:

Коржова В. С.

4 курс 4 группа

Проверил

кандидат технических наук

Сулим П.Е.

Отчет по лабораторной работе

защищен с отметкой баллов

Минск 2024

**Лабораторная работа № 2**

**ОЗНАКОМЛЕНИЕ СО СТАНДАРТАМИ ЯЗЫКОВ**

**ПРОГРАММИРОВАНИЯ В ЧАСТИ**

**СПЕЦИФИКАЦИЙ *ECMA* 5 И *ECMA* 6. РЕАЛИЗАЦИЯ ООП**

***Цель* *работы*:** Написать код реализующий необходимый функционал на *ES*5, используя *Node.js* или браузер *Google Chrome.*

***Ход работы:*** Необходимо ознакомится с *ES*6 и *ES*5, привести основные различия. Переписать аналогичный функционал на *ES*5, используя *Node.js* или браузер *Google Chrome*.

***Теоретические сведения:***

ECMAScript — это встраиваемый расширяемый не имеющий средств ввода-вывода язык программирования, используемый в качестве основы для построения других скриптовых языков. Стандартизирован международной организацией ECMA в спецификации ECMA-262. Расширения языка: JavaScript, JScript и ActionScript.

ES — это просто сокращение для ECMAScript. Каждое издание ECMAScript получает аббревиатуру ES с последующим его номером.

ES5 был выпущен в декабре 2009 года, спустя 10 лет после выхода третьего издания. Среди изменений можно отметить:

1. поддержку строгого режима (strict mode);
2. аксессоры getters и setters;
3. возможность использовать зарезервированные слова в качестве ключей свойств и ставить запятые в конце массива;
4. многострочные строковые литералы;
5. новую функциональность в стандартной библиотеке;
6. поддержку JSON.

Версия ES6/ES2015 вышла в июне 2015 года. Это также принесло некую путаницу в связи с названием пакета, ведь ES6 и ES2015 — это одно и то же. С выходом этого пакета обновлений комитет принял решение перейти к ежегодным обновлениям. Поэтому издание было переименовано в ES2015, чтобы отражать год релиза. Последующие версии также называются в соответствии с годом их выпуска. В этом обновлении были сделаны следующие изменения:

1. добавлено деструктурирующее присваивание;
2. добавлены стрелочные функции;
3. в шаблонных строках можно объявлять строки с помощью ` (обратных кавычек). Шаблонные строки могут быть многострочными, также могут интерполироваться;
4. let и const — альтернативы var для объявления переменных. Добавлена «временная мертвая зона»;
5. итератор и протокол итерации теперь определяют способ перебора любого объекта, а не только массивов. Symbol используется для присвоения итератора к любому объекту;
6. добавлены функции-генераторы. Они используют yield для создания последовательности элементов. Функции-генераторы могут использовать yield\* для делегирования в другую функцию генератора, кроме этого они могут возвращать объект генератора, который реализует оба протокола;
7. добавлены промисы.

На рисунке 1.1 представлены даты релиза различных версий JavaScript.



Рисунок 1.1 – Даты релиза различных версий JavaScript

**ECMAScript 5 (ES5P):**

ES5 также известен как ECMAScript 2009, поскольку он выпущен в 2009 году. Это функция, которая фокусируется на том, как создаются экземпляры объектов. Для ES5 вы должны написать ключевое слово функции и вернуть его, чтобы использовать для определения функции, как обычный общий язык JavaScript.

**ECMAScript 6 (ES6):**

ES6 также известен как ECMAScript 2015, поскольку он выпущен в 2015 году. Его класс позволяет разработчикам создавать экземпляр объекта с помощью оператора new, используя функцию со стрелкой, в случае, если для определения функции не требуется использовать ключевое слово function, также можно избежать ключевого слова return для извлечения значения компьютера.

В таблице 1.1 представлены различия между ES5 и ES6.

|  |  |
| --- | --- |
| Функция | |
| function greetings (name) { return 'hello ' + name } | const greetings = (name) => { return `hello ${name}`; } |
| Манипулирование объектами | |
| var obj1 = { a: 1, b: 2 } var obj2 = { a: 2, c: 3, d: 4} var obj3 = Object.assign(obj1, obj2) | const obj1 = { a: 1, b: 2 } const obj2 = { a: 2, c: 3, d: 4} const obj3 = {...obj1, ...obj2} |
| Извлечение объектов | |
| var obj1 = { a: 1, b: 2, c: 3, d: 4 } var a = obj1.a var b = obj1.b var c = obj1.c var d = obj1.d | const obj1 = { a: 1, b: 2, c: 3, d: 4 } const { a, b, c, d } = obj1 |
| Объявление объекта | |
| var a = 1 var b = 2 var c = 3 var d = 4var obj1 = { a: a, b: b, c: c, d: d } | var a = 1 var b = 2 var c = 3 var d = 4var obj1 = { a, b, c, d } |
| Экспорт и импорт модулей | |
| var myModule = { x: 1, y: function(){ console.log('This is ES5') }}  module.exports = myModule; | const myModule = { x: 1, y: () => { console.log('This is ES5') }}  export default myModule; |
| var myModule = require('./myModule'); | import myModule from './myModule'; |

Таблица 1.1 – Различия между ES5 и ES6

***Практическая часть:***

|  |  |  |
| --- | --- | --- |
| Имя файла | ES5 | ES6 |
| main.js | const NPC = require("./NPC");  const MeleeEnemy = require("./MeleeEnemy");  const settings = require("./settings");  let wolf1 = new MeleeEnemy(  settings.HardWolf.name,  settings.HardWolf.lvl,  );  let wolf2 = new MeleeEnemy(  settings.SimpleWolf.name,  settings.SimpleWolf.lvl,  settings.SimpleWolf.attackStrength,  settings.SimpleWolf.attackSpeed  );  let NPCExample = NPC.create();  console.log( wolf1 );  console.log( wolf2 );  console.log( NPCExample );  console.log( wolf1.name );  console.log( ); | import NPC from "./NPC.mjs";  import MeleeEnemy from "./MeleeEnemy.mjs";  import settings from "./settings.json";  let wolf1 = new MeleeEnemy(  settings.HardWolf.name,  settings.HardWolf.lvl,  );  let wolf2 = new MeleeEnemy(  settings.SimpleWolf.name,  settings.SimpleWolf.lvl,  settings.SimpleWolf.attackStrength,  settings.SimpleWolf.attackSpeed  );  let NPCExample = NPC.create();  console.log( wolf1 );  console.log( wolf2 );  console.log( NPCExample );  console.log( wolf1.name );  console.log( ); |
| MeleeEnemy.js | const NPC = require("./NPC.js");  class MeleeEnemy extends NPC {  constructor(name, pos, lvl, attackStrength, attackSpeed) {  super(name, pos);  this.lvl = lvl;  this.attackStrength = attackStrength;  this.attackSpeed = attackSpeed;  }  walkForward() {  super.walkForward();  console.log('--> Argh!');  }  walkBack() {  super.walkBack();  console.log('<--')  }  }  if (module.exports) {  module.exports = MeleeEnemy;  } else {  throw new Exception("Use node.js!");  } | import NPC from "./NPC.mjs";  export default class MeleeEnemy extends NPC {  constructor(name, pos, lvl, attackStrength, attackSpeed) {  super(name, pos);  this.lvl = lvl;  this.attackStrength = attackStrength;  this.attackSpeed = attackSpeed;  }  walkForward = () => {  this.\_pos++;  }  walkBack = () => {  this.\_pos++;  }  } |
| NPC.js | class NPC {  constructor(){  var name = arguments.length <= 0 || arguments[0] === undefined ? "NPC" : arguments[0];  var pos = arguments.length <= 1 || arguments[1] === undefined ? 0 : arguments[1];  this.\_name = name;  this.\_pos = pos;  NPC.count++;  }  get name() {  return this.\_name;  }  set name(value) {  this.\_name = value;  }  get pos() {  return this.\_pos;  }  static create() {  return new NPC();  }  walkForward() {  this.\_pos++;  }  walkBack() {  this.\_pos++;  }  }  NPC.count = 0;  if (module.exports) {  module.exports = NPC;  } else {  throw new Error("Use Node.js!");  } | export default class NPC {  constructor(name = "NPC", pos = 0) {  this.\_name = name;  this.\_pos = pos;  NPC.count++;  }  get name() {  return this.\_name;  }  set name(value) {  this.\_name = value;  }  get pos() {  return this.\_pos;  }  static create() {  return new NPC();  }  walkForward = () => {  this.\_pos++;  }  walkBack = () => {  this.\_pos++;  }  }  NPC.count = 0; |

**Выводы:** В процессе выполнения этой лабораторной работы было проведено сравнение между стандартами ES5 и ES6, а также выполнен перевод исходного кода, созданного с использованием более старого стандарта, в эквивалентный код, соответствующий стандарту ES6.