МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)

БГТУ.СМК-Ф-4.2-К5-01

Факультет И (шифр) Информационные и управляющие системы (наименование)

Кафедра И5 (шифр) Информационные системы и программные технологии (наименование)

Дисциплина Программирование на языке высокого уровня

ПРАКТИЧЕСКАЯ РАБОТА №%NUMBER%

на тему

%TITLE%

Выполнил студент группы *И-582*

Махнев П.С.
Фамилия И.О.

РУКОВОДИТЕЛЬ

Спирин Д.О. _____
Фамилия И.О. Подпись

Оценка _____

« ____ » _____ 2019 г.

САНКТ-ПЕТЕРБУРГ
2019 г.

1) Все элементы храняться в статическом массиве базового класса окна;

2) Все индентификаторы объектов уникальны;

3) Для добавления элемента используется статическая функция `addObject`, которая принимает единственным параметром указатель на добавляемый объект. В случае если объект с таким идентификатором существует, то объект не добавляется;

4) Для получения объекта используется статическая функция `getElementById`, которая принимает единственным параметром строковый идентификатор. В случае отсутствия элемента с заданным идентификатором возвращает `nullptr`;

5) Для получения набора элементов используется статическая функция `getElementsByClassName`, которая принимает единственным параметром строку содержащую в себе строковый класс, который должны содержать элементы из возвращаемой выборки.

   Для возвращаемого набора существует несколько функций, такие как:
   - `each(callback(Object* caller))` — вызывающая переданную функцию `callback` для каждого из элементов выборки;
   - `addEventListener` — рассмотренна далее.

6) Каждый элемент отслеживает 7 возможных событий:
   - Событие клика по элементу;
   - Событие наведение на элемент;
   - События движения мыши над элементом;
   - Событие нажатия кнопки мыши;
   - Событие отпускания кнопки мыши;
   - Событие попадания мыши на элемент;
   - Событие выхода за пределы элемента.

   Для установки прослушивателя для каждого из событий используется статическая функция `addEventListener`, которая принимает первым параметром строку идентификатор события, для которого ставится прослушиватель,

а вторым функцию, которая будет выполняться каждый раз когда данное событие произойдет.

Для выборки полученной с помощью `getElementsByClassName`, при вызове данной функции, прослушиватель будет назначен для каждого элемента выборки.

7) Для добавления нового дочернего элемента используется функция `append`, так как `mainContainer` является основным контейнером, любые другие элементы добавляются в него. Для добавления сразу нескольких объектов, удобно использовать следующую конструкцию:

```
ParentElement
START_CHILDS

    Container::create(new Container(...)

    NEXT_CHILD

    Container::create(new Container(...)

END_CHILDS;
```

Так же такую форму можно использовать для единовременного добавления объектов с большой глубиной вложенности:

```
ParentElement
START_CHILDS
    Container::create(...)
    START_CHILDS
        Container::create(...)
        START_CHILDS
            Container::create(...)
            NEXT_CHILD
            Container::create(...)
            START_CHILDS
                Container::create(...)
                START_CHILDS
                    Container::create(...)
                END_CHILDS
            END_CHILDS
        END_CHILDS
    END_CHILDS
END_CHILDS;
```

Данная запись является оберткой над функцией `appendSomeObjects`, которя принимает массив объектов для добавления.

8) Каждому классу может соотвествовать свой объект стилей. Все стили хранятся в статическом ассоциативном массиве базового класса окна, в котором ключом является класс, а значением объект стилей. Для добавления нового стиля, используется статическая функция `addStyle`, которая принимает 2 параметра, первым строковый класс, а втором указатель на класс стилей. Например:

```
ObjectStyle* containerStyles = new ObjectStyle;
containerStyles->backgroundColor(0x555555ff);
containerStyles->activeBackgroundColor(0x333333ff);
containerStyles->borderColor(0xffffffff);
containerStyles->hoverBorderColor(0x00ffffff);
Window::addStyle("container", containerStyles);
```

устанавливает стили для класса `container`. Теперь все объекты с классом `container` будут стилизованы в соотвествии с этими стилями.

Все цвета задаются в виде 32 битного беззнакового целого, где первые 8 бит отводятся под красную составляющую, вторые — под зеленую, третьи — синию, а четвертые под альфа-канал.