МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ВятГУ)

Факультет компьютерных и физико-математических наук
Кафедра фундаментальной математики

Выпускная квалификационная работа

по направлению подготовки
02.03.01 Математика и компьютерные науки
Профиль: «Математические основы компьютерных наук»

на тему «Разработка модуля интеграции онлайн-сервиса построения графиков и геометрических чертежей в визуальный онлайн HTML-редактор»

Выполнил: студент МКб-4301-51-00
Мельков Алексей Константинович

_____
(*подпись*)

Научный руководитель:
канд. физ.-матем. наук, доцент
Марков Роман Владимирович

_____
(*подпись*)

Допущено к защите в ГЭК: «____»_____20____ г.
Заведующий кафедрой фундаментальной математики

_____ Е. М. Вечтомов
(*подпись*)

Декан факультета компьютерных и физико-математических наук

_____ Н. А. Бушмелева
(*подпись*)

Киров
2022

Реферат

Мельков Алексей Константинович

Разработка модуля интеграции онлайн-сервиса построения графиков и геометрических чертежей в визуальный онлайн HTML-редактор

ПЗ : Выпускная квалификационная работа, каф. фундаментальной математики; рук. Марков Роман Владимирович. Киров, 2022.

ПЗ 23 с., 0 рис., 0 табл., 2 источников, 1 прил.

<КЛЮЧЕВЫЕ СЛОВА>

Объект разработки: модуль интеграции онлайн-сервиса построения графиков и геометрических чертежей в визуальный онлайн HTML-редактор

Цель: <Переносится цель ВКР дословно>

Методы проведения работы: <Описываются применяемые методы: анализ научной литературы, сравнительный анализ, математическое моделирование, методы математического анализа и т. д.>

Рассматриваются вопросы: связи онлайн-сервиса построения графиков и геометрических чертежей GeoGebra и визуальный онлайн HTML-редактор TinyMCE

Оглавление

Введение

Выпускная квалификационная работа посвящена разработке модуля интеграции онлайн-сервиса построения графиков и геометрических чертежей в визуальный онлайн HTML-редактор

Актуальность работы обусловлена оптимизацией времени и удобности написания статей, вопросов, заданий. Данный модуль позволит интегрировать распространенный онлайн-сервис построения графиков и геометрических чертежей GeoGebra в онлайн HTML-редактор TinyMCE, который присутствует на платформе Moodle. Что добавит возможность составления новых тестов, лекций, практических заданий для обучения студентов.

Объектом исследования является изучение связи программ TinyMCE, GeoGebra, платформы Moodle. Для этого потребовалось рассмотреть их API.

Предмет исследования —

Цель работы: написание плагина для платформ Moodle, который расширит возможности TinyMCE добавлением новой кнопки, которая сможет открыть сервис построения графиков GeoGebra.

Для достижения поставленной цели сформулированы следующие **задачи:**

1. Изучение основ программирования на языке JavaScript
2. Изучение API GeoGebra, TinyMCE
3. Узнать как создается плагин для расширения TinyMCE на плафторме Moodle
4. Создание плагина

Теоретическая значимость работы состоит в следующем...

Практическая значимость работы состоит в следующем...

В целом работа носит **теоретический (или прикладной, или практический)** характер.

Структура работы. Выпускная квалификационная работа, общим объемом 23 стр., состоит из введения, двух глав, заключения, библиографического списка.

Первая глава посвящена теоритической части

Вторая глава посвящена практической части

В заключении представлены основные результаты дипломной работы.

В библиографический список включено (кол-во) источников.

Результаты работы **апробированы** (в докладах..., статьях ..., внедрены...)

ГЛАВА 1

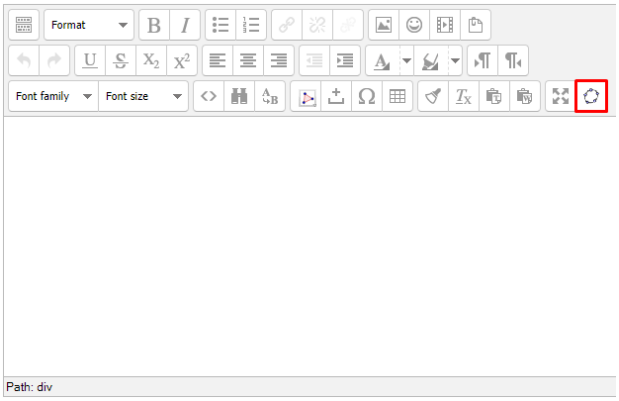
Теоретическая часть

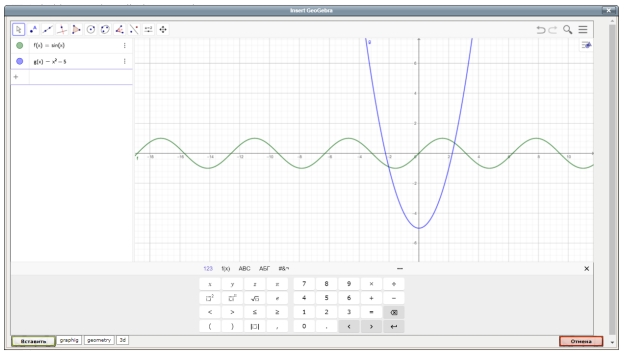
1.1 Анализ готовых решений

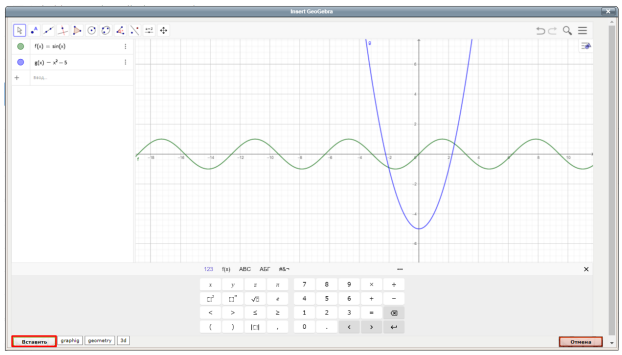
Существует несколько способов вставки графиков в текстовый редактор в Moodle. Первый способ представлен самой GeoGebra. Они предлагают с своего сайта построить необходимые графики, затем нажать комбинацию клавиш, что позволит скопировать HTML код. После этого в редакторе TinyMCE есть кнопка редактирования HTML кода, при ее нажатии откроется окно, куда надо вставить скопированные данные.

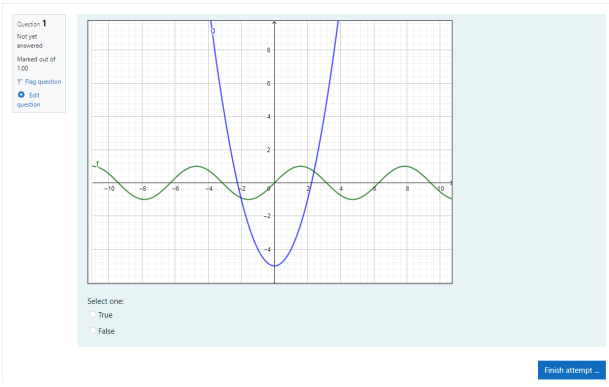
1.2 Описание своего решения

В Moodle представлены три онлайн-редактора. Atto, TinyMCE, обычная область. Обычная область - текстовое пространтсво без кнопок, его нельзя расширять. Поэтому осталось выбрать из двух. Так как, отличий от них не много, я решил выбрать TinyMCE, потому что он показался мне более красивым. Среди онлайн сервисов построения графиков и геометрических чертежей была выбрана GeoGebra, так как она популярна и проста для использования. В итоге, предлагается расширить возможности редактора TinyMCE с помощью разработки собственной кнопки, которая при нажатии открывает окно GeoGebra. В этом окне можно построить графики, после чего нажать на внутреннюю кнопку вставки. В результате, все данные перенесутся в текстовую область. Полученный график будет виден при просмотре страницы:

1. 
Format
Font family
Font size
Path: div

2. 

3. 

4. 
Question 1
Not yet answered
Marked out of 1.00
Flag question
Edit question
Select one:
True
False
Finish attempt ...

1.3 Необходимые знания

1.3.1 основы JavaScript

JavaScript - это язык программирования. Поддерживает объектно-ориентированныйи стили.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Переменные. Они бывают разных типов:

— Number: любое число, включая десятичные дроби

— String: любая группа символов, заключенная в одинарные кавычки: '...' или двойные кавычки "..."

— Boolean: этот тип данных имеет только два возможных значения: либо true, либо false. Полезно думать о логических значениях как о переключателях включения и выключения или как об ответах на вопрос “да” или “нет”

— Null: этот тип данных представляет намеренное отсутствие значения и представлен ключевым словом null

— Undefined: этот тип данных обозначается ключевым словом undefined. Он также представляет отсутствие значения, хотя его использование отличается от null. Отличие в том, что Undefined означает, что данное значение не существует.

— Object: коллекция связанных данных

Свойства. Когда вы вводите новый фрагмент данных в программу JavaScript, браузер сохраняет его как экземпляр типа данных. Все типы данных имеют доступ к определенным свойствам, которые передаются каждому экземпляру. Например, каждый экземпляр string имеет свойство length, в котором хранится количество символов в этой строке. Вы можете получить информацию о свойстве, добавив строку с точкой и именем свойства, к примеру:

```
Console.log('Привет'.length);
```

выведет в консоль 6

Методы - это действия, которые мы можем выполнять. Типы данных имеют доступ к определенным методам, которые позволяют нам обрабатывать экземпляры этого типа данных. Чтобы вызвать метод, необходимо после переменной написать точку, его название, парные скобочки (). К примеру,

```
Console.log('привет'.toUpperCase()) ;
```

выведет в консоль ПРИВЕТ

Функции. В программировании часто используется код для многократного выполнения определенной задачи. Вместо того, чтобы переписывать один и тот же код, можно сгруппировать блок кода вместе и связать его с одной задачей, затем можно повторно использовать этот блок кода всякий раз, когда нужно выполнить задачу снова. Мы достигаем этого, создавая функцию. Функция - это повторно используемый блок кода, который группирует последовательность инструкций для выполнения определенной задачи.

Как объявляется функция:

```
function Идентификатор(параметры){
  тело функции;
}
```

Альтернативным способом можно внести функцию в переменную. Тогда синтаксис объявления поменяется следующим образом:

```
var Переменная = (параметры) => {
  тело функции;
}
```

Объекты. Объекты JavaScript нужны для создания более сложных структур данных. По своей сути объекты JavaScript представляют собой контейнеры, хранящие связанные данные и функциональные возможности.

Создание объектов, могут быть назначены переменным точно так же, как и любому типу JavaScript. Для обозначения объекта используются

фигурные скобки {}. Объект можно заполнять неупорядоченными данными. Эти данные организованы в пары ключ-значение. Ключ подобен имени переменной, которое указывает на место в памяти, содержащее значение. Значение ключа может иметь любой тип данных в языке, включая функции или другие объекты.

Мы создаем пару ключ-значение, записывая имя ключа или идентификатор, за которым следует двоеточие, а затем значение. Мы разделяем каждую пару ключ-значение в объектном литерале запятой ,:

```
var Переменная_объекта = {
ключ: 'что-то',
'ключ': 123,
};
```

Есть два способа получить доступ к свойству объекта. Первый, как и любое свойство - через точку . и название самого свойства. Второй способ получить доступ к значению ключа - это использовать обозначение в квадратных скобках, [ ].

Для того, чтобы интегрировать JavaScript код в HTML страницу надо вставить парный тег <script>. Но также можно подключить JS файл с помощью ссылки на него тем же тегом, использованием атрибута src, то есть надо указать где лежит этот файл.

1.3.2 API MOODLE для плагинов.

API (Application programming interface) - это инструмент для взаимодействия нескольких программ. API содержит в себе некие «мостики», позволяющие программе А получить доступ к данным из программы Б или к некоторым ее возможностям. Таким образом, программисты могут расширять функциональность своего продукта и связывать его с чужими разработками. Для создания своего плагина TinyMCE необходимо добавить три php файла, которые обращаются к Moodle.

version.php содержит информацию о версии плагина Moodle

```
<?php
defined('MOODLE_INTERNAL') || (die);
#текущая версия плагина (Дата: ГГГГММДДЧЧ)
$plugin -> version = 2012112900;
#требующиеся версия Moodle (Дата: ГГГГММДДЧЧ)
$plugin -> requires = 2012112900;
#полное имя плагина
$plugin -> component = 'tinymce_ИмяПлагина';
```

lib.php содержит код плагина Moodle, здесь должен находиться по крайней мере класс tinymce_ИмяПлагина с методом update_init_params()

```
<?php
defined('MOODLE_INTERNAL') || (die);

    class tinymce_ИмяПлагина extends editor_tinymce_plugin {
protected $buttons = array ('ИмяПлагина');

    protected function update_init_params(array & $params, context $context,
array $options = null) {

    $this -> add_button_after($params,3,'ИмяПлагина','spellchecker');

    $this ->add_js_plugin($params);

    }
}
```

tinymce_ИмяПлагина.php. Файл обязан содержать запись

```
<?php
$string['pluginname'] = 'ИмяПлагина';
```

где 'pluginname' нельзя изменять.

1.3.3 API Tinymce

Чтобы вставить TinyMCE редактор в HTML страницу надо добавить следующие скрипты
<script src="https://cdn.tiny.cloud/1/no-api-key/tinymce/5/tinymce.min.js"referrerp
Эта строка ссылает нашу страницу на сайт редактора
<script>
tinymce.init({
});
</script>
Этот скрипт запускает TinyMCE. Так как в Moodle уже загружен редактор и я расширяю его возможности, то эти скрипты мне не нужны. В итоге я изменяю параметры init.

1.3.4 API Geogebra

Чтобы на своей странице загрузить GeoGebra нужна добавить несколько скриптов и место куда будет добавляться приложение. Сначала нам нужно, сделать ссылку на источник GeoGebra: <script src="https://www.geogebra.org/apps/deployggb.js»</script>.
Затем создадим контейнер, куда вставим наше приложение: <div id="ggb-element»
Теперь нам нужно задать параметры и открыть окно GeoGebra:
<script>
var params = {"appName": "graphing "width": 800, "height": 600, "showToolBar": true, "showAlgebraInput": true, "showMenuBar": true };
var applet = new GGBApplet(params, true);
window.addEventListener("load function() {
applet.inject('ggb-element');
});
</script>
Разберем параметры:

— appName - тип вычислений. Можно выбрать graphing (построение графиков), geometry (геометрия), 3d.
— width, height - высота и ширина окна в px.
— showToolBar, showAlgebraInput, showMenuBar - отображение инструментов и панелей. Если true - отображаются, если false - скрыты

После мы создаем GGBApplet с указанными параметрами, при загрузке страницы, срабатывает событие добавления GGBApplet в div контейнер с id 'ggb-element'

1.4 Создание плагина для TinyMCE на платформе Moodle

Для создания плагина TinyMCE нужно сгенерировать каталог.

В нем минимум должны находиться:
— Подкатолог /lang
— Подкатолог /pix
— Подкатолог /tinymce
— Файл lib.php
— Файл version.php

/lang предназначен для хранения языковых папок. В моем примере в ней существует подкаталог /en в котором находится файл tinymce_ИмяПлагина.php
Файл обязан содержать запись
<?php
$string['pluginname'] = 'ИмяПлагина';
где 'pluginname' нельзя изменять.

/pix содержит иконку вашего плагина в расширении .png, обычно это 20 × 20 пикселей.

/tinymce оснвоной католог, содержащий js файл. В нем находится скрипт, что будет делать наш плагин.

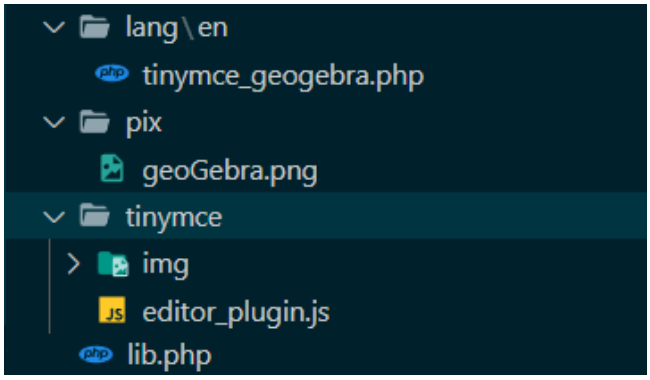
lib.php содержит код плагина Moodle, здесь должен находиться по крайней мере класс tinymce_ИмяПлагина с методом update_init_params()
<?php
defined('MOODLE_INTERNAL') || (die);

```
    class tinymce_ИмяПлагина extends editor_tinymce_plugin {
protected $buttons = array ('ИмяПлагина');

    protected function update_init_params(array & $params, context $context,
array $options = null) {

    $this -> add_button_after($params,3,'ИмяПлагина','spellchecker');

    $this ->add_js_plugin($params);

    }
}
```

version.php содержит информацию о версии плагина Moodle

```
<?php
defined('MOODLE_INTERNAL') || (die);
#текущая версия плагина (Дата: ГГГГММДДЧЧ)
$plugin -> version = 2012112900;
#требующиеся версия Moodle (Дата: ГГГГММДДЧЧ)
$plugin -> requires = 2012112900;
#полное имя плагина
$plugin -> component = 'tinymce_ИмяПлагина';
```

В итоге мы должны получить примерно такой каталог:



ГЛАВА 2

Практическая часть

В этой главе будут рассмотрены файлы плагина с кодом и пояснения к ним.

2.1 editor_plugin.js

Главный файл модуля интеграции, так как именно в нем описывается добавление нестандартной кнопки, а также здесь задается её функция. Этот файл написан на языке JavaScript. Он из себя представляет большую функцию, в которой содержатся два метода: tinymce.create() и tinymce.PluginManager.add().

2.1.1 tinymce.create()

Этим методом создается объекта. Сначала объявляется его идентификатор, затем описывается тело класса.

```
tinymce.create("tinymce.plugins.geogebra", {
  /**
  * @param {tinymce.Editor} ed
  * @param {string} url
  */
  ...
}
```

— @param {tinymce.Editor} ed - тег @param позволяет указать тип, имя и описание параметра функции. В данном случае, параметр ed объявлен типом tinymce.Editor. Это экземпляр переменных редактора, в котором инициализируется плагин.

— @param string url - параметр, который представлен строкой. Абсолютный URL-адрес, по которому находится плагин.

init

```
init: function (ed, url) {
  ed.addCommand("mceGeogebra", function () {
    ed.windowManager.open(
      {
        file: ed.getParam("moodle_plugin_base") + "geogebra/geogebra.
        width: 840 + parseInt(ed.getLang("geogebra.delta_width", 0)),
        height: 480 + parseInt(ed.getLang("geogebra.delta_height", 0)
        inline: 1,
      },
      {
        plugin_url: url,
      }
    );
  });

  ed.addButton("geogebra", {
    title: "GeoGebra Plugin",
    cmd: "mceGeogebra",
    image: url + "/img/geoGebra.gif",
  });
}
```

Ключ объекта. init - функция, содержащая два входных параметра ed, url. В ней вызывается метод addCommand, который аналогичен init, имеет название команды, что она делает. Теперь это функция, а не объект. Именно здесь задается логика плагина.

— ed.windowManager.open(). Открывает диалоговое окно с конфигом, прописанным в параметре функции. Рассмотрим его подробнее. File - абсолютный путь, того что откроет кнопка, при нажатии. В этом случае, открывается php-страница, которая будет рассмотрена ниже. Width, Height - высота, ширина открывающегося окна. Plugin_url - ссылка на плагин.

— ed.addButton(). Добавляет кнопку. Title - текст, который будет отображаться, при наведении на кнопку. Cmd - то, что выполнит кнопка. Image

- ссылка на картинку, которая будет отображаться на кнопке.

getInfo Второй ключ - функция. При вызвое возвращает информацию о плагине.

```
getInfo: function () {
  return {
    longname: "GeoGebra plugin",
    author: "Alexei Melkov",
    authorurl: "alekseymelkov@gmail.com",
    infourl: "alekseymelkov@gmail.com",
    version: "1.061",
  };
},
```

2.1.2 tinymce.PluginManager.add()

Регистрирует собственный плагин в TinyMCE, используя PluginManager. PluginManager.add() принимает строку для идентификатора плагина и объект, содержащую код для инициализации плагина.

```
tinymce.PluginManager.add("geogebra", tinymce.plugins.geogebra);
```

Идентификатор "geogebra а переданный объект tinymce.plugins.geogebra, который был создан выше с помощью метода tinymce.create().

2.2 geogebra.php

Страница, которая открывается при нажатии на кнопку. Она написана на языке PHP. Вначале записан PHP код, который при обработке нажатия, вставится в заголовок диалогового окна.

```
<?php

define('NO_MOODLE_COOKIES', true);
```

```
require('../../../../../config.php');

$PAGE->set_context(context_system::instance());
$PAGE->set_url('/lib/editor/tinymce/plugins/geogebra/geogebra.php');
$PAGE->set_title(get_string('title', 'tinymce_geogebra'));
$PAGE->set_pagelayout('embedded');

$editor = get_texteditor('tinymce');
$plugin = $editor->get_plugin('geogebra');

$PAGE->requires->js(new moodle_url($editor->get_tinymce_base_url() . '
$PAGE->requires->js(new moodle_url($plugin->get_tinymce_file_url('js/d

echo $OUTPUT->header();

?>
```

define - определение константы.
require - включает и выполняет указанный файл.
$PAGE - экземпляр moodle_page, который хранит всю информацию и используется библиотекой вывода $OUTPUT при отображении страницы.
set_context, set_url, set_title, set_pagelayout - по API Moodle выставляются параметры диалогового окна.
После получаем доступ к плагину, чтобы экспортировать нужны файлы.

Потом инициализируется контейнер с GeoGebra и пятью кнопками. Параметры GeoGebra такие же как при описании API, только теперь есть условие, которое обновит приложение, если до этого был сохранен график в текстовом редакторе. Кнопки делятся на меняющие режим GeoGebra и обновляющие диалоговое окно. Обработчики событий этих кнопок, добавлены в отдельный файл dialog.js, кроме нажатия на кнопку отмены. Она использует стандартный метод close класса tinyMCEPopup.

```
<script src="https://www.geogebra.org/apps/deployggb.js"></script>
```

```
<script>
// var str = window.tinyMCE.activeEditor.getContent();
// var appName = 'graphig';
// var begin = str.indexOf('ggbBase64')

// if (begin != -1) {
//   begin = str.indexOf('ggbBase64')+11;
//   var end = str.indexOf("=='};")+2;
//   appName = str.slice(str.indexOf('appName')+10,str.indexOf('width'
//   var code64 = str.slice(begin, end);
//   var params = {
//   appName: ``${appName}``,
//   width: window.screen.width - 100,
//   height: window.screen.height - 150,
//   showToolBar: true,
//   showAlgebraInput: true,
//   showMenuBar: true,
//   ggbBase64: ``${code64}``,
//   };
// }
// else {
var params = {
// appName: ``${appName}``,
appName: 'graphig',
width: window.screen.width - 100,
height: window.screen.height - 150,
showToolBar: true,
showAlgebraInput: true,
showMenuBar: true,
};
// }
var applet = new GGBApplet(params, true);
window.onload = function () {applet.inject('ggb-element');};
```

```
</script>

<div id="ggb-element">
</div>
<div>
<input type="button" id="graphig" name="graphig" value="graphig" oncli
<input type="button" id="geometry" name="geometry" value="geometry" on
<input type="button" id="3d" name="3d" value="3d" onclick="GeoDialog.r
<input type="button" id="insert" name="insert" value="Вставить" onclic
<input type="button" id="cancel" name="cancel" value="Отмена" onclick=
```

Затем добавляется подвал (футер) страницы.

```
<?php
echo $OUTPUT->footer();
```

2.3 dialog.js

В этом файле создаются обработчики событий кнопок. Все они хранятся в объекте GeoDialog.

```
var GeoDialog = {
  insert: function () {
    tinyMCE.activeEditor.setContent("");
    var code = ggbApplet.getBase64();
    var inserted =
      "<p><div id='applet_container'></div>" +
      "<script type='text/javascript' src='https://www.geogebra.org/ap
      "<script type='text/javascript'>var params={appName: 'graphig', 
      ''${code}'' +
      "};var applet = new GGBApplet(params, true);" +
      "window.onload = function () {applet.inject('applet_container');
      "</script></p>";
    tinyMCEPopup.editor.execCommand("mceInsertContent", false, inserte
```

```
    tinyMCEPopup.close();
},
```

insert - функция, которая сначала очищает текстовую область, затем записывает все изменения из приложения GeoGebra в отдельную переменную code. Объявляется переменная inserted - это строка, содержащая параметры GeoGebra и div контейнер, куда оно вставится. Выполняется команда tinyMCEPopup.editor.execCommand(). Имеет 3 параметра: название команды, True/false(должен ли быть представлен пользовательский интерфейс), то что вставляется. После этого диалоговое окно закрывается методом close().

reloadGraphig, reloadGeometry, reload3d - похожие функции, которые вызывают прогрузку приложения GeoGebra в id "ggb-element". Вся их разница в параметре appName. То есть, загрузка разных режимов GeoGebra.

```
reloadGraphig: function () {
  var params = {
    appName: "graphig",
    width: window.screen.width - 100,
    height: window.screen.height - 100,
    showToolBar: true,
    showAlgebraInput: true,
    showMenuBar: true,
  };
  var applet = new GGBApplet(params, true);
  applet.inject("ggb-element");
```

В конце файла вызывается функция tinyMCEPopup.onInit.add(GeoDialog.init, GeoDialog). Срабатывает при открытии диалогового окна и инициализирует GeoDialog.

Заключение

Библиографический список

1. Библиография оформляется по ГОСТ 7.0.5-2008
2. Библиография оформляется по ГОСТ 7.0.5-2008

ПРИЛОЖЕНИЕ А

Листинг программы

```
1 f(x, F) := block([i, S],
2     S: zeromatrix(dim, dim),
3     for i: 1 thru length(F) do
4         S: S+mod(F[i]*(x^^(i-1)), P),
5     return(mod(S, P))
6 );
```