

Как создать свой набор задач для Чертёжника-Blockly?

Что это такое?

Чертёжник-Blockly [1] – это приложение для обучения программированию, в котором исполнитель Чертёжник из системы «Исполнители» [2] управляется с помощью программы, собранной из блоков (как в среде Scratch).

На странице [1] (см. список ссылок ниже) размещены несколько готовых наборов задач, которые можно использовать как в режиме онлайн, так и локально (на компьютере, не имеющем доступа к Интернету).

Оффлайн-версию можно скачать со страницы [1]. Эта версия позволит вам создать собственные наборы задач для Чертёжника, собрать собственную палитру блоков и установить собственные ограничения на количество используемых блоков. Про то, как это сделать, рассказывает этот документ.

Если у вас получился удачный набор задач и вы готовы поделиться им с коллегами, присылайте архив с файлами на почту kpolyakov@mail.ru.

Ссылки:

[1] Робот-Blockly: <http://kpolyakov.spb.ru/school/robots/blockly.htm>

[2] Система «Исполнители»: <http://kpolyakov.spb.ru/school/robots/robots.htm>

1. Скачивание архива

Сначала нужно скачать архив `rblockly.zip` со страницы [1] или по прямой ссылке:

<http://kpolyakov.spb.ru/loadstat.php?f=/download/rblockly.zip>

Распакуйте архив в отдельный каталог. Вы должны увидеть каталоги

`js` – скрипты, которые используются при работе приложения;

`media` – рисунки и звуковые файлы;

`solutions` – решения всех задач, опубликованных на странице [1].

2. Создание файлов вашего приложения

Для создания своего приложения необходимо отредактировать два файла – веб-страницу на языке HTML и файл на языке JavaScript, в котором описываются все уровни игры. Рекомендуется давать этим файлам имена с одинаковым окончанием. Например, пусть мы хотим создать блок задач по искусственному интеллекту (AI – artificial intelligence). Тогда можно выбрать окончание `ai`, так что наши файлы будут называться

`drawer_ai.html`

`drawer_ai.js`

Файл `drawer_ai.html` должен быть в корне, то есть в том же каталоге, где находятся все HTML-файлы. А файл `drawer_ai.js` нужно записать в подкаталог `js`.

Для упрощения жизни в архиве есть файлы-заготовки, с которых можно начать работу:

```
drawer_.html
js/drawer_.js
```

Создайте их копии с именами `drawer_ai.html` и `drawer_ai.js` (важно, чтобы второй файл был создан в подкаталоге `js`).

Редактировать эти файлы можно в любом текстовом редакторе, например, в Блокноте, NotePad++, Sublime Text и др.

3. Редактирование HTML-страницы

- 1) Откройте файл `drawer_ai.html`. Найдите в начале файла строки

```
<!-- Измените название вашего приложения -->
<title>Чертёжник: введите здесь название набора задач</title>
<!-- -->
```

Вместо текста, выделенного маркером, введите название своего приложения. Эта строка появится в заголовке страницы и в самом верху тела страницы.

- 2) Затем найдите в том же файле строчки

```
<!-- Добавьте ссылку на файл, который содержит набор задач -->
<script src="js/drawer_.js"></script>
<!-- -->
```

и исправьте название JavaScript-файла на `drawer_ai.js`:

```
<!-- Добавьте ссылку на файл, который содержит набор задач -->
<script src="js/drawer_ai.js"></script>
<!-- -->
```

4. Редактирование набора задач для Черепашки

Все остальные данные хранятся в файле `drawer_ai.js`.

Найдите в начале файла `drawer_ai.js` массив `Maps`, начало которого выглядит примерно так:

```
var Maps =
[ {}, // Level 0
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 1
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 2
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 3
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 4
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 5
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 6
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 7
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 8
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 9
  { 'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2, 'unit': 30 }, // Level 10
];
```

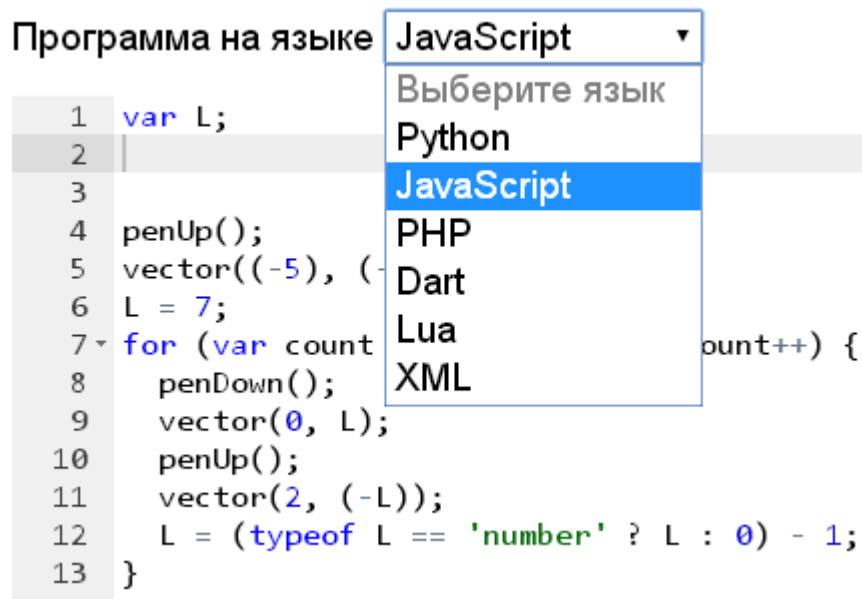
Это массив, в котором каждый уровень описывается как объект. Уровень 0 не используется. Для каждого уровня 1-10 можно определить следующие поля JavaScript-объекта, которые определяют начальное состояние Чертежника:

x, y: начальные координаты (точка (0,0) – это центр поля Чертёжника);
show: логическое значение, если оно равно **true**, Чертёжник видна на экране;
pen: логическое значение, если оно равно **true**, перо Чертёжника опущено;
width: толщина линии.

Задачи для Чертёжника хранятся в виде программ, после выполнения которых на экране появляется нужный рисунок. Эти программы хранятся в массиве **Answer**:

```
var Answers = [ null, // Level 0 not used
// Level 1
function() {
},
// Level 2
function() {
},
...
];
```

Самый простой способ построения такой функции-ответа состоит в том, чтобы построить из блоков и отладить нужную программу, затем, используя выпадающий список в нижней части окна, выбрать язык JavaScript, и скопировать всю полученную программу в тело функции-ответа для нужного уровня:



5. Редактирование ограничений

Для каждого уровня вы можете задать максимальное количество блоков, которые можно использовать в решении. Найдите в файле **drawer_ai.js** массив **BlockLimit**:

```
var BlockLimit = [0, // Level 0 unused
50, // Level 1
50, // Level 2
50, // Level 3
```

```

50, // Level 4
50, // Level 5
[30, 40, 50], // Level 6
[30, 40, 50], // Level 7
[30, 40, 50], // Level 8
[30, 40, 50], // Level 9
[30, 40, 50], // Level 10
];

```

Предельное количество блоков задаётся для каждого уровня отдельно. Сейчас на первых пяти уровнях установлено ограничение в 50 блоков. Для следующих уровней установлено три ступени ограничений (массив). Наибольшее число в массиве (50) – это максимальное число блоков, которые ученик может использовать в программе. При этом решение засчитывается и решение получает рейтинг «три звезды». Следующее число (40) – это максимальное число блоков, за которое можно получить «серебряный кубок» и рейтинг «четыре звезды». Первое число означает количество блоков, за которое программа получает рейтинг «пять звёзд» и «золотой кубок».

Если вы изменяете количество уровней, проверьте, что количество значений в массиве `BlockLimit` вы тоже изменили, иначе будет ошибка.

Вы можете задать предельное количество блоков какого-то типа. Например, вы хотите, чтобы ученик на уровне 5 использовать блок **вперёд** только два раза. Найдите в файле `drawer_ai.js` массив `someBlockLimit` и добавьте ограничения к уровню Level 5:

```

var someBlocksLimit = [ {}, // Level 0 unused
{ }, // Level 1
{ }, // Level 2
{ }, // Level 3
{ }, // Level 4
{ 'drawer_vector': 2 }, // Level 5
{ }, // Level 6
{ }, // Level 7
{ }, // Level 8
{ }, // Level 9
{ }, // Level 10
];

```

В этом массиве ограничения каждого уровня задаются в виде объекта JavaScript (словаря). Ключи этого словаря – это названия блоков, а значения – наибольшее разрешенное количество блоков этого типа. Как только это количество блоков израсходовано, блок в палитре становится неактивным.

Приведём кодовые названия некоторых блоков:

подними перо	<code>drawer_pen_up</code>	
опусти перо	<code>drawer_pen_down</code>	
в точку	<code>drawer_point_simple</code>	только число
	<code>drawer_point</code>	можно с переменной
вектор	<code>drawer_vector_simple</code>	только число
	<code>drawer_vector</code>	можно с переменной

повторить N раз	controls_repeat_list
процедура	procedures_defnoreturn
вызов процедуры	procedures_callnoreturn

6. Редактирование справки

Вы можете для каждого уровня определить сообщение пользователю (инструкцию), которое появляется при загрузке уровня. Найдите в файле `drawer_ai.js` массив `HelpContent`:

```
var HelpContent = [ '', // Level 0 not used
// Level 1
'',
// Level 2
'',
...
];
```

Если сообщение для какого-то уровня пустое, оно не выводится. Сообщение представляет собой символьную строку, в которой можно использовать HTML-тэги (например, выделять слова жирным и курсивом, вставлять рисунки). Вот пример, в котором на уровне 1 справка содержит таблицу и рисунок:

```
var HelpContent = [ '', // Level 0 not used
// Level 1
'<table><tr><td></td><td>' +
'Новый блок &laquo;если&raquo; позволяет выполнить группу ' +
'команд только в том случае, когда верно (истинно) ' +
'условие после слова &laquo;если&raquo;. ' +
'Эти команды нужно поставить внутри блока ' +
' в правильном порядке.<br> </td></tr></table>',
// Level 2
'',
...
];
```

7. Редактирование палитры блоков

Вы можете собрать свою собственную палитру блоков, удалив ненужные блоки. Найдите в конце файла `drawer_ai.js` функцию `BlocklyBlocks`:

```
function BlocklyBlocks( Level ) {
  if( [1,2,3,4,5,6,7,8,9,10].includes(Level) ) return '' +
  '<xml xmlns="https://developers.google.com/blockly/xml" ' +
  'id="toolbox" style="display: none">' +
  //===== Команды Чертёжника =====
  '<category name="Чертёжник" colour="%{BKY_ROBOT_HUE}">' +
```

```

    // вектор(число1, число2)
    ' <block type="drawer_vector_simple"></block>' +
    // вектор(a, b)
    ' <block type="drawer_vector">' +
    '   <value name="DELTAX">' +
    '     <shadow type="math_number">' +
    '       <field name="NUM">10</field>' +
    '     </shadow>' +
    '   </value>' +
    '   <value name="DELTAY">' +
    '     <shadow type="math_number">' +
    '       <field name="NUM">10</field>' +
    '     </shadow>' +
    '   </value>' +
    ' </block>' +
    ...

```

Комментарии, которые начинаются с символов //, показывают, какой блок далее добавляется. Строчки, где добавляются ненужные вам блоки, можно просто удалить.

В примере, приведённом выше, на всех уровнях используется один и тот же набор блоков. Это не обязательно так. Можно для каждого уровня определить свою палитру блоков:

```

if( [1,2,3,4,5].includes(Level) )
  return '' + ...;
if( [6,7,8,9,10].includes(Level) )
  return '' + ...;

```

В этом примере для уровней 1-5 задаётся один набор блоков, а для уровней 6-10 – другой.

8. Отладка набора задач

Рекомендуется следующая последовательность отладки нового набора задач:

- 1) подготовить файл `drawer_ai.html` и открыть его в браузере Chrome или Firefox, где есть инструменты разработчика;
- 2) определить нужный набор блоков;
- 3) используя блоки, собрать программу для построения желаемого рисунка для уровня 1;
- 4) скорректировать начальное положение Черепахи в массиве `Maps` так, чтобы рисунок располагался примерно в центре поля Черепахи;
- 5) перевести программу на язык JavaScript и добавить решение тело функции нужного уровня в массиве `Answers`;
- 6) обновить страницу; проверить, чтобы желаемый рисунок был показан на заднем плане;

7) после успешной отладки первого уровня перейти ко второму и т.д.