Лабораторная работа 6

Spread, деструктуризация, модули, динамический импорт

(20 баллов)

Выполните самостоятельно следующие задания и оформите отчет.

Требования по отчету:

Наличие титульного листа. Размер страницы должен соответствовать формату А4 (210х297), размеры полей: левое – 30 мм, правое – 10 мм, верхнее – 15 мм, нижнее – 20 мм. Шрифт Times new Roman, размер 14 pt полуторный междустрочный интервал. Выравнивание текста – по ширине, красная строка – 1,25 см, отступ слева и справа – 0 мм.

Spread оператор

1. Дан массив с числами. Используя Math.min и spread, выведите на экран минимальное значение массива.
2. Имеется следующий код:

```
let arr1 = [1, 2, 3];
let arr2 = [...arr1, 4, 5, 6];
let arr3 = [...arr2, 7, 8, 9];

let arr  = <>;
console.log(arr);
```

   Что нужно подставить вместо <> (используя arr3 и spread), чтобы в arr появился массив чисел от 0 до 11 включительно.
3. Даны два варианта инициализации:
   a. `let arr = [...12345];`
   b. `let arr = [..."12345"];`

   Что произойдет при запуске кода каждого варианта и почему?
4. Напишите функцию mean, которая будет принимать параметрами произвольное количество чисел и возвращать их среднее арифметическое.
5. Напишите функцию unite, которая параметрами будет принимать произвольное количество массивов и сливать их в один двухмерный.

```
let result = unite([1, 2, 3], [4, 5, 6], [7, 8, 9]);
console.log(result); // выведет [ [1, 2, 3,] [4, 5, 6], [7, 8, 9] ]
```

6. Напишите функцию merge, параметрами принимающую произвольное количество массивов и сливающую их элементы в один массив.

```
let result = merge([1, 2, 3], [4, 5, 6], [7, 8, 9]);
console.log(result); // выведет [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Деструктуризация массивов

7. В следующем коде части массива записываются в соответствующие переменные:

```
let arr = ['John', 'Smit', 'development', 'programmer', 2000]

let name = arr[0]
let surname = arr[1]
let department = arr[2]
let position = arr[3]
let salary = arr[4]
```

   Переделайте этот код через деструктуризацию.
8. В следующем коде части массива записываются в соответствующие переменные:

```
function func () {
  return ['John', 'Smit', 'development', 'programmer', 2000]
}

let arr = func()

let name = arr[0]
let surname = arr[1]
let department = arr[2]
let position = arr[3]
let salary = arr[4]
```

   Переделайте этот код через деструктуризацию.
9. В следующем коде части массива записываются в соответствующие переменные:

```
let arr = ['John', 'Smit', 'development', 'programmer', 2000]

let department = arr[2]
let position = arr[3]
```

   Переделайте этот код через деструктуризацию, используя пропуски значений.
10. В следующем коде части массива записываются в соответствующие переменные:

```
let arr = ['John', 'Smit', 'development', 'programmer', 2000]

let name = arr[0]
let surname = arr[1]

let info = arr.slice(2) // все элементы со второго до конца массива
```

    Переделайте этот код через деструктуризацию, используя запись части значений в массив.
11. В следующем коде части массива записываются в соответствующие переменные:

```
let arr = ['John', 'Smit', 'development', 'programmer']

let name = arr[0]
let surname = arr[1]
```

```
let department = arr[2]

let position
if (arr[3] !== undefined) {
  position = arr[3]
} else {
  position = 'trainee'
}
```

Переделайте этот код через деструктуризацию, используя значения по умолчанию.

12. В следующем коде значение по умолчанию указывается в виде результата работы функции:

```
function func () {
  return new Date().getDate()
}

let [year, month, day = func()] = arr
```

Модифицируйте код так, чтобы при отсутствии в массиве значения для месяца по умолчанию брался текущий месяц, а при отсутствии значения для года - соответственно текущий год.

13. Переделайте следующий код через деструктуризацию параметров функции:

```
function func (employee) {
  let name = employee[0]
  let surname = employee[1]
  let department = employee[2]
  let position = employee[3]
  let salary = employee[4]
}

func(['John', 'Smit', 'development', 'programmer', 2000])
```

14. Переделайте следующий код через деструктуризацию параметров функции:

```
function func (employee) {
  let name = employee[0]
  let surname = employee[1]
  let department = employee[2]

  let position
  if (arr[3] !== undefined) {
    position = arr[3]
  } else {
    position = 'джуниор'
  }
}

func(['John', 'Smit', 'development'])
```

15. Переделайте следующий код через деструктуризацию параметров функции:

```
function func (department, employee, hired) {
  let name = employee[0]
  let surname = employee[1]

  let year = hired[0]
  let month = hired[1]
  let day = hired[2]
}

func('development', ['John', 'Smit'], [2018, 12, 31])
```

Деструктуризация объектов

16. В следующем коде части объекта записываются в соответствующие переменные:

```
let options = {
  color: 'red',
  width: 400,
  height: 500
}

let color = options.color
let width = options.width
let height = options.height
```

Переделайте этот код через деструктуризацию.

17. В следующем коде части объекта записываются в соответствующие переменные:

```
let options = {
  color: 'red',
  width: 400,
  height: 500
}

let c = options.color
let w = options.width
let h = options.height
```

Переделайте этот код через деструктуризацию, чтобы имена переменных не совпадали с именами ключей объекта.

18. В следующем коде части объекта записываются в соответствующие переменные:

```
let options = {
  width: 400,
  height: 500
}

let color
if (options.color !== undefined) {
  color = options.color
```

```
} else {
  color = 'black'
}

let width = options.width
let height = options.height
```

Переделайте этот код через деструктуризацию, используя значения по умолчанию.

19. В следующем коде части объекта записываются в соответствующие переменные:

```
let options = {
  width: 400,
  height: 500
}

let c
if (options.c !== undefined) {
  c = options.color
} else {
  c = 'black'
}

let w = options.width
let h = options.height
```

Переделайте этот код через деструктуризацию, используя значения по умолчанию и названия переменных, не совпадающих с ключами объекта.

20. Переделайте следующий код через деструктуризацию параметров функции:

```
function func (options) {
  let color = options.color
  let width = options.width
  let height = options.height
}

func({ color: 'red', width: 400, height: 500 })
```

21. Переделайте следующий код через деструктуризацию параметров функции:

```
function func (options) {
  let width = options.width
  let height = options.height

  let color
  if (options.color !== undefined) {
    color = options.color
  } else {
    color = 'black'
  }
}

func({ color: 'red', width: 400, height: 500 })
```

Модули, динамический импорт

22. Возьмите версию задания №2 из лабораторной №4, которая реализована в лабораторной №5 через async/await. Поместите каждую функцию в отдельный модуль. В главном файле выполните динамический импорт всех функций из соответствующих модулей и запустите их в нужном порядке.