Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

**Лабораторная работа №2**

**по дисциплине**

**«Графический и веб-дизайн»**

**АСИНХРОННОЕ ПРОГРАММИРОВАНИЕ**

**Выполнил**:

ст. гр. ПРИ-120

Д. А. Грачев

**Принял**:

Шамышев А. А.

Владимир, 2023

ЦЕЛЬ РАБОТЫ

Изучить принципы работы цикла событий в JavaScript; получить навыки написания асинхронного кода с помощью колбэков, промисов и асинхронных функций; освоить механизм вещания и обработки событий.

ВЫПОЛНЕНИЕ РАБОТЫ

1. Был сформирован html-файл, куда был подключен скрипт с js

<!doctype *html*>  
<html *lang*="ru">  
<head>  
 <meta *charset*="UTF-8">  
 <meta *name*="viewport"  
 *content*="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta *http-equiv*="X-UA-Compatible" *content*="ie=edge">  
 <title>Лабораторная работа 2</title>  
 <script *src*="script.js"></script>  
</head>  
<body>  
 <h1>Название</h1>  
 <br>  
</body>  
</html>

1. Была написана функция для подключения скрипта на страницу

*function* loadScript(src, callback) {  
 *let* script = document.createElement('script')  
 script.src = src  
 script.onload = () => callback(*null*)  
 script.onerror = () => callback(  
 *new* Error(`Не удалось загрузить скрипт ${src}`)  
 )  
 document.head.append(script)  
}

1. Реализовать условную конструкцию с использованием на колбэков и промисов
   1. Листинг

*function* conditional(conditional, action1, action2, action3, action4) {  
 action1()  
 *if* (conditional) {  
 action2()  
 } *else* {  
 action3()  
 }  
 action4()  
}

conditional(*true*,  
 () => print('1.Выполнилось первое действие '),  
 () => print('1.Выполнилось второе действие '),  
 () => print('1.Выполнилось третье действие '),  
 () => print('1.Выполнилось четвертое действие ')  
)

* 1. Описание работы скрипта: в функцию передается несколько колбэков, которые выполняются в зависимости от переданного условия

1. Реализовать цикл с использованием колбэков и промисов
   1. Листинг

*function* loop(loopCount, action1, action2, action3, eventTarget) {  
 *let* count = 1  
 action1()  
 *while* (count < loopCount) {  
 eventTarget.dispatchEvent(event)  
 action2(count)  
 count += 1  
 }  
 action3()  
}

loop(6,  
 () => print('2.Выполнилось первое действие '),  
 (iteration) => print(`2.Выполнилось второе действие ${iteration} раз `),  
 () => print('2.Выполнилось третье действие '),  
 target  
)

* 1. Описание работы скрипта: в функцию передается количество итераций и несколько колбэков, один из которых выполняется в цикле указанное количество раз

1. Переписать задачи 2 и 3 с использованием асинхронных функций
   1. Листинг

*async function* conditionalAsync(conditional, action1, action2, action3, action4) {  
 *await* action1(1, 3)  
 *if* (conditional) {  
 *await* action2(2, 3)  
 } *else* {  
 action3(3, 3)  
 }  
 *await* action4(4, 3)  
}  
  
*async function* loopAsync(loopCount, action1, action2, action3) {  
 *let* count = 1  
 *await* action1()  
 *while* (count < loopCount) {  
 *await* action2()  
 count += 1  
 }  
 action3()  
}

*function* secondFunction(x, y) {  
 *return new* Promise(resolve => {  
 setTimeout(() => {  
 print(`${y}.Выполнилось ${x} действие `)  
 resolve(x, y)  
 }, 1000)  
 })  
}

conditionalAsync(*true*,  
 secondFunction,  
 secondFunction,  
 secondFunction,  
 secondFunction  
)  
  
loopAsync(4,  
 () => secondFunction(1, 4),  
 () => secondFunction(2, 4),  
 () => print('4.Выполнилось 3 действие '),  
)

* 1. Описание работы скрипта: работа аналогично предыдущим двум заданиям, за исключением того, что, все функции стали асинхронными и ожидают выполнения каждого колбэка, который выполняется с задержкой

1. Создать событие и подписаться на него
   1. Листинг

*const* event = *new* Event('clack')

*const* target = *new* EventTarget()  
target.addEventListener(  
 "clack",  
 () => print('<br/>Всем привет!')  
)

* 1. Описание работы алгоритма: был создано событие с названием “clack”, который вызывается в цикле в функции loop

1. Полный листинг основного файла скрипта:

*function* loadScript(src, callback) {  
 *let* script = document.createElement('script')  
 script.src = src  
 script.onload = () => callback(*null*)  
 script.onerror = () => callback(  
 *new* Error(`Не удалось загрузить скрипт ${src}`)  
 )  
 document.head.append(script)  
}  
  
*function* print(text) {  
 document.body.innerHTML += text  
}  
  
*function* secondFunction(x, y) {  
 *return new* Promise(resolve => {  
 setTimeout(() => {  
 print(`${y}.Выполнилось ${x} действие `)  
 resolve(x, y)  
 }, 1000)  
 })  
}  
  
*const* target = *new* EventTarget()  
target.addEventListener(  
 "clack",  
 () => print('<br/>Всем привет!')  
)  
  
loadScript('functions.js', *async function* (error) {  
 *if* (error) {  
 console.error(error)  
 } *else* {  
 conditional(*true*,  
 () => print('1.Выполнилось первое действие '),  
 () => print('1.Выполнилось второе действие '),  
 () => print('1.Выполнилось третье действие '),  
 () => print('1.Выполнилось четвертое действие ')  
 )  
 print('<br/>')  
 loop(6,  
 () => print('2.Выполнилось первое действие '),  
 (iteration) => print(`2.Выполнилось второе действие ${iteration} раз `),  
 () => print('2.Выполнилось третье действие '),  
 target  
 )  
 print('<br/>')  
 conditionalAsync(*true*,  
 secondFunction,  
 secondFunction,  
 secondFunction,  
 secondFunction  
 )  
 print('<br/>')  
 loopAsync(4,  
 () => secondFunction(1, 4),  
 () => secondFunction(2, 4),  
 () => print('4.Выполнилось 3 действие '),  
 )  
 }  
})

1. Полный листинг файла functions.js

*const* event = *new* Event('clack')  
  
  
*function* conditional(conditional, action1, action2, action3, action4) {  
 action1()  
 *if* (conditional) {  
 action2()  
 } *else* {  
 action3()  
 }  
 action4()  
}  
  
*function* loop(loopCount, action1, action2, action3, eventTarget) {  
 *let* count = 1  
 action1()  
 *while* (count < loopCount) {  
 eventTarget.dispatchEvent(event)  
 action2(count)  
 count += 1  
 }  
 action3()  
}  
  
*async function* conditionalAsync(conditional, action1, action2, action3, action4) {  
 *await* action1(1, 3)  
 *if* (conditional) {  
 *await* action2(2, 3)  
 } *else* {  
 action3(3, 3)  
 }  
 *await* action4(4, 3)  
}  
  
*async function* loopAsync(loopCount, action1, action2, action3) {  
 *let* count = 1  
 *await* action1()  
 *while* (count < loopCount) {  
 *await* action2()  
 count += 1  
 }  
 action3()  
}

1. Скриншот результата. На нем можно заметить порядок выполнения скриптов и его изменение в асинхронных функциях

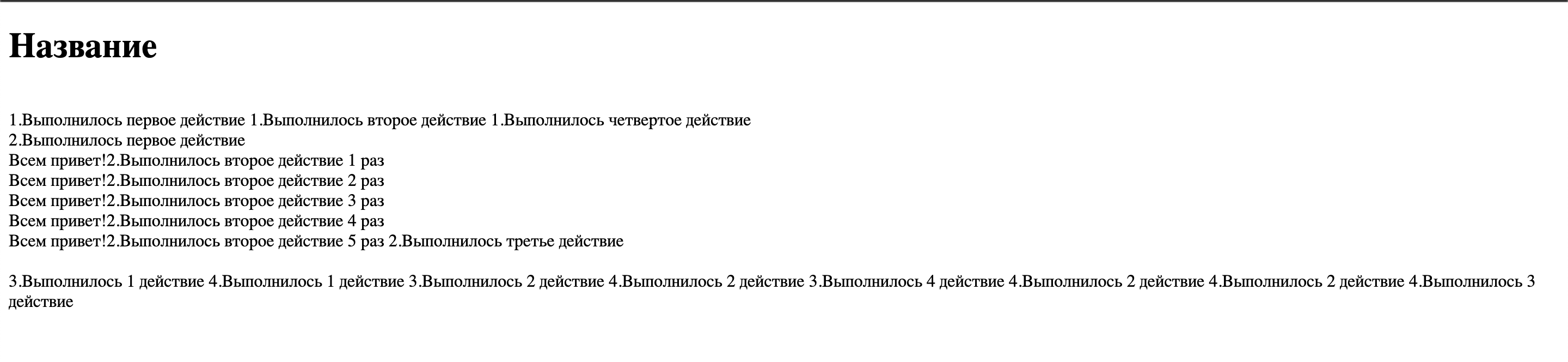


Рисунок . Результат работы всех скриптов

ВЫВОД

В ходе выполнения работы были изучены принципы работы цикла событий в JavaScript, получены навыки написания асинхронного кода с помощью колбэков, промисов и асинхронных функций, освоен механизм вещания и обработки событий.