Лабораторная работа № 3

Тема: Функции-конструкторы.

Цель: Освоить создание объектов в JavaScript с помощью фабричных функций, функций-конструкторов и встроенных конструкторов. Научиться применять оператор new, разбираться в ограничениях функций-конструкторов и использовать встроенные типы (Array, Date, Function).

🔷 Вариант 1

1. Написать фабричную функцию createBook(title, author), возвращающую объект книги с полями title, author и методом getInfo().

2. Реализовать функцию-конструктор User(name, age) с методом sayHello(). Создать 2 экземпляра.

3. Проверить проблему дублирования методов у разных объектов, созданных через конструктор (сравнить ссылки на метод).

4. Создать объект даты с помощью конструктора Date, вывести текущий год и месяц.

5. Использовать конструктор Function для динамического создания функции сложения чисел и протестировать её.

🔷 Вариант 2

1. Написать фабричную функцию createCar(brand, model), возвращающую объект машины с методом drive().

2. Создать функцию-конструктор Product(name, price) с методом getPrice(). Проверить работу нескольких экземпляров.

3. Добавить в конструктор Product возврат простого объекта через return {…}. Объяснить результат.

4. Создать массив через new Array(5), заполнить его случайными числами от 1 до 100, найти минимальное и максимальное значение.ты.

5. С помощью конструктора Date определить день недели для даты рождения (например, new Date(2000, 0, 1)).

⚡ **Отчет должен содержать (см. образец):**
- номер и тему лабораторной работы;
- фамилию, номер группы студента и вариант задания;
- скриншоты окна VSC с исходным кодом программ;
- скриншоты с результатами выполнения программ;
- пояснения, если необходимо;
- выводы.

Отчеты в формате **pdf** отправлять на email:
colledge20education23@gmail.com

Шпаргалка по фабричным функциям и функциям-конструкторам
1. Фабричные функции
Обычная функция, возвращающая объект.

```
function createUser(name, age) {
  return {
    name,
    age,
    sayHi() {
      console.log(`Привет, я ${this.name}`);
    }
  };
}

const user1 = createUser("Аня", 20);
user1.sayHi(); // Привет, я Аня
```

2. Функция-конструктор
Функция для создания объектов через new.
По соглашению начинается с заглавной буквы.

```
function User(name, age) {
  this.name = name;
  this.age = age;
  this.sayHi = function() {
```

```
    console.log(`Привет, я ${this.name}`);
  };
}

const u1 = new User("Петя", 25);
u1.sayHi(); // Привет, я Петя
```

3. Как работает new (шаги)

```
new User("Вася", 30)
// 1. Создаётся пустой объект {}
// 2. Этот объект привязывается к this внутри функции
// 3. Выполняется тело функции
// 4. Возвращается объект (если return не указан явно)
```

4. Ограничения функций-конструкторов

Если вызвать без new → this = undefined (в строгом режиме).

Методы внутри конструктора дублируются для каждого объекта → лишний расход памяти.

Нельзя использовать стрелочные функции как конструкторы (new () => {} даст ошибку).

Конструктор может вернуть объект через return, и тогда он заменит созданный через new.

```
function Test() {
  this.x = 10;
  return { y: 20 }; // заменяет объект с this
}
console.log(new Test()); // { y: 20 }
```

5. Встроенные конструкторы

Массив (Array)

```
const arr = new Array(3, 5, 7);
console.log(arr.length); // 3
console.log(arr);        // [3, 5, 7]
const nums = new Array(5); // [пустых 5]
for (let i = 0; i < nums.length; i++) {
  nums[i] = Math.floor(Math.random() * 100) + 1;
}
console.log(nums);
```

```
console.log("min =", Math.min(...nums), "max =",
Math.max(...nums));
```

Дата (Date)

```
const date = new Date(2000, 0, 1); // 1 января 2000
console.log(date.toDateString());  // Sat Jan 01 2000
console.log("День недели:", date.getDay()); // 6 (сб)

const now = new Date();
const endYear = new Date(now.getFullYear(), 11, 31);
const diff = Math.ceil((endYear - now) /
(1000*60*60*24));
console.log("Дней до конца года:", diff);
```

Функция (Function)

```
const sum = new Function("a", "b", "return a + b");
console.log(sum(2, 3)); // 5
```