

Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №42 «Криптология и кибербезопасность»



## ОТЧЁТ

О выполнении лабораторной работы №4 «SCSS/SASS. TypeScript»

по дисциплине «Основы разработки Frontend»

**Выполнил:** Дзарданов Г.В.

**Группа:** Б22-525

**Преподаватель:** Степанова Анна

## СОДЕРЖАНИЕ

1 Лабораторная «SCSS/SASS» .....	3
1.1 Задание №1. Настройка окружения.....	3
1.2 Задание №2. Генерация динамических стилей. Циклы SCSS .....	7
1.3 Задание №3. Вложенность и mixin-ы.....	9
1.4 Задание №4. Изучение синтаксисов. Сравнение SCSS/SASS .....	12
1.5 Задание №5. Синтаксис Less .....	15
2 Лабораторная «TypeScript».....	19
2.1 Задание №1. Настройка среды .....	19
2.2 Задание №2. Создание и типизация класса. IUser .....	20
2.3 Задание №3. Типизация с использованием псевдонимов. Type .....	22
2.4 Задание №4. Перегрузка функций .....	24
2.5 Задание №5. Бинарное дерево.....	26
2.6 Задание №6. Реализация паттернов Adapter, Strategy, Observer.....	27
Заключение.....	37

# 1 Лабораторная «SCSS/SASS»

Данная лабораторная работа направлена на развитие навыков структурированной и эффективной стилизации веб-интерфейсов с использованием препроцессора SCSS/SASS.

## 1.1 Задание №1. Настройка окружения

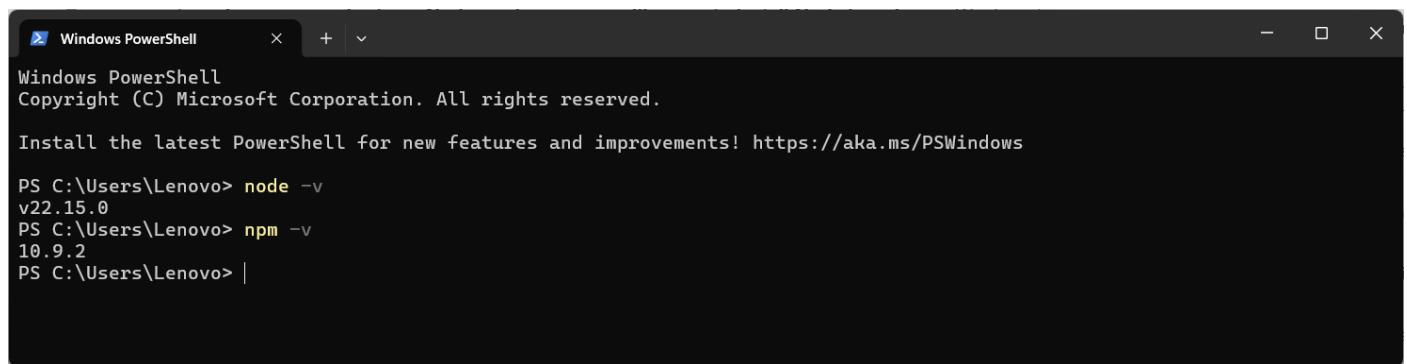
### Задание

Установите SASS/SCSS и запустите компиляцию в реальном времени из sass/scss файла в css файл (так, чтобы при любом изменении в исходном sass/scss файле выполнялась перекомпиляция в css).

### Выполнение

С помощью файла `node-v22.15.0-x64.msi` установим Node.js и пакетный менеджер npm на систему под управлением OS Windows 11. С помощью следующих команд проверим успешность установки:

```
node -v  
npm -v
```



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command `node -v` was run and returned the version `v22.15.0`. The command `npm -v` was run and returned the version `10.9.2`.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

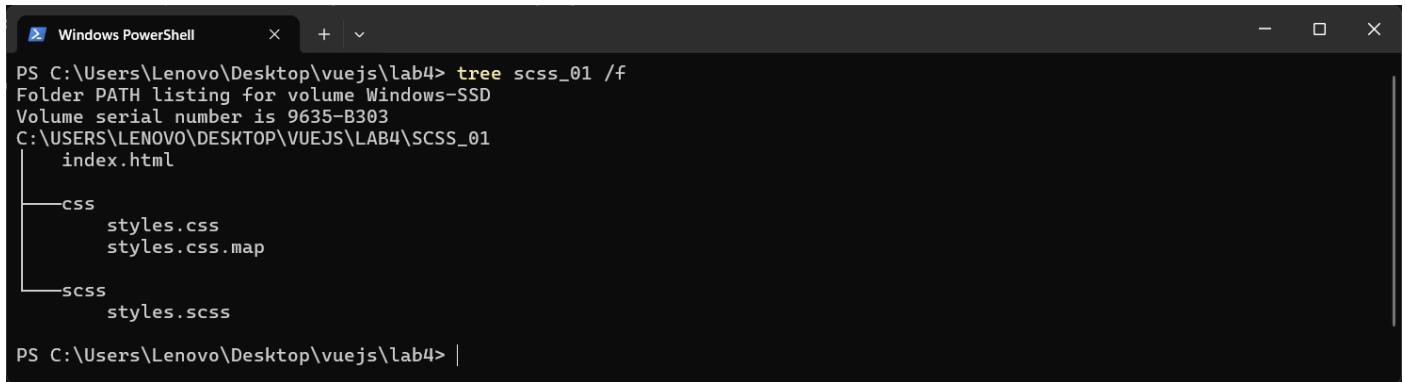
PS C:\Users\Lenovo> node -v
v22.15.0
PS C:\Users\Lenovo> npm -v
10.9.2
PS C:\Users\Lenovo> |
```

Рис. 1 – Проверка успешности установки Node.js и npm

Далее установим SCSS с помощью команды ниже:

```
npm install -g sass
```

Поставим отслеживание на изменение scss-файла (директории, содержащей scss-файлы), автоматически перекомпилирующее изменённый scss в css. В качестве целевой директории возьмём директорию с файлами для Задания №2.



```
PS C:\Users\Lenovo\Desktop\vuejs\lab4> tree scss_01 /f
Folder PATH listing for volume Windows-SSD
Volume serial number is 9635-B303
C:\USERS\LENOVO\DESKTOP\VUEJS\LAB4\SCSS_01
    index.html

    --css
        styles.css
        styles.css.map

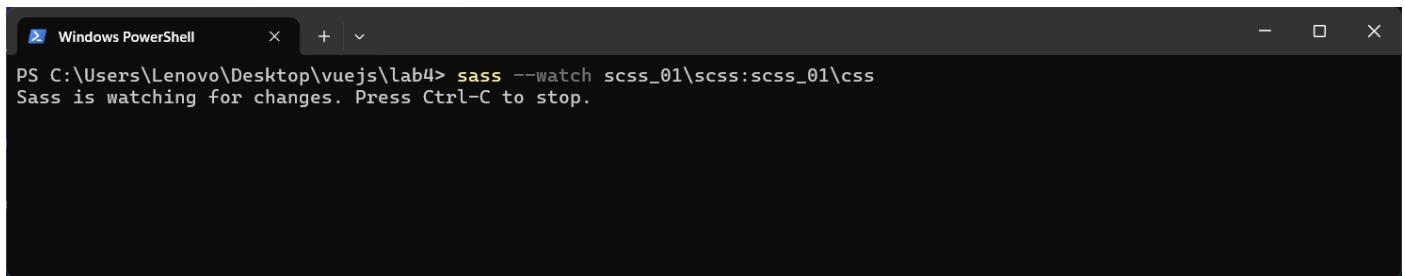
    --scss
        styles.scss

PS C:\Users\Lenovo\Desktop\vuejs\lab4> |
```

Рис. 2 – Просмотр директории, содержащей целевой каталог

Для установки наблюдения пропишем следующую команду:

**sass --watch scss\_01\scss:scss\_01\css** (1.1.4)



```
PS C:\Users\Lenovo\Desktop\vuejs\lab4> sass --watch scss_01\scss:scss_01\css
Sass is watching for changes. Press Ctrl-C to stop.
```

Рис. 3 – Установка наблюдения за изменениями внутри директории scss-файлов

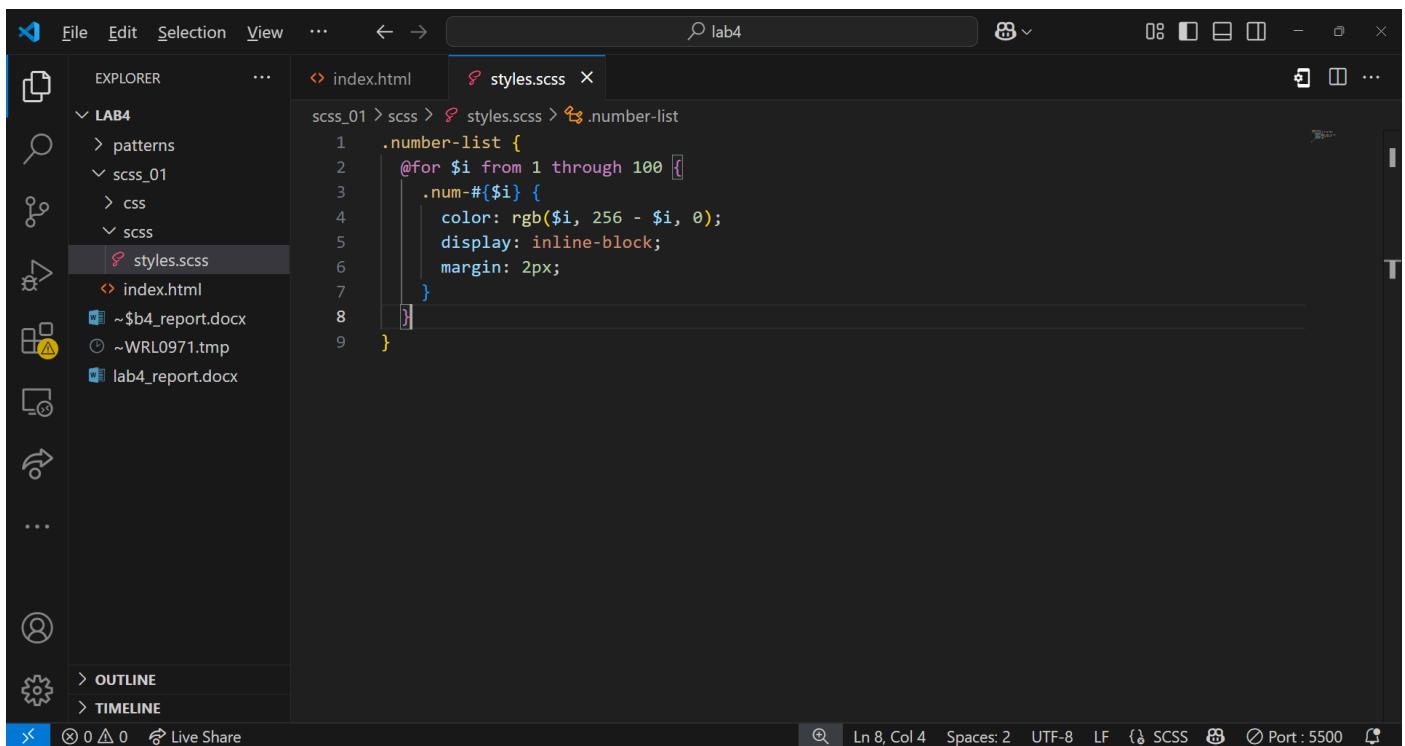


Рис. 4 – Листинг кода scss-файла до изменений

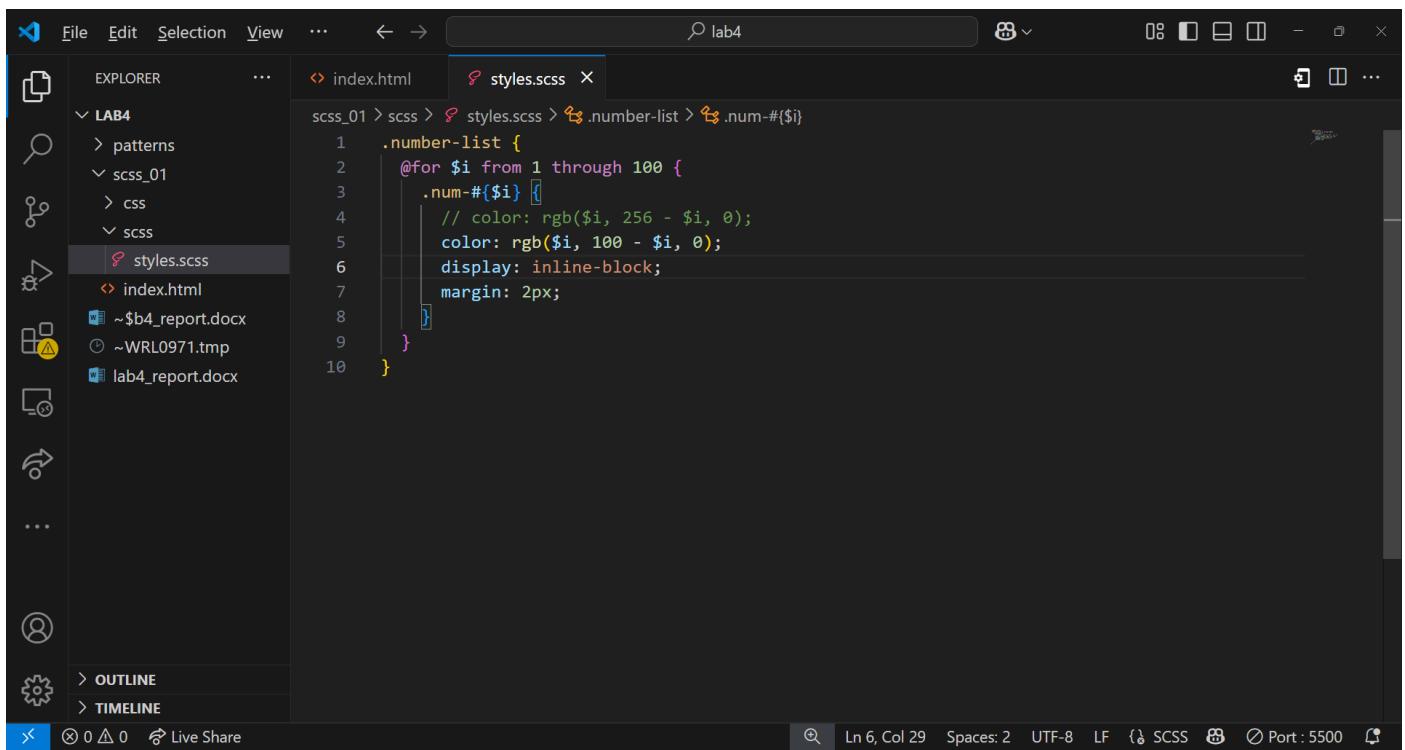


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57  
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

## Lab 4 «SCSS/SASS» Tasks 1, 2

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57  
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Рис. 5 – Отображение веб-страницы до внесения изменений



File Edit Selection View ... < > lab4 index.html styles.scss

EXPLORER LAB4 patterns scss\_01 css scss styles.scss index.html ~\$b4\_report.docx ~WRL0971.tmp lab4\_report.docx

```
scss_01 > scss > styles.scss > .number-list > .num-#${$i}
1   .number-list {
2     @for $i from 1 through 100 {
3       .num-#${$i} [
4         // color: rgb($i, 256 - $i, 0);
5         color: rgb($i, 100 - $i, 0);
6         display: inline-block;
7         margin: 2px;
8       ]
9     }
10 }
```

Ln 6, Col 29 Spaces: 2 UTF-8 LF { SCSS Port: 5500

Рис. 6 – Листинг кода scss-файла после изменений (отменок цвета)



## Lab 4 «SCSS/SASS» Tasks 1, 2

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57  
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Рис. 7 – Отображение изменений на веб-странице после внесения изменений

```
[2025-05-01 23:32] Compiled scss_01\scss\styles.scss to scss_01\css\styles.css.  
[2025-05-01 23:33] Compiled scss_01\scss\styles.scss to scss_01\css\styles.css.  
[2025-05-01 23:33] Compiled scss_01\scss\styles.scss to scss_01\css\styles.css.  
[2025-05-01 23:33] Compiled scss_01\scss\styles.scss to scss_01\css\styles.css.
```

Рис. 8 – Отображение перекомпиляции в панели наблюдения за изменениями

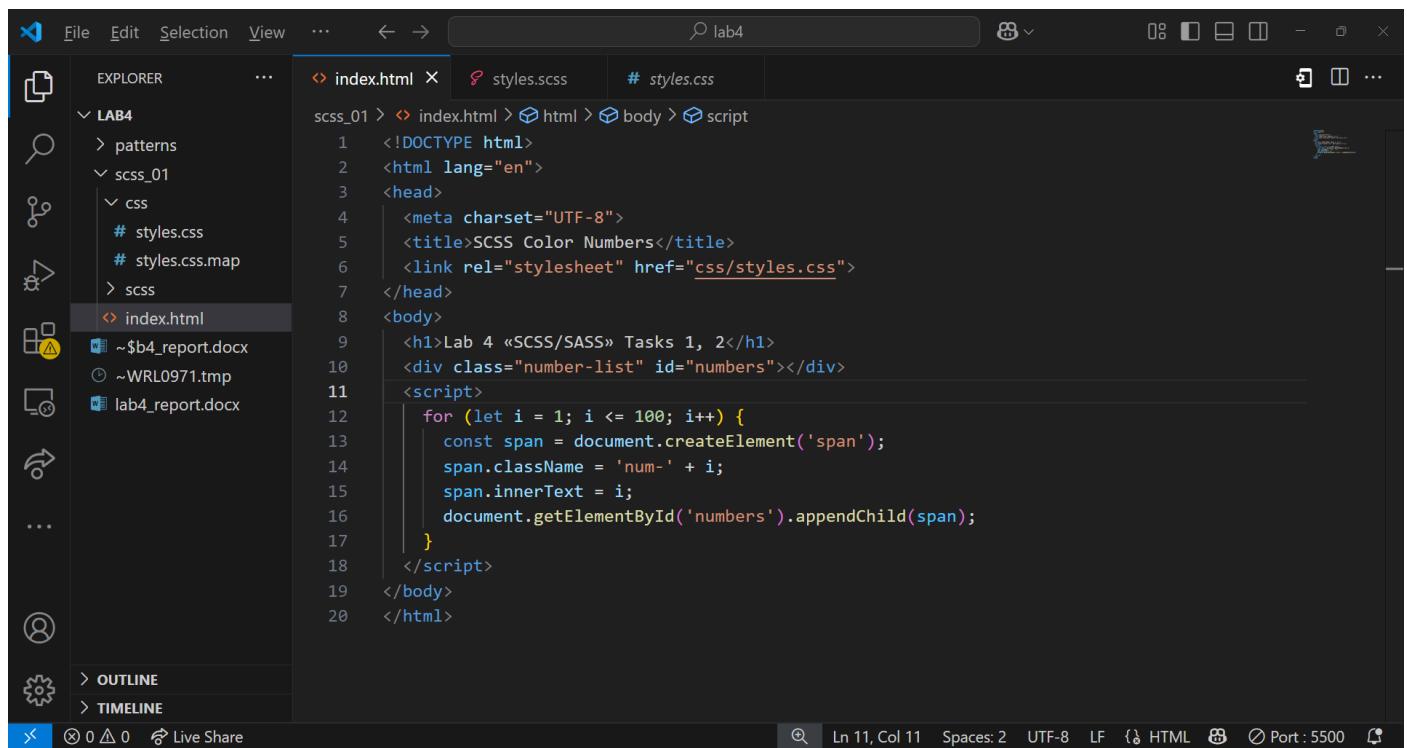
## 1.2 Задание №2. Генерация динамических стилей. Циклы SCSS

### Задание

Сделать страницу со списком чисел от 1 до 100 и покрасить каждое из них в свой цвет (red: n, green: 256-n, blue: 0), где n - текущее число. Использовать циклы SCSS/SASS.

### Выполнение

Структура проекта отражена на Рис. 2, визуальная демонстрация веб-страницы и scss-код приведены на Рис. 5, 6. Листинг кода .html-файла отражён ниже на Рис. 9:



The screenshot shows a code editor interface with the following details:

- Project Structure:** The left sidebar shows a folder named "LAB4" containing "patterns", "scss\_01" (which contains "css" with files "# styles.css" and "# styles.css.map"), and "scss". A file named "index.html" is selected in the list.
- Code Editor:** The main pane displays the content of "index.html".

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>SCSS Color Numbers</title>
<link rel="stylesheet" href="css/styles.css">
</head>
<body>
<h1>Lab 4 «SCSS/SASS» Tasks 1, 2</h1>
<div class="number-list" id="numbers"></div>
<script>
for (let i = 1; i <= 100; i++) {
  const span = document.createElement('span');
  span.className = 'num-' + i;
  span.innerText = i;
  document.getElementById('numbers').appendChild(span);
}
</script>
</body>
</html>
```
- Bottom Status Bar:** Shows file statistics: 0△0, Live Share, Ln 11, Col 11, Spaces: 2, UTF-8, LF, HTML, Port: 5500.

Рис. 9 – Листинг кода html-файла

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, ...
- Search Bar:** / lab4
- Toolbar:** Back, Forward, Home, Stop, Refresh, etc.
- Explorer Sidebar:** Shows a project structure under "LAB4". The "css" folder contains "# styles.css", "# styles.css.map", "scss", "index.html", "scss\_02", and two temporary files. "# styles.css" is currently selected.
- Code Editor:** Displays the content of "# styles.css". The code defines a ".number-list" class with numbered items from 7 to 11, each having a margin of 2px and displayed as inline-block elements with specific colors (e.g., num-7 is green).
- Bottom Status Bar:** ShowsLn 1, Col 1, Spaces: 2, UTF-8, LF, { CSS, Port: 5500, Live Share icon.

```
# styles.css
scss_01 > css > # styles.css > .number-list .num-1
31 .number-list .num-7 {
32   display: inline-block;
33   margin: 2px;
34 }
35 .number-list .num-8 {
36   color: #rgb(8, 248, 0);
37   display: inline-block;
38   margin: 2px;
39 }
40 .number-list .num-9 {
41   color: #rgb(9, 247, 0);
42   display: inline-block;
43   margin: 2px;
44 }
45 .number-list .num-10 {
46   color: #rgb(10, 246, 0);
47   display: inline-block;
48   margin: 2px;
49 }
50 .number-list .num-11 {
51   color: #rgb(11, 245, 0);
52   display: inline-block;
53   margin: 2px;
54 }
```

Рис. 10 – Листинг кода сгенерированного css-файла

### 1.3 Задание №3. Вложенность и mixin-ы

#### Задание

Сверстайте страницу, как на картинке, с применением вложенных правил и миксинов SCSS/SASS:

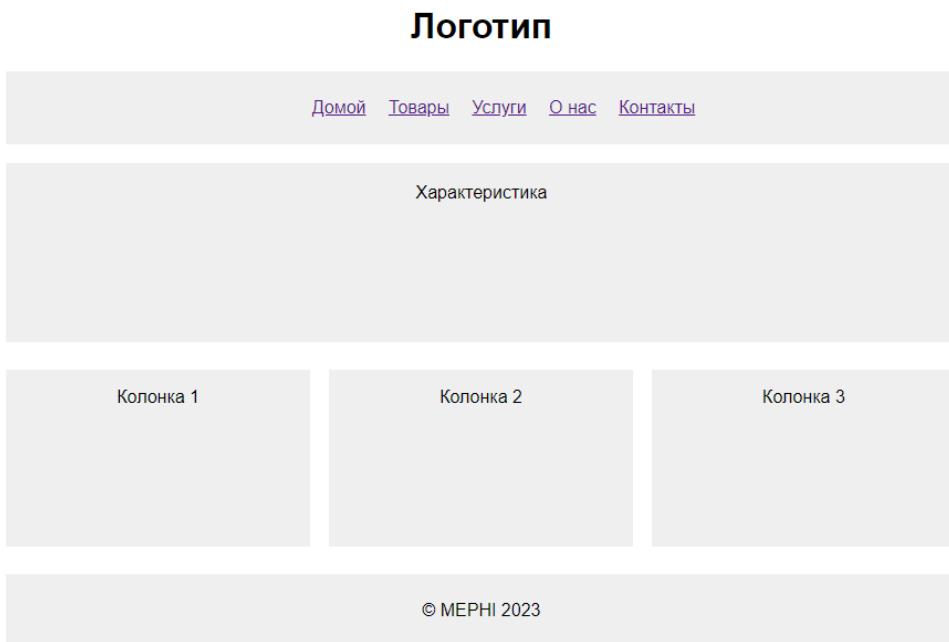


Рис. 11 – Референсное изображение для Задания №3

#### Выполнение

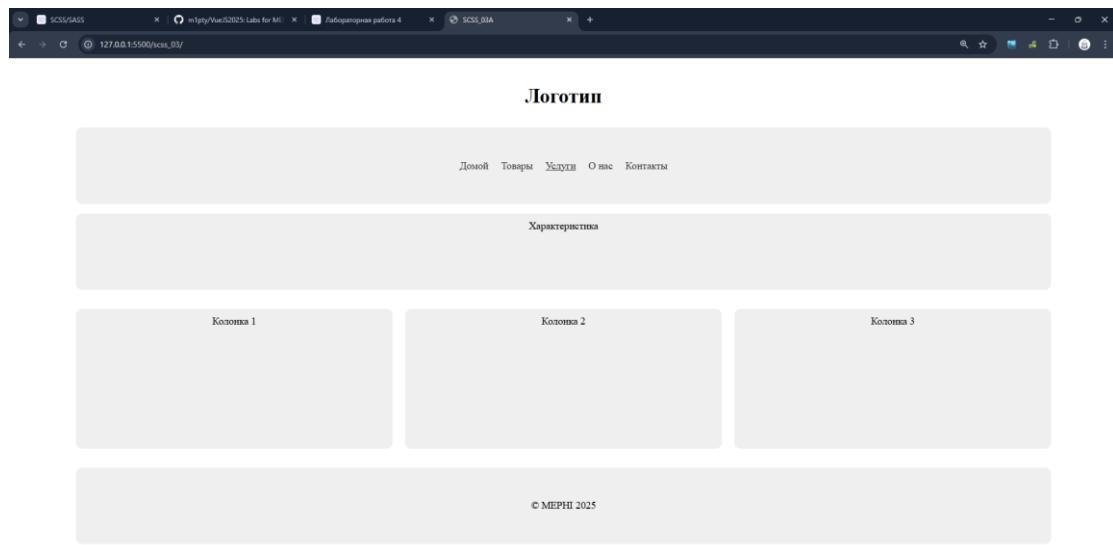


Рис. 12 – Отображение созданной веб-страницы

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a project structure under "LAB4" containing "patterns", "scss\_01-02", "scss\_03", "css", and "index.html".
- Active File:** "index.html" is open in the editor.
- Code Content:**

```
scss_03 > index.html > <!DOCTYPE html>
1   <html lang="ru">
2     <head>
3       <meta charset="UTF-8">
4       <meta name="viewport" content="width=device-width, initial-scale=1.0">
5       <title>SCSS_03A</title>
6       <link rel="stylesheet" href="css/styles.css">
7     </head>
8   <body>
9     <div class="header">
10       <h1>Логотип</h1>
11     </div>
12
13     <div class="container">
14       <div class="field field--wide nav-links">
15         <a href="https://example.com" class="nav-link">Домой</a>
16         <a href="https://example.com" class="nav-link">Товары</a>
17         <a href="https://example.com" class="nav-link">Услуги</a>
18         <a href="https://example.com" class="nav-link">Нас</a>
19         <a href="https://example.com" class="nav-link">Контакты</a>
20       </div>
21       <div class="field field--wide">Характеристика</div>
22       <div class="fields-row">
23         <div class="field field--third">Колонка 1</div>
24         <div class="field field--third">Колонка 2</div>
25         <div class="field field--third">Колонка 3</div>
26       </div>
27       <div class="field field--wide--centered">© МЕРГИ 2025</div>
28     </div>
29   </body>
```
- Status Bar:** Shows "Ln 15, Col 50" and other system information.

Рис. 13 – Листинг кода в файле задания «index.html»

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a project structure under "LAB4" containing "patterns", "scss\_01-02", "scss\_03", "css", and "index.html".
- Active File:** "styles.scss" is open in the editor.
- Code Content:**

```
scss_03 > scss > styles.scss > field-styles
1 $block_bg_color: #FFFFFF;
2 $basic_height: 100px;
3
4 @mixin field-styles {
5   background-color: $block_bg_color;
6   border-radius: 10px;
7   height: $basic_height;
8   margin: [
9     top: 10px; bottom: 5px;
10    left: 10px; right: 10px;
11  ]
12   padding: [
13     top: 10px;
14     bottom: 10px;
15   ]
16   text-align: center;
17 }
18
19 @mixin centred-styles {
20   display: flex;
21   align-items: center;
22   justify-content: center;
23   text-align: center;
24 }
25
26 body {
27   margin: 0;
28   padding: 20px;
29   background-color: #fff;
30   justify-content: center;
31 }
32
33 h1 { @include centred-styles; }
34
35 .container {
36   width: 100%;
37   display: flex;
```
- Status Bar:** Shows "Ln 9, Col 29" and other system information.

Рис. 14 – Листинг кода в файле задания «scss/styles.scss»

*Рис. 15 – Листинг кода в файле задания «scss/styles.scss»*

Рис. 16 – Листинг кода в файле задания «scss/styles.scss»

В рамках выполнения данного задания были использованы 2 mixin-а:

- **field-styles**, задающий базовые настройки отображения для серых блоков-полей;
  - **centred-styles**, центрирующий объекты и текст.

## 1.4 Задание №4. Изучение синтаксисов. Сравнение SCSS/SASS

### Задание

Перепишите код из предыдущих заданий на синтаксис SASS.

### Выполнение

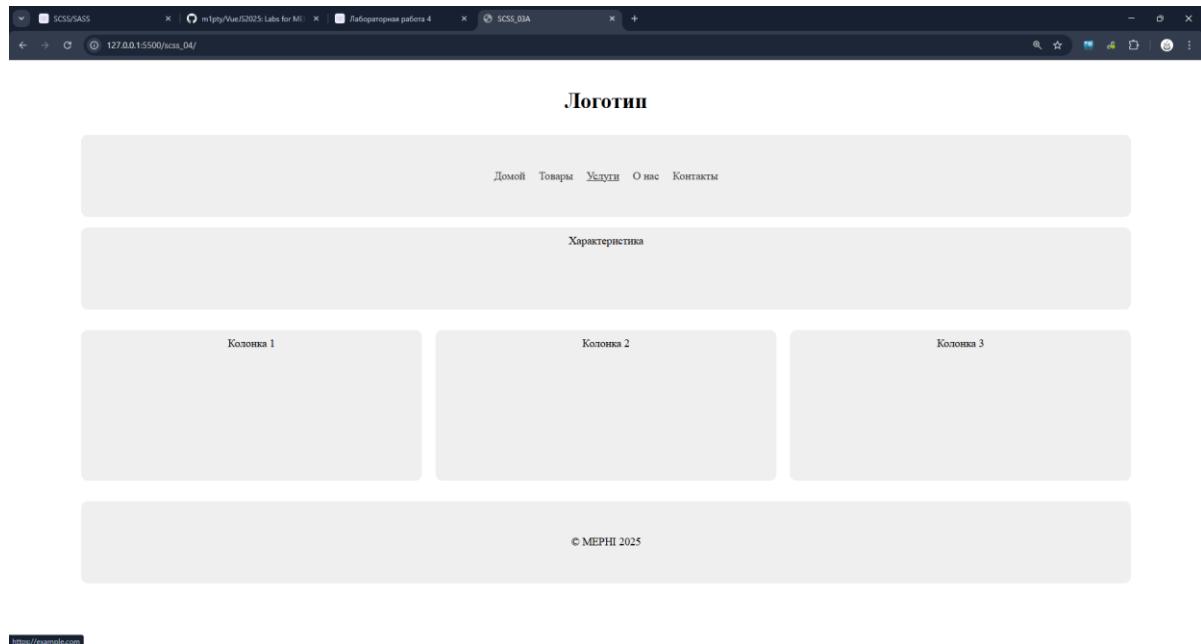


Рис. 16 – Отображение созданной веб-страницы, аналогичной Рис. 12 (SCSS)

```
File Edit Selection View Go Run ... lab4
EXPLORER styles.sass 2 # styles.css
LAB4
  > patterns
  > scss_01-02
  > scss_03
    > css
    > sass
    > index.html
  > scss_04
    > css
    > sass
      > styles.sass 2
        > index.html
        ~$b4_report.docx
        ~-WRL0971.tmp
        lab4_report.docx
        > OUTLINE
        > TIMELINE
        > Live Share
scss_04 > sass > styles.sass > ...
1  $block_bg_color: #FFFFFF
2  $basic_height: 100px
3
4  @mixin field-styles
5    background-color: $block_bg_color
6    border-radius: 10px
7    height: $basic_height
8    margin:
9      top: 10px
10     bottom: 5px
11     left: 10px
12     right: 10px
13    padding:
14      top: 10px
15      bottom: 10px
16      text-align: center
17
18  @mixin centred-styles
19    display: flex
20    align-items: center
21    justify-content: center
22    text-align: center
23
24  body
25    margin: 0
26    padding: 20px
27    background-color: #ffff
28    justify-content: center
29
30  h1
```

Рис. 17 – Листинг кода в файле задания «sass/styles.sass»

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "LAB4" with files like "patterns", "scss\_01-02", "scss\_03", "index.html", "scss\_04", "css", "sass", and "styles.css".
- Editor:** The main pane displays the content of "styles.sass".
- Status Bar:** Shows "Ln 43, Col 11" and "Port: 5500".

```
File Edit Selection View Go Run Terminal Help <- > lab4
EXPLORER ... # styles.css
LAB4
> patterns
> scss_01-02
> scss_03
> css
> sass
index.html
> scss_04
> css
> sass
styles.css 2
index.html
-$14_report.docx
-WRL0971.bmp
lab4_report.docx
Ln 43, Col 11 Spaces: 2 UTF-8 LF SCSS Port: 5500
```

```
28 scss_04 > sass > # styles.css ...
29 justify-content: center
30 h1
31 @include centred-styles
32 .container
33 width: 100%
34 display: flex
35 flex-direction: column
36 align-items: center
37
38 .header
39 @include centred-styles
40 width: 100%
41
42 .nav-links
43 @include centred-styles
44 gap: 20px
45 .nav-link
46 color: #333
47 text-decoration: none
48 &:hover
49 | text-decoration: underline
50
51 .field
52 @include field-styles
53
54 &--wide
55 width: 90%
56
57 &--centered
58 @include centred-styles
59
60 &--wide--centered
61 @extend .field--wide
62 @extend .field--centered
63
64
65 &--third
66 width: 100%
67 max-width: 33%
68 height: $basic_height * 2
69 flex: 1
70
71 // нулевой отступ от левого края в ряду
72 &:first-child
73 | margin-left: 0
74
75 // нулевой отступ от правого края в ряду
76 &:last-child
77 | margin-right: 0
78
79 .fields-row
80 margin: 15px 0px
81 width: 90%
82 display: flex
83 justify-content: space-between
```

Рис. 18 – Листинг кода в файле задания «sass/styles.sass»

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "LAB4" with files like "patterns", "scss\_01-02", "scss\_03", "index.html", "scss\_04", "css", "sass", and "styles.css".
- Editor:** The main pane displays the content of "styles.sass".
- Status Bar:** Shows "Ln 43, Col 11" and "Port: 5500".

```
File Edit Selection View Go Run Terminal Help <- > lab4
EXPLORER ... # styles.css
LAB4
> patterns
> scss_01-02
> scss_03
> css
> sass
index.html
> scss_04
> css
> sass
styles.css 2
index.html
-$14_report.docx
-WRL0971.bmp
lab4_report.docx
Ln 43, Col 11 Spaces: 2 UTF-8 LF SCSS Port: 5500
```

```
49 &:hover
50 | text-decoration: underline
51
52 .field
53 @include field-styles
54
55 &--wide
56 width: 90%
57
58 &--centered
59 @include centred-styles
60
61 &--wide--centered
62 @extend .field--wide
63 @extend .field--centered
64
65 &--third
66 width: 100%
67 max-width: 33%
68 height: $basic_height * 2
69 flex: 1
70
71 // нулевой отступ от левого края в ряду
72 &:first-child
73 | margin-left: 0
74
75 // нулевой отступ от правого края в ряду
76 &:last-child
77 | margin-right: 0
78
79 .fields-row
80 margin: 15px 0px
81 width: 90%
82 display: flex
83 justify-content: space-between
```

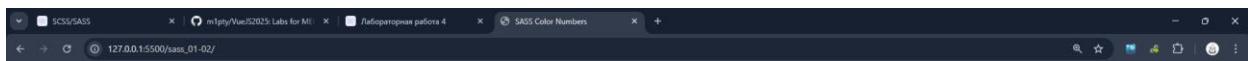
Рис. 19 – Листинг кода в файле задания «sass/styles.sass»

The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows a project structure under "LAB4" with files like "index.html", "sass\_03", "scss\_01-02", and "scss\_03".
- Code Editor (Center):** Displays the "styles.sass" file content:

```
sass_01-02 > sass > styles.sass > ...
1   .number-list
2   @for $i from 1 through 100
3     .num-#$i
4       color: rgb($i, 256 - $i, 0)
5       display: inline-block
6       margin: 2px
```
- Status Bar (Bottom):** Shows "Ln 6, Col 18" and "SCSS".
- Right Sidebar:** A "Layout Controls" panel is open, showing checked items: "Menu Bar", "Command Center", "Navigation Controls", "Layout Controls", and "Copilot Controls".

Рис. 20 – Листинг кода в файле задания «sass/styles.sass» для аналога Задания №2



## Lab 4 «SCSS/SASS» Task 4

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
98 99 100
```

Рис. 21 – Отображение созданной веб-страницы, аналогичной Рис. 5 (SCSS)

В рамках выполнения данного задания код Заданий №1-3 был переписан на синтаксис SASS, в результате чего вложенность теперь регулируется отступами, отсутствуют символы фигурных скобок и «;» на конце строк.

## 1.5 Задание №5. Синтаксис Less

### Задание

Перепишите код из предыдущих заданий на синтаксис LESS.

### Выполнение

Для использования LESS пропишем команду установки:

```
npm install -g less
```

 (1.5.1)

Для компиляции будем использовать следующую команду:

```
lessc .\less_03\less\styles.less .\less_03\css\styles.css
```

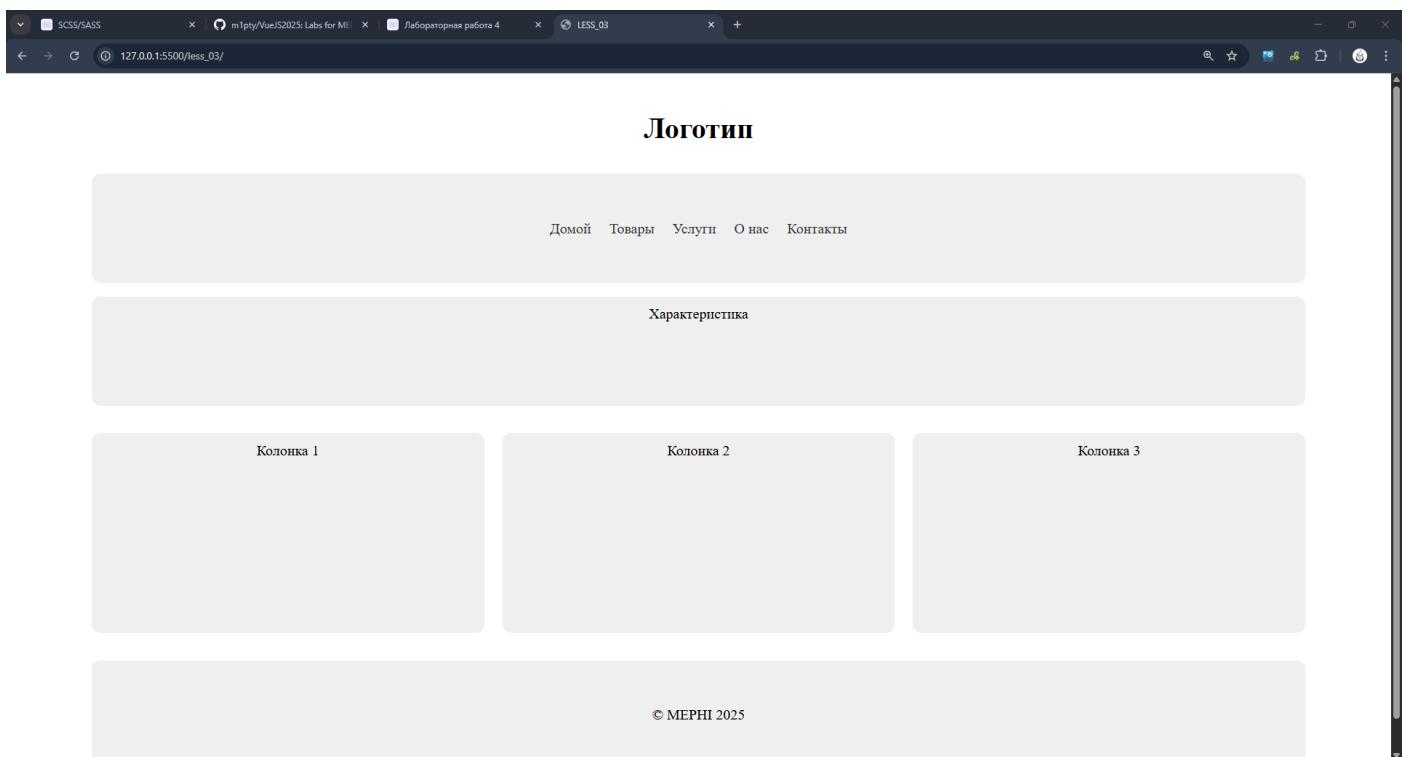
 (1.5.2)

Рис. 22 – Отображение созданной веб-страницы, аналогичной Рис. 12 (SCSS)

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows project files including `LAB4`, `less_01-02`, `less_03`, `css`, `less`, `styles.less`, `index.html`, `patterns`, `sass_01-02`, `sass_03`, `scss_01-02`, `scss_03`, `css`, `scss`, `styles.scss`, `index.html`, `-$b4_report.docx`, and `lab4_report.docx`.
- Editor (Center):** The `styles.less` file is open, displaying LESS code. The code includes a `.field-styles()` function defining background color, border radius, height, and padding/margins. It also includes a `.centred-styles()` function for flexbox styling. The `body` selector sets the background color to white (#fff). The `h1` selector applies the `.centred-styles()` function. The `.container` selector uses flexbox properties like width, display, flex-direction, and align-items.
- Bottom Status Bar:** Shows file paths (`index.html`), line number (Ln 13, Col 24), spaces (Spaces: 4), encoding (UTF-8), and other status information.

Рис. 23 – Листинг кода в файле задания «less/styles.less»

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a project structure with files like `index.html`, `styles.less`, and several `.less` and `.css` files under `LAB4`.
- Editor (Top):** The title bar says "lab4". The left sidebar shows the file `styles.less` is open.
- Editor (Bottom):** The right pane displays the generated CSS code from `styles.less`, which includes styles for `.header`, `.nav-links`, `.nav-link`, and `.field` with various width and flex properties.

Рис. 24 – Листинг кода в файле задания «less/styles.less»

```

less_03 > less > styles.less > .field-styles
45 .nav-links {
55 }
56
57 // добавляем миксины для включения
58 .make-wide() { width: 90%; }
59 .make-centered() { .centred-styles(); }
60
61 .field {
62   .field-styles();
63
64   &--wide { .make-wide(); }
65   &--centered { .make-centered(); }
66   &--wide--centered {
67     .make-wide();
68     .make-centered();
69   }
70   &--third {
71     width: 100%;
72     max-width: 33%;
73     height: @basic_height * 2;
74     flex: 1;
75
76     // нулевой отступ от левого края в ряду
77     &:first-child { margin-left: 0; }
78     // нулевой отступ от правого края в ряду
79     &:last-child { margin-right: 0; }
80   }
81 }
82
83 .fields-row {
84   margin: 15px 0px;
85   width: 90%;
86   display: flex;
87   justify-content: space-between;
88 }

```

Рис. 25 – Листинг кода в файле задания «less/styles.less»



## Lab 4 «SCSS/SASS» Tasks 1, 2

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79  
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Рис. 26 – Отображение созданной веб-страницы, аналогичной Рис. 5 (SCSS)

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar displays a project structure under 'LAB4' with files like 'index.html', 'less\_01-02', 'less', 'styles.less', 'sass\_01-02', 'sass\_03', 'scss\_01-02', 'scss\_03', and 'index.html'. The main editor area shows the content of 'styles.less'. The code uses LESS syntax, including a recursive mixin named '.generate-numbers' that generates a list of numbers from 1 to n. The code is color-coded for readability.

```
.number-list {
    .generate-numbers(100);

    .generate-numbers(@n, @i: 1) when (@i <= @n) {
        .num-@{i} [
            color: rgb(@i, (256 - @i), 0);
            display: inline-block;
            margin: 2px;

            .generate-numbers(@n, (@i + 1));
        ]
    }
}
```

Рис. 27 – Листинг кода в файле задания «less/styles.less»

В рамках выполнения данного задания конструкция `for` была заменена на рекурсивный `mixin` ввиду отсутствия `for`-альтернативы внутри LESS. Также наследование двух дочерних элементов визуала класса в один было произведено также через `mixin`-ы.

## 2 Лабораторная «TypeScript»

Данная лабораторная работа направлена на изучение и закрепление основных возможностей языка TypeScript, ориентированных на строгую типизацию, объектно-ориентированное программирование и разработку масштабируемого кода.

### 2.1 Задание №1. Настройка среды

#### Задание

Установите TypeScript и запустите компиляцию в реальном времени из ts файла в js файл (так, чтобы при любом изменении в исходном ts файле выполнялась перекомпиляция в js).

#### Выполнение

Установим TypeScript с помощью команды ниже:

```
npm install -g typescript
```

(2.1.1)

Создадим в директории задания конфигурационный файл с помощью команды:

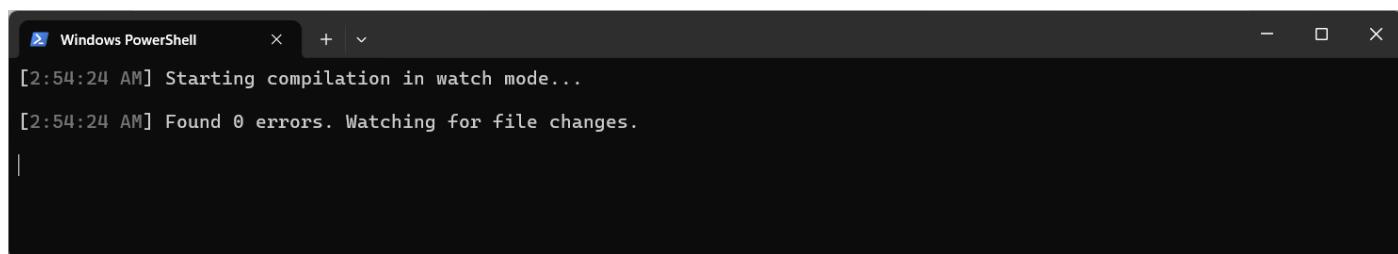
```
tsc --init
```

(2.1.2)

Запустим наблюдение и автоматическую компиляцию ts-файлов в js-файлы с флагом, упреждающим компиляцию файла в случае обнаружения ошибок:

```
tsc --watch --noEmitOnError
```

(2.1.3)



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command "tsc --watch --noEmitOnError" was run, resulting in the following output:

```
[2:54:24 AM] Starting compilation in watch mode...
[2:54:24 AM] Found 0 errors. Watching for file changes.
```

Рис. 28 – Успешный запуск наблюдения за ts-файлами проекта

## 2.2 Задание №2. Создание и типизация класса. IUser

### Задание

Создайте класс User с полями name, age и методом hello(). Этот метод должен выводить в консоль “Hi! My name is <name>. And I am <age> years old.” (вместо <name> и <age> должны выводиться name и age пользователя). С помощью интерфейсов TS типизируйте данный класс (То есть создайте отдельный тип для экземпляров этого класса).

### Выполнение

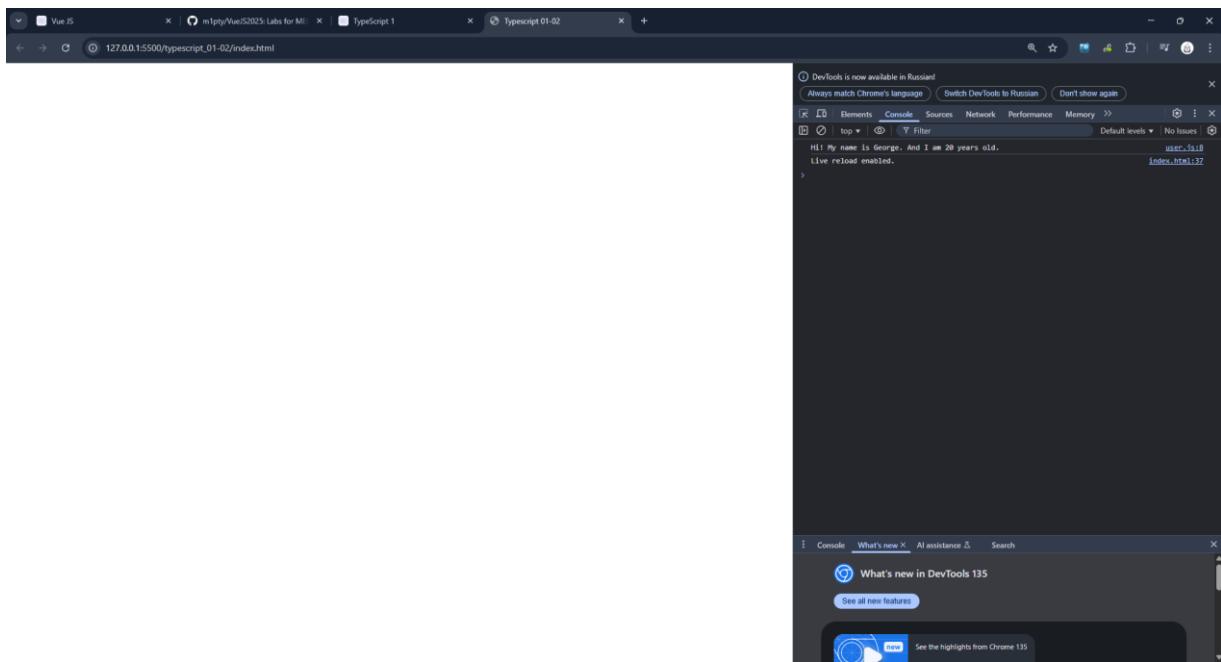


Рис. 29 – Веб-страница, отображающая в консоли вывод метода экземпляра класса

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a folder structure under "LAB4" containing files like "less\_01-02", "sass\_01-02", "typescript\_01-02", "ts", "user.js", "user.ts", "index.html", "tsconfig.json", and two ".docx" files.
- Editor (Center):** The "user.ts" tab is active, displaying the following TypeScript code:

```
1  interface IUser {  
2      name: string;  
3      age: number;  
4      hello(): void;  
5  }  
6  
7  class User implements IUser {  
8      constructor(public name: string, public age: number) {}  
9      hello(): void {  
10         console.log(`Hi! My name is ${this.name}. And I am ${this.age} years old.`);  
11     }  
12 }  
13  
14 const user = new User("George", 20);  
15 user.hello();
```
- Bottom Status Bar:** Shows "Ln 15, Col 14" and "TypeScript".

Рис. 30 – Листинг кода ts-файла

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Same folder structure as in Figure 30.
- Editor (Center):** The "user.js" tab is active, displaying the generated JavaScript code:

```
1  "use strict";  
2  class User {  
3      constructor(name, age) {  
4          this.name = name;  
5          this.age = age;  
6      }  
7      hello() {  
8          console.log(`Hi! My name is ${this.name}. And I am ${this.age} years old.`);  
9      }  
10 }  
11 const user = new User("George", 20);  
12 user.hello();  
13
```
- Bottom Status Bar:** Shows "Ln 1, Col 1" and "JavaScript".

Рис. 31 – Листинг кода сгенерированного js-файла

## 2.3 Задание №3. Типизация с использованием псевдонимов. Type

### Задание

Типируйте класс из предыдущего задания с помощью псевдонимов типов.

### Выполнение

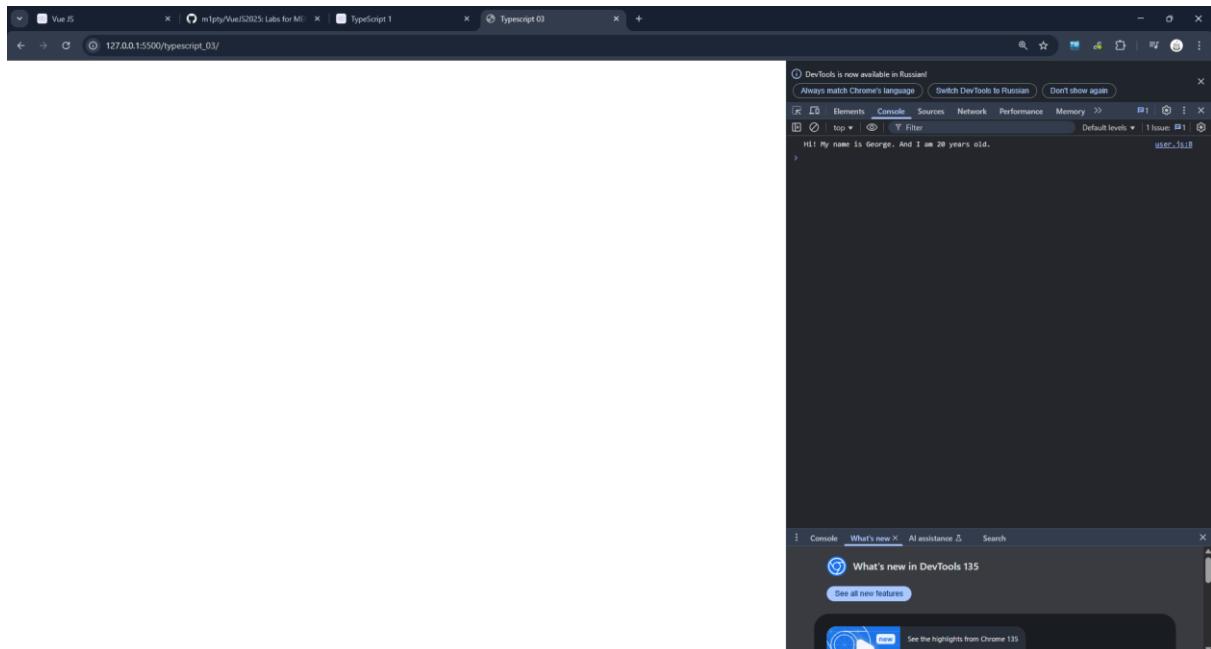


Рис. 32 – Веб-страница, отображающая в консоли вывод метода экземпляра класса

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows a project structure with files like 'user.ts', 'user.js', 'index.html', and 'tsconfig.json'. The main editor pane displays the following TypeScript code:type UserType = {  
 name: string;  
 age: number;  
 hello(): void;  
};  
  
class User implements UserType {  
 constructor(public name: string, public age: number) {}  
  
 hello(): void {  
 console.log(`Hi! My name is \${this.name}. And I am \${this.age} years old.`);  
 }  
}  
  
const user = new User("George", 20);  
user.hello();The status bar at the bottom indicates the file is 'user.ts', has 4 lines and 19 columns, and is using UTF-8 encoding.

Рис. 33 – Листинг кода ts-файла

```
typescript_03 > ts > JS user.js > ...
1  'use strict';
2  class User {
3      construct (property) User.name: any
4          this.name = name;
5          this.age = age;
6      }
7      hello() {
8          console.log(`Hi! My name is ${this.name}. And I am ${this.age} years old.`);
9      }
10 }
11 const user = new User("George", 20);
12 user.hello();
13
```

Рис. 33 – Листинг кода сгенерированного ls-файла

## 2.4 Задание №4. Перегрузка функций

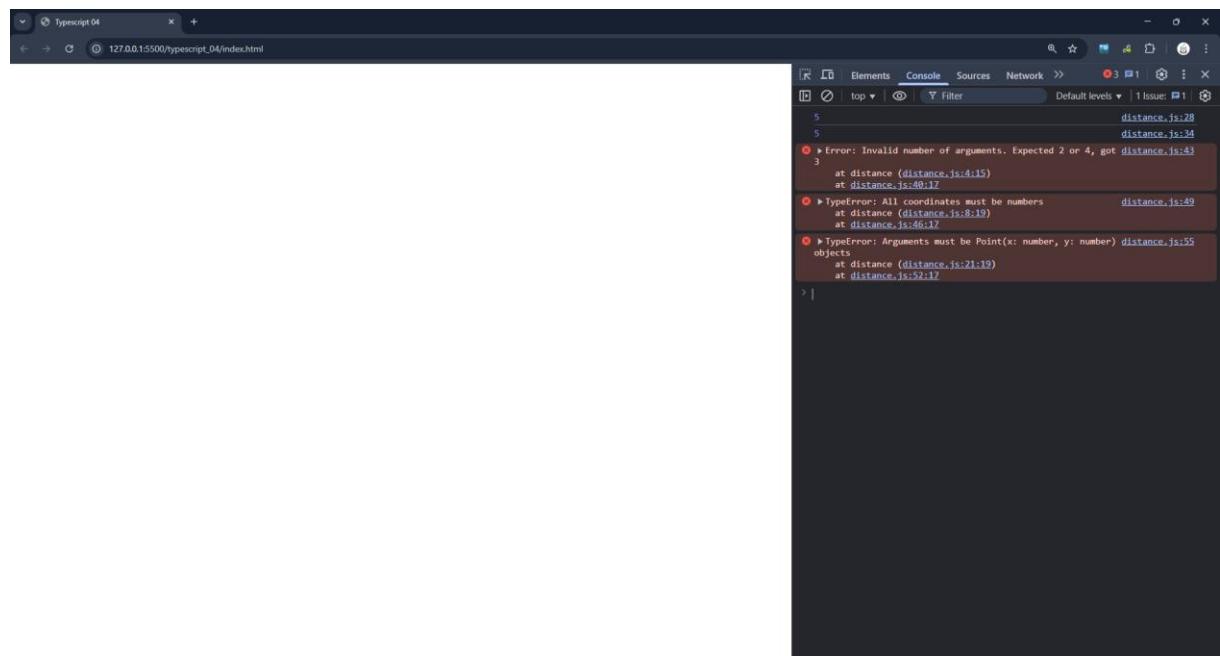
### Задание

Реализуйте с помощью TS перегруженную функцию `distance`, определяющую расстояние между двумя точками. Сделайте так, чтобы функцию можно было вызывать двумя способами:

- `distance(x1, y1, x2, y2)` – с передачей координат двух точек;
- `distance(p1, p2)` – с передачей точек (причем каждая точка – это объект с полями `x` и `y`, например, `{x:10, y:20}`).

Все типы (в том числе объектов точек) должны быть явно указаны (запрещено использовать `any` в явной или неявной форме).

### Выполнение



The screenshot shows a browser developer tools console window titled "Console". It displays several error messages from a TypeScript file named "distance.js". The errors are:

- Error: Invalid number of arguments. Expected 2 or 4, got 5 at distance (distance.js:4:15) at distance.js:43
- TypeError: All coordinates must be numbers at distance (distance.js:8:19) at distance.js:46:17
- TypeError: Arguments must be Point(x: number, y: number) objects at distance (distance.js:21:19) at distance.js:52:17

Рис. 34 – Консольный вывод результата работы программы  
(2 корректных формы работы и 3 обработки исключений)

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "LAB4" containing files like "less\_01-02", "less\_03", "sass\_01-02", "sass\_03", "scss\_01-02", "scss\_03", "typescript\_01-02", "typescript\_03", and "typescript\_04".
- Editor:** Displays the file "distance.ts" with the following code:

```
typescript_04 > ts > TS distance.ts > distance
1 type Point = { x: number, y: number };
2
3 function distance(x1: number, y1: number, x2: number, y2: number): number;
4 function distance(p1: Point, p2: Point): number;
5
6 function distance(...args: unknown[]): number {
7     if (args.length !== 2 && args.length !== 4) {
8         throw new Error(`Invalid number of arguments. Expected 2 or 4, got ${args.length}`);
9     }
10
11     if (args.length === 4) {
12         if (!args.every((arg): arg is number => typeof arg === 'number')) {
13             throw new TypeError('All coordinates must be numbers');
14         }
15
16         const [x1, y1, x2, y2] = args as [number, number, number, number];
17         return Math.hypot(x2 - x1, y2 - y1);
18     } else {
19         const isValidPoint = (p: unknown): p is Point =>
20             typeof p === 'object' &&
21             p !== null &&
22             'x' in p &&
23             'y' in p &&
24             typeof p.x === 'number' &&
25             typeof p.y === 'number';
26
27         if (!args.every(isValidPoint)) {
28             throw new TypeError('Arguments must be Point(x: number, y: number) objects');
29         }
30     }
}
```

Bottom status bar: Ln 6, Col 48 | Spaces: 2 | UTF-8 | LF | TypeScript | Port: 5500

Рис. 35 – Листинг кода файла «ts/distance.ts»

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "LAB4" containing files like "less\_01-02", "less\_03", "sass\_01-02", "sass\_03", "scss\_01-02", "scss\_03", "typescript\_01-02", "typescript\_03", and "typescript\_04".
- Editor:** Displays the file "distance.ts" with the following code:

```
typescript_04 > ts > TS distance.ts > distance
6 function distance(...args: unknown[]): number {
7     if (args.length !== 2 && args.length !== 4) {
8         throw new TypeError(`Arguments must be Point(x: number, y: number) objects`);
9     }
10
11     const [p1, p2] = args as [Point, Point];
12     return Math.hypot(p2.x - p1.x, p2.y - p1.y);
13 }
14
15 try {
16     console.log(distance(0, 0, 3, 4));
17 } catch (e) { console.error(e); }
18
19 try {
20     console.log(distance({x: 0, y: 0}, {x: 3, y: 4}));
21 } catch (e) { console.error(e); }
22
23 try {
24     console.log(distance(0, 1, 2));
25 } catch (e) { console.error(e); }
26
27 try {
28     console.log(distance(0, '0', 1, 2));
29 } catch (e) { console.error(e); }
30
31 try {
32     console.log(distance({}, {}));
33 } catch (e) { console.error(e); }
```

Bottom status bar: Ln 6, Col 48 | Spaces: 2 | UTF-8 | LF | TypeScript | Port: 5500

Рис. 36 – Листинг кода файла «ts/distance.ts»

## 2.5 Задание №5. Бинарное дерево

### Задание

С помощью TS написать класс, реализующий бинарное дерево. Предусмотреть методы поиска, вставки, удаления, изменения элемента и определения высоты дерева. Использовать настройки strict и noImplicitAny. Избегать утверждения типов (операторы as и non-null assertion).

### Выполнение

The screenshot shows a browser developer tools console with two tree structures displayed.

**Tree 1 (Initial State):**

```
tree.show()
The tree has been created with inserted values of
3,1,5,2,4,6,0,10,3.5,2.7,19,11.2
> tree.show()
  19
  / \
  10  11.2
 / \   \
 6   5   3.5
 / \   / \
 4   3   2.7
 / \   / \
 3   2   2.7
 / \   / \
 1   0   0
```

**Tree 2 (After Operations):**

```
< undefined
> tree.insert(9)
< undefined
> tree.delete(3)
< undefined
> tree.find(2.7)
< true
> tree.replace(11.2, 22.1)
< undefined
> tree.show()
  22.1
  / \
  19  9
  / \
  10  6
  / \
  5  4
  / \
  3.5  3
  / \
  2.7  2
  / \
  2  0
```

**Console Log:**

```
< undefined
> tree.get_height()
< 6
```

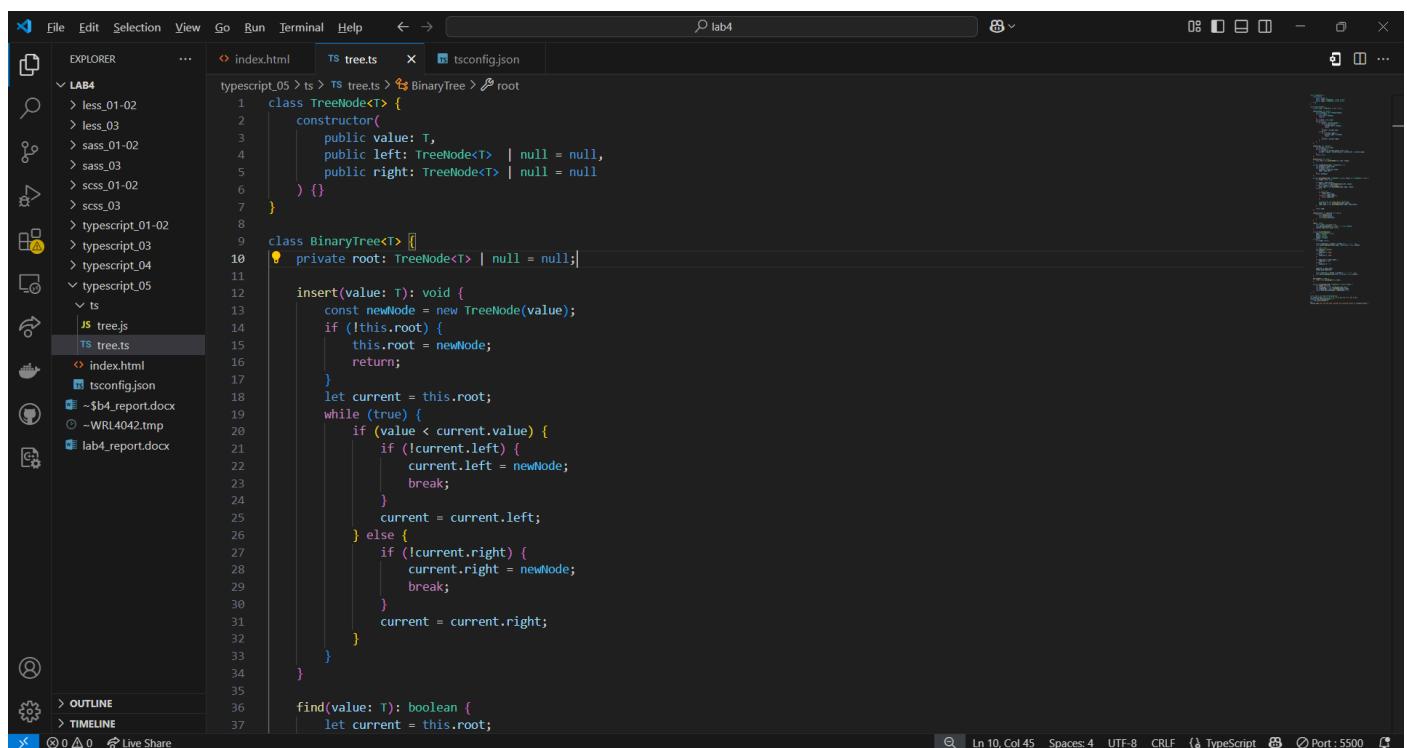
Рис. 37 – Результат выполнения методов класса *BinaryTree*, реализующего бинарное дерево

В данном задании были реализованы публичные методы шаблонного класса *BinaryTree*<T>:

- `insert(value: T): void` – вставка значения;
- `find(value: T): boolean` – поиск по значению;
- `delete(value: T): void` – удаление по значению;
- `replace(value: T, newvalue: T): void` – замена по значению;
- `get_height(): number` – определение высоты дерева;
- `show(): void` – вывод дерева.

Также были созданы приватные методы, используемые в публичных:

- `private calcHeight(node: TreeNode<T> | null): number` – подсчёт высоты;
- `private buildTreeOutput(node: TreeNode<T> | null, prefix: string, isLeft: boolean, output: string[])` : `void` – построение построчного вывода дерева в выходной массив;
- `private deleteNode(node: TreeNode<T> | null, value: T): TreeNode<T> | null` – удаление узла дерева с возвратом узла-замены;
- `private findMinValue(node: TreeNode<T>): T` – поиск минимального значения в поддереве;



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "LAB4" containing several files: less\_01-02, less\_03, sass\_01-02, sass\_03, scss\_01-02, scss\_03, typescript\_01-02, typescript\_03, typescript\_04, typescript\_05, and two files named "tree.js" and "tree.ts".
- Editor:** The "tree.ts" file is open, displaying the following TypeScript code:

```
1  class TreeNode<T> {
2      constructor(
3          public value: T,
4          public left: TreeNode<T> | null = null,
5          public right: TreeNode<T> | null = null
6      ) {}
7
8
9  class BinaryTree<T> {
10     private root: TreeNode<T> | null = null;
11
12     insert(value: T): void {
13         const newNode = new TreeNode(value);
14         if (!this.root) {
15             this.root = newNode;
16             return;
17         }
18         let current = this.root;
19         while (true) {
20             if (value < current.value) {
21                 if (!current.left) {
22                     current.left = newNode;
23                     break;
24                 }
25                 current = current.left;
26             } else {
27                 if (!current.right) {
28                     current.right = newNode;
29                     break;
30                 }
31                 current = current.right;
32             }
33         }
34     }
35
36     find(value: T): boolean {
37         let current = this.root;
```

The code defines a `TreeNode` class with properties for `value`, `left`, and `right`. It also defines a `BinaryTree` class with a private `root` and methods for inserting a new node and finding a specific value.

Рис. 38 – Листинг кода файла «ts/tree.ts»

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "LAB4" containing files like "less\_01-02", "sass\_01-02", "typescript\_01-02", and "tree.ts".
- Editor:** The main editor tab is "tree.ts", showing TypeScript code for a binary tree. The code includes methods for finding a value, deleting a value, and finding the minimum value.
- Status Bar:** Shows "Ln 10, Col 45" and other status information.

```
typescript_05 > ts > ts tree.ts > BinaryTree > root
9 class BinaryTree<T> {
10     ...
11 }
12
13     }
14
15     find(value: T): boolean {
16         let current = this.root;
17         while (current) {
18             if (value === current.value) return true;
19             current = value < current.value ? current.left : current.right;
20         }
21         return false;
22     }
23
24     delete(value: T): void {
25         this.root = this.deleteNode(this.root, value);
26     }
27
28     private findMinValue(node: TreeNode<T>): T {
29         let minValue = node.value;
30         while (node.left) {
31             minValue = node.left.value;
32             node = node.left;
33         }
34         return minValue;
35     }
36
37     private deleteNode(node: TreeNode<T> | null, value: T): TreeNode<T> | null {
38         if (!node) return null;
39
40         if (value < node.value) {
41             node.left = this.deleteNode(node.left, value);
42         } else if (value > node.value) {
43             node.right = this.deleteNode(node.right, value);
44         } else {
45
46             // 1 ПОТОМКИ
47             if (!node.left) {
48                 return node.right;
49             } else if (!node.right) {
50                 return node.left;
51             }
52
53             // 2 ПОТОМКА, "жертва" в правом поддереве
54             node.value = this.findMinValue(node.right);
55             node.right = this.deleteNode(node.right, node.value);
56
57         }
58
59         return node;
60     }
61
62     replace(value: T, newValue: T): void {
63         if (this.find(value)) {
64             this.delete(value);
65             this.insert(newValue);
66         }
67     }
68
69     show(): void {
70         const output: string[] = [];
71
72         ...
73
74         return output;
75     }
76
77     ...
78
79     ...
80
81     ...
82
83     ...
84
85     ...
86
87     ...
88
89     ...
90 }
```

Рис. 39 – Листинг кода файла «ts/tree.ts»

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "LAB4" containing files like "less\_01-02", "sass\_01-02", "typescript\_01-02", and "tree.ts".
- Editor:** The main editor tab is "tree.ts", showing TypeScript code for a binary tree. The code includes methods for finding a value, deleting a value, and finding the minimum value.
- Status Bar:** Shows "Ln 10, Col 45" and other status information.

```
typescript_05 > ts > ts tree.ts > BinaryTree > root
137     }
138
139     private findMinValue(node: TreeNode<T>): T {
140         ...
141     }
142
143     private deleteNode(node: TreeNode<T> | null, value: T): TreeNode<T> | null {
144         if (!node) return null;
145
146         if (value < node.value) {
147             node.left = this.deleteNode(node.left, value);
148         } else if (value > node.value) {
149             node.right = this.deleteNode(node.right, value);
150         } else {
151
152             // 1 ПОТОМКИ
153             if (!node.left) {
154                 return node.right;
155             } else if (!node.right) {
156                 return node.left;
157             }
158
159             // 2 ПОТОМКА, "жертва" в правом поддереве
160             node.value = this.findMinValue(node.right);
161             node.right = this.deleteNode(node.right, node.value);
162
163         }
164
165         return node;
166     }
167
168     replace(value: T, newValue: T): void {
169         if (this.find(value)) {
170             this.delete(value);
171             this.insert(newValue);
172         }
173     }
174
175     show(): void {
176         const output: string[] = [];
177
178         ...
179
180         return output;
181     }
182
183     ...
184
185     ...
186
187     ...
188
189     ...
190 }
```

Рис. 40 – Листинг кода файла «ts/tree.ts»

```

class Binarytree<T> {
    replace(value: T, newvalue: T): void {
        ...
    }

    show(): void {
        const output: string[] = [];
        this.buildTreeOutput(this.root, "", true, output);
        console.log(output.join("\n"));
    }

    private buildTreeOutput(
        node: TreeNode<T> | null,
        prefix: string,
        isLeft: boolean,
        output: string[]
    ): void {
        if (!node) return;

        const rightPrefix = prefix + (isLeft ? "| " : " ");
        this.buildTreeOutput(node.right, rightPrefix, false, output);

        // текущийузел
        let nodeLine = prefix;
        if (isLeft) {
            nodeLine += "└—";
        } else {
            nodeLine += "┌—";
        }

        if (node.left || node.right) {
            nodeLine += "-";
        } else {
            nodeLine += " ";
        }

        nodeLine += node.value;
        output.push(nodeLine);
    }
}

```

Рис. 41 – Листинг кода файла «ts/tree.ts»

```

class Binarytree<T> {
    private buildTreeOutput(
        node: TreeNode<T> | null,
        prefix: string,
        isLeft: boolean,
        output: string[]
    ): void {
        if (!node) return;

        const rightPrefix = prefix + (isLeft ? "| " : " ");
        this.buildTreeOutput(node.right, rightPrefix, false, output);

        let nodeLine = prefix;
        if (isLeft) {
            nodeLine += "└—";
        } else {
            nodeLine += "┌—";
        }

        nodeLine += node.value;
        output.push(nodeLine);
    }

    get_height(): number {
        return this.calcheight(this.root);
    }

    private calcheight(node: TreeNode<T> | null): number {
        if (!node) return 0;
        let leftHeight = this.calcheight(node.left);
        let rightHeight = this.calcheight(node.right);
        return Math.max(leftHeight, rightHeight) + 1;
    }

    const tree = new Binarytree<number>();
    let insert_array = [3, 1, 5, 2, 4, 6, 0, 10, 3.5, 2.7, 19, 11.2];
    insert_array.forEach(element => {
        tree.insert(element);
    });
    console.log(`The tree has been created with inserted values of ${insert_array}`);
}

```

Рис. 42 – Листинг кода файла «ts/tree.ts»

```
typescript_05 > tsconfig.json > {} compilerOptions
      "compilerOptions": {
        /* Type Checking */
        "strict": true,
        "noImplicitAny": true,
        // "strictNullChecks": true,
        // "strictFunctionTypes": true,
        // "strictBindCallApply": true,
        // "strictPropertyInitialization": true,
        // "strictBuiltInIteratorReturn": true,
        // "noImplicitThis": true,
        // "useUnknownInCatchVariables": true,
        // "alwaysStrict": true,
        // "noUnusedLocals": true,
        // "noUnusedParameters": true,
        // "exactOptionalPropertyTypes": true,
        // "noImplicitReturns": true,
        // "noFallthroughCasesInSwitch": true,
        // "noUncheckedIndexedAccess": true,
        // "noImplicitOverride": true,
        // "noPropertyAccessFromIndexSignature": true,
        // "allowUnusedLabels": true,
        // "allowUnreachableCode": true,
        // "allowDynamicallyDefinedFunctions": true
      }
    
```

Рис. 43 – Применение правил «strict» и «noImplicitAny» в конфигурационном файле

## 2.6 Задание №6. Реализация паттернов Adapter, Strategy, Observer

### Задание

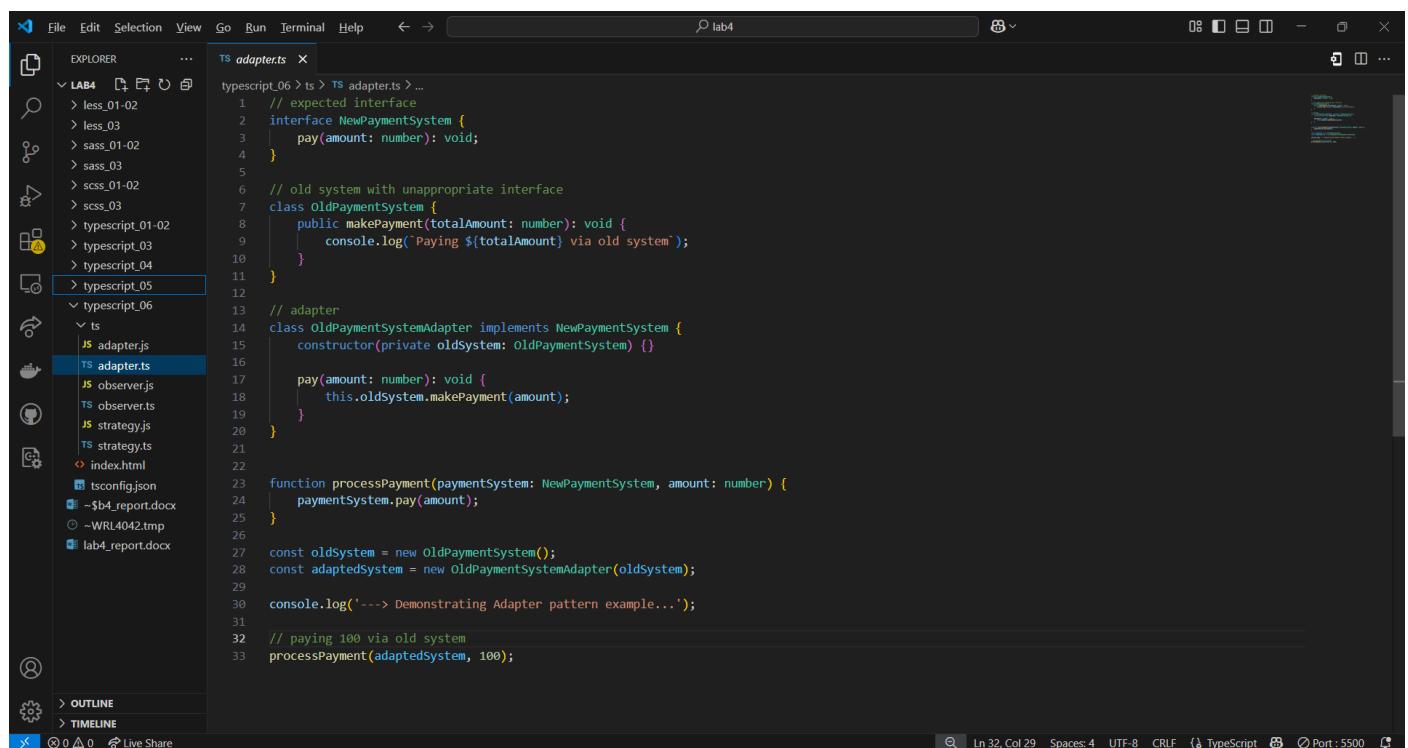
Реализовать с использованием TypeScript паттерны Adapter, Strategy, Observer.

Привести примеры использования собственных классов.

### Выполнение

Паттерн **Adapter** позволяет классам с несовместимыми сигнатурами методов взаимодействовать между собой. Он инкапсулирует адаптируемый класс, предоставляя новый интерфейс.

В приведённом примере рассматривается адаптация платежей через старую банковскую систему к платежам через новую систему.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure with folders like LAB4, less\_01-02, less\_03, sass\_01-02, sass\_03, scss\_01-02, sess\_03, typescript\_01-02, typescript\_03, typescript\_04, typescript\_05, and typescript\_06. Inside typescript\_06, there are files adapter.ts, observer.js, observer.ts, strategy.js, and strategy.ts.
- Code Editor:** Displays the content of `adapter.ts`. The code defines an interface `NewPaymentSystem` with a method `pay(amount: number): void`. It then defines a class `OldPaymentSystem` that implements this interface, containing a `makePayment(totalAmount: number): void` method that logs a message. Finally, it defines an adapter class `OldPaymentsystemAdapter` that implements `NewPaymentSystem`, constructor-injecting an `oldSystem` instance. The `pay(amount: number)` method calls the `makePayment` method on the injected system. A function `processPayment` is also defined to demonstrate the usage of the adapter.
- Status Bar:** Shows the current line (Ln 32), column (Col 29), spaces used (Spaces: 4), encoding (UTF-8), line endings (CRLF), file type (TypeScript), port (Port: 5500), and other status information.

Рис. 44 – Листинг кода файла «ts/adapter.ts»

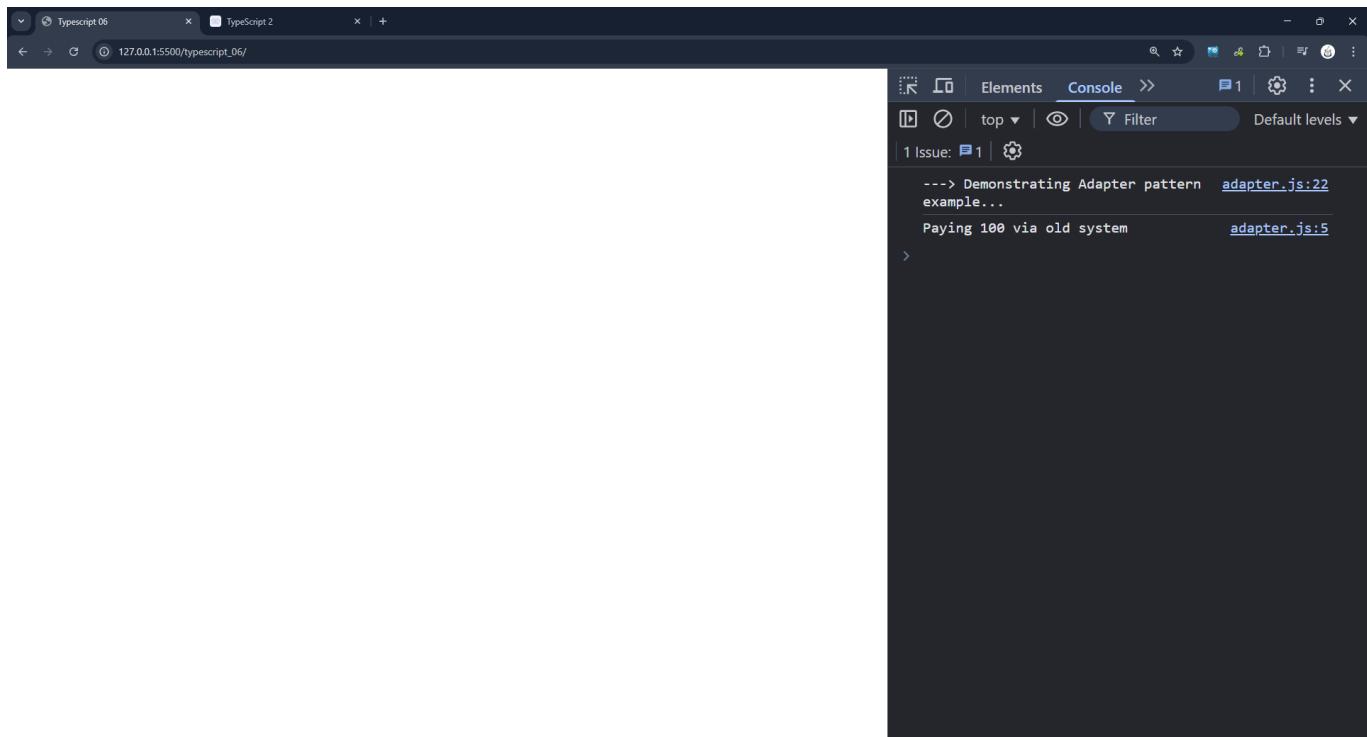


Рис. 45 – Демонстрация работы паттерна Adapter

Паттерн **Strategy** позволяет определять стратегию выполнения программы во время исполнения кода. Он определяет схожие алгоритмы и помещает их в разные классы, после чего алгоритмы можно взаимозаменять в рамках выполнения программы.

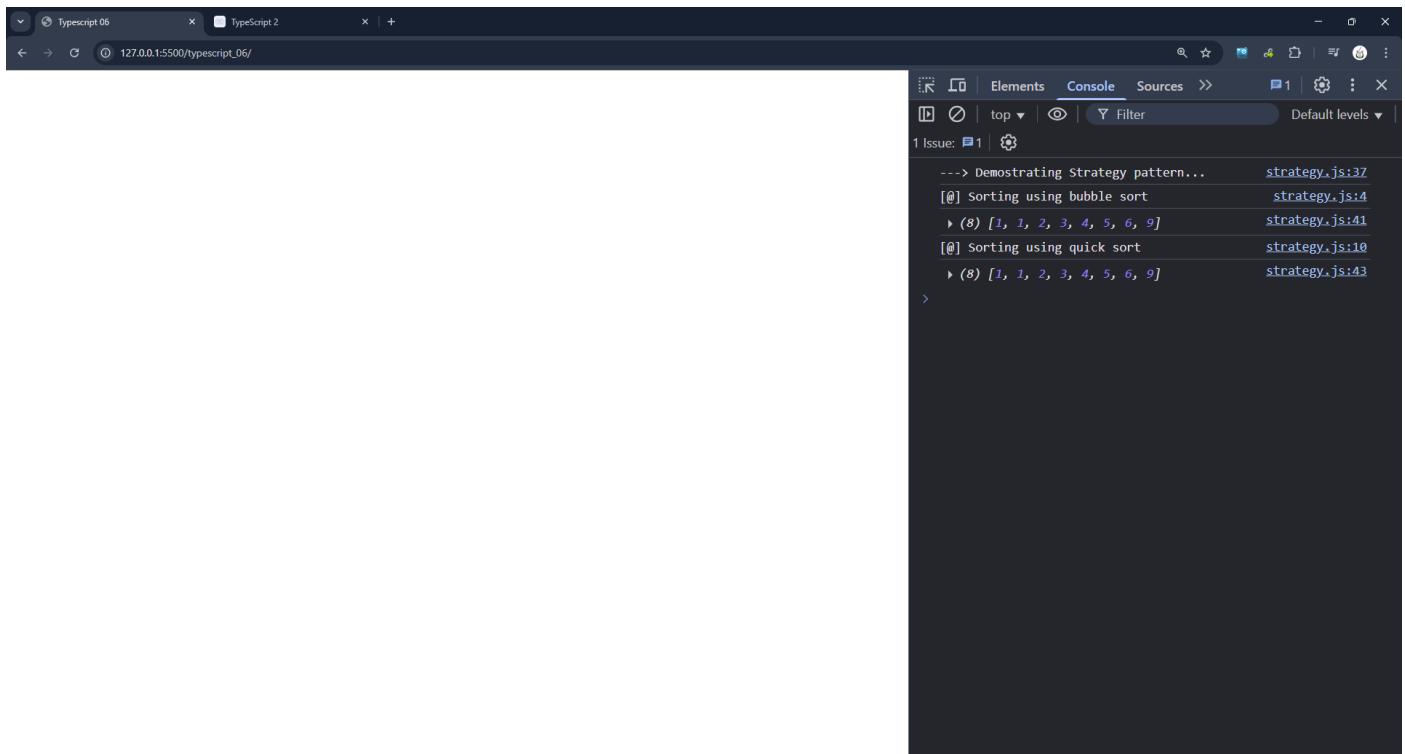


Рис. 46 – Демонстрация работы паттерна *Strategy*

```

File Edit Selection View Go Run Terminal Help < > lab4
EXPLORER strategy.ts index.html
typescript_06 > ts > strategy.ts > BubbleSortStrategy
1 // strategy interface
2 interface SortingStrategy {
3     sort(data: number[]): number[];
4 }
5
6 class BubbleSortStrategy implements SortingStrategy {
7     sort(data: number[]): number[] {
8         console.log("[@] Sorting using bubble sort");
9         return [...data].sort((a, b) => a - b);
10    }
11 }
12
13 class QuickSortStrategy implements SortingStrategy {
14     sort(data: number[]): number[] {
15         console.log("[@] Sorting using quick sort");
16         return this.quickSort([...data]);
17     }
18
19     private quickSort(data: number[]): number[] {
20         if (data.length <= 1) return data;
21
22         const pivot = data[0];
23         const left = [];
24         const right = [];
25
26         for (let i = 1; i < data.length; i++) {
27             if (data[i] < pivot) {
28                 left.push(data[i]);
29             } else {
30                 right.push(data[i]);
31             }
32         }
33
34         return [...this.quickSort(left), pivot, ...this.quickSort(right)];
35     }
36 }
37

```

Рис. 47 – Листинг кода файла «ts/strategy.ts»

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a project structure under 'LAB4' with files like 'less\_01-02', 'less\_03', 'sass\_01-02', 'sass\_03', 'scss\_01-02', 'scss\_03', 'typescript\_01-02', 'typescript\_03', 'typescript\_04', 'typescript\_05', 'typescript\_06', 'tsconfig.json', and 'tsconfig.json'. The file 'strategy.ts' is currently selected.
- Editor (Center):** Displays the content of 'strategy.ts'. The code defines a class 'Sorter' with a private field 'strategy' of type 'SortingStrategy'. It has a constructor that takes a 'strategy' parameter and sets it. A method 'setStrategy' also sets the 'strategy'. The 'sort' method returns the sorted array. The code then demonstrates the pattern by creating a 'Sorter' instance with a 'BubbleSortStrategy', then changing it to a 'QuickSortStrategy'.
- Bottom Status Bar:** Shows the current file is 'strategy.ts', line 7, column 54, with 4 spaces, using UTF-8 encoding, and is a TypeScript file. It also shows the port is 5500.

Рис. 48 – Листинг кода файла «ts/strategy.ts»

Паттерн **Observer** позволяет одним объектам подписываться на события других объектов и реагировать на них.

В приведённом примере рассматривается наблюдение за состоянием реактора NuclearReactor объектов ControlRoomDisplay, DataLogger и EmergencySystem.

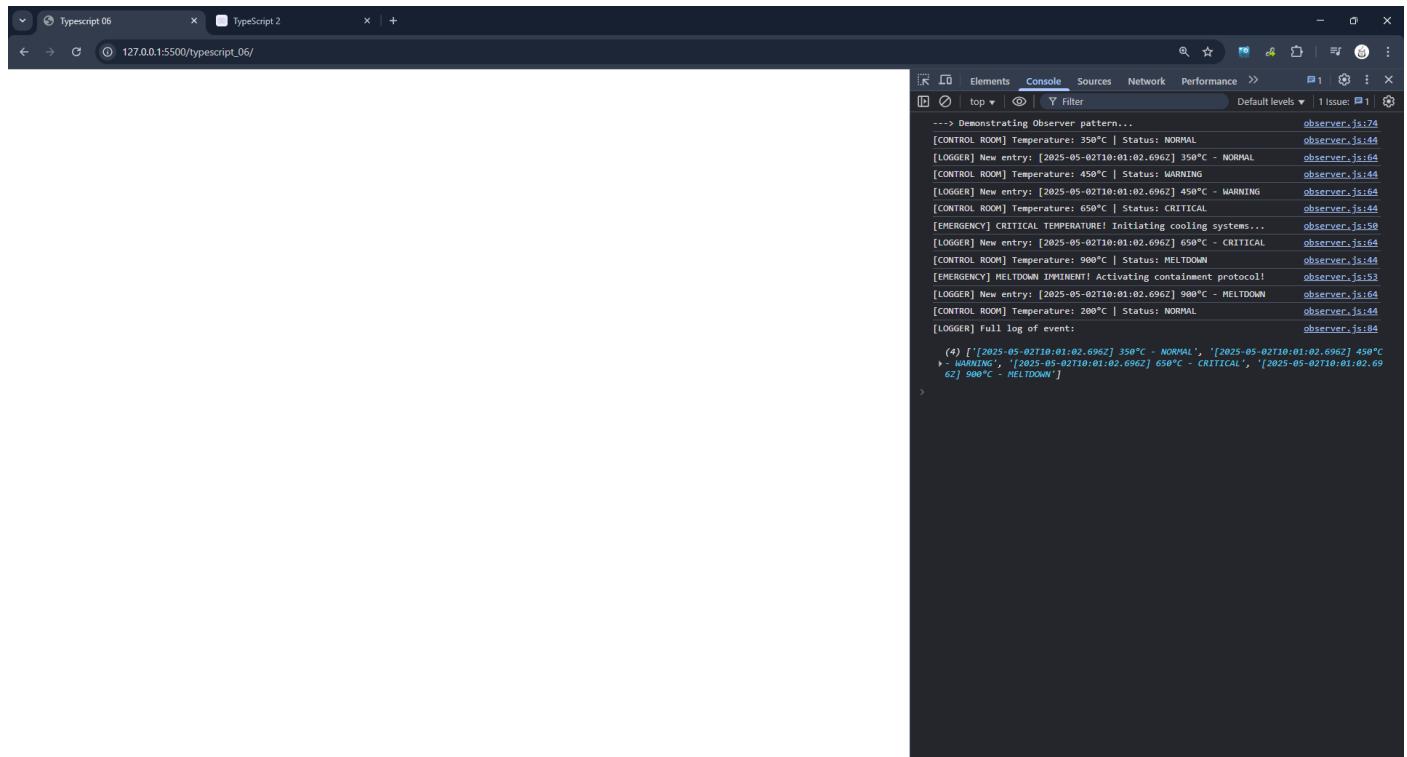


Рис. 49 – Демонстрация работы паттерна Observer

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows the project structure with files like 'index.html', 'observer.ts', 'strategy.ts', and 'adapter.ts'. The main editor area displays the 'observer.ts' file. The code implements the Observer pattern with an interface 'ReactorObserver' and a class 'NuclearReactor' that acts as a subject. It includes methods for setting temperature, updating status, attaching observers, and detaching them. The code uses TypeScript syntax with interfaces, classes, and methods.

Рис. 50 – Листинг кода файла «ts/observer.ts»

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "LAB4" containing various files like "less\_01-02", "sass\_01-02", "typescript\_01-02", etc., and subfolders "ts" and "JS".
- Code Editor:** The active file is "ts/observer.ts". The code implements the Observer pattern with classes like NuclearReactor, ControlRoomDisplay, EmergencySystem, and DataLogger.
- Status Bar:** Shows "Ln 48, Col 8" and other settings like "Spaces: 4", "UTF-8", "CRLF", "TypeScript", and "Port: 5500".

```
typescript_06 > ts > ts/observer.ts > ControlRoomDisplay
11 class NuclearReactor implements ReactorSubject {
12     public detach(observer: ReactorObserver): void {
13         this.observers.splice(index, 1);
14     }
15     public notify(): void {
16         for (const observer of this.observers) {
17             observer.update(this.temperature, this.status);
18         }
19     }
20 }
21
22 class ControlRoomDisplay implements ReactorObserver {
23     update(temperature: number, status: string): void {
24         console.log(`[CONTROL ROOM] Temperature: ${temperature}°C | status: ${status}`);
25     }
26 }
27
28 class EmergencySystem implements ReactorObserver {
29     update(temperature: number, status: string): void {
30         if (status === "CRITICAL") {
31             console.log("[EMERGENCY] CRITICAL TEMPERATURE! Initiating cooling systems...");
32         } else if (status === "MELTDOWN") {
33             console.log("[EMERGENCY] MELTDOWN IMMINENT! Activating containment protocol!");
34         }
35     }
36 }
37
38 class DataLogger implements ReactorObserver {
39     private log: string[] = [];
40     update(temperature: number, status: string): void {
41         const entry = `${new Date().toISOString()} ${temperature}°C - ${status}`;
42         this.log.push(entry);
43         console.log(`[LOGGER] New entry: ${entry}`);
44     }
45 }
```

Рис. 51 – Листинг кода файла «ts/observer.ts»

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "LAB4" containing various files like "less\_01-02", "sass\_01-02", "typescript\_01-02", etc., and subfolders "ts" and "JS".
- Code Editor:** The active file is "ts/observer.ts". The code demonstrates the Observer pattern with a main reactor and multiple observers.
- Status Bar:** Shows "Ln 48, Col 8" and other settings like "Spaces: 4", "UTF-8", "CRLF", "TypeScript", and "Port: 5500".

```
typescript_06 > ts > ts/observer.ts > ControlRoomDisplay
64 class DataLogger implements ReactorObserver {
65     private log: string[] = [];
66
67     update(temperature: number, status: string): void {
68         const entry = `${new Date().toISOString()} ${temperature}°C - ${status}`;
69         this.log.push(entry);
70         console.log(`[LOGGER] New entry: ${entry}`);
71     }
72
73     getLog(): string[] {
74         return this.log;
75     }
76 }
77
78 const reactor = new NuclearReactor();
79 const controlRoom = new ControlRoomDisplay();
80 const emergency = new EmergencySystem();
81 const logger = new DataLogger();
82
83 console.log(`--- Demonstrating Observer pattern...`)
84
85 reactor.attach(controlRoom);
86 reactor.attach(emergency);
87 reactor.attach(logger);
88
89 reactor.setTemperature(350);
90 reactor.setTemperature(450);
91 reactor.setTemperature(650);
92 reactor.setTemperature(900);
93
94 reactor.detach(logger);
95 reactor.setTemperature(200); // только controlRoom и emergency получат обновление
96
97 console.log(`[LOGGER] Full log of event:\n${logger.getLog()}`);
98
```

Рис. 52 – Листинг кода файла «ts/observer.ts»

## **Заключение**

В рамках выполнения данной лабораторной работы были изучены и применены на практике две технологии разработки Frontend: препроцессоры CSS (SCSS/SASS и LESS) и язык программирования TypeScript. Для каждой технологии в рамках поставленных задач произведены следующие действия:

### **SCSS/SASS:**

- **Настройка окружения:** Установлены Node.js, npm и препроцессор SASS, настроена автоматическая компиляция SCSS/SASS в CSS.
- **Динамические стили:** Использованы циклы SCSS для генерации уникальных стилей для элементов, что продемонстрировало возможности динамического создания CSS-кода.
- **Вложенность и миксины:** Применены вложенные правила и миксины для создания структурированных и повторно используемых стилей, что упростило поддержку кода.
- **Сравнение синтаксисов:** Код переписан с SCSS на SASS и LESS, что позволило сравнить особенности синтаксисов этих препроцессоров, включая различия в форматировании и возможностях.

### **TypeScript:**

- **Настройка среды:** Установлен TypeScript, настроена автоматическая компиляция TS в JS с использованием режима наблюдения.
- **Типизация:** Изучены интерфейсы и псевдонимы типов (type) для строгой типизации классов и объектов.
- **Перегрузка функций:** Реализована перегруженная функция для расчёта расстояния между точками, что продемонстрировало гибкость TypeScript в работе с разными форматами входных данных.

– **Бинарное дерево:** Создан класс для работы с бинарным деревом, включая методы вставки, поиска, удаления и определения высоты, с соблюдением строгой типизации.

– **Паттерны проектирования:** Реализованы паттерны Adapter, Strategy и Observer.