Setting up Blockly and creating your custom block

Contents

[1 Download and Setup 2](#_Toc97311467)

[1.1 Requirements 2](#_Toc97311468)

[1.2 Official guide 2](#_Toc97311469)

[1.3 Important Steps 2](#_Toc97311470)

[2 My template code 3](#_Toc97311471)

[2.1 Introduction 3](#_Toc97311472)

[2.1 Creating a sample XjObject block 4](#_Toc97311473)

[2.2 Linking your new XjObject Operator 5](#_Toc97311474)

# 1 Download and Setup

## 1.1 Requirements

|  |  |
| --- | --- |
| 1 | Install on npm `npm install blockly` |
| 2 | Clone blockly official repository `git clone https://github.com/google/blockly` |
| 3 | Clone my template sample code to make adjustments to `git clone https://github.com/Berttwm/Blockly-XjObject` |

## 1.2 Official guide

Please refer to official guides below for a better understanding on blockly features.  
Guide: <https://developers.google.com/blockly/guides/overview>

## 1.3 Important Steps

1) Open blockly official block creator  
Inside blockly official folder, head to `./demos/blockfactory/ ` and open the `index.html` file. This will be where you create custom blocks to be inserted into your projects. My guide will only briefly teach how to use this. For full tutorial, please refer to the official guide on how to use this tool here: <https://developers.google.com/blockly/guides/create-custom-blocks/blockly-developer-tools>   
Graphical user interface, text

Description automatically generated

2) Open my template folder file  
Go to the folder where you downloaded from <https://github.com/Berttwm/Blockly-XjObject>, and use a text editor to open “./index.html”, “./code.js” and “./scripts/xjobject\_block.js”. 

|  |  |
| --- | --- |
| index.html | Front-end page for my project. |
| code.js | Important execution functions for this project (also where the function used to save python code is found and should be edited) |
| xjobject\_block.js | Where you copy and paste the Javascript code from the official block creator in step 1) to. |

# 2 My template code

## 2.1 Introduction

Table

Description automatically generated

Tabs

Run

XjOBject folder

Newly created blocks

In this project, I have created 3 special blocks using the block creator and added them into the XjObject folder. You can click on the Run button to execute the code (runs on your browser by evaluating JavaScript code). You can also click on the tabs to view the language-specific code generated by blockly’s code generator.   
  
**Please Note: If you click on ‘Python’ tab, my sample will automatically download a ‘python\_runnable.py’ file and save it in your downloads folder before showing you the python code. This is only a showcase feature and should be edited out in future (inside code.js line 372). Rightfully, this function should change to 1) prompt user for download path, 2) save file according to user provided name.  
Text

Description automatically generated**

The index.html portion which is holding my XjObject operators and creator is in line 427-431. All new blocks created need to be appended here as a <block> type.  
  
Text

Description automatically generated

## 2.1 Creating a sample XjObject block

1) Generating Block using official block creator  
Example of the XjObject operator sample in my template should look like this in the block creator. Text

Description automatically generated  
  
The following section will highlight some important features to set before copying the code into your xjobject\_block.js file.

Copy and paste these two sections into **xjobject\_block.js**

return type

Input type

Standardized color

block name

|  |  |
| --- | --- |
| Block name | This block name will be also used in the index.html file to be appended as a <block type=**$BLOCK\_NAME**></block> (refer to previous section) |
| Input type | In my examples, the XjObject is an array of various types (string, number, list). I have to specify this so that the user will not place the wrong input type into the input slot |
| Return type | Similar to input type, the operator you create should return an XjObject to be chained with other Xj Operators. This will ensure it can also fit into and be chained with other Xj Operators |
| Color | For consistency, you should label all your Xj blocks with the same colour. For all my examples, I use ‘0’ as my colour. |
| Code block | Both Code blocks MUST be copied into the xjobject\_block.js before linking with your index.html file. Refer to my current file to see how I have previously done it. \*Please note: For “Block Definition” and “Generator Stub”, I generated **JavaScript** format. |

## 2.2 Linking your new XjObject Operator

Sanity Check: By now you should have 1) Created your new block, 2) copied and pasted your Block definition and Generator Stub into your xjobject\_block.js file, and 3) appended a block inside your XjObject category inside your index.html file.

Please note that the code generator for blockly works through string generation. Meaning you have to generate the entire code in string format for it to appear inside “JavaScript” or “Python” tab.   
Text

Description automatically generated  
> Example of python code generated.  
  
  
  
  
  
Refer to my completed examples to understand better how this works.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
1) Now copy/paste a second generator stub for the python code inside xjobject\_block.js.   
  
Text

Description automatically generated  
  
To create the Python stub, simply change all occurances of “**JavaScript**” in your JavaScript Stub to “**Python**” (4 total in my case – more if you have more inputs). All code functionality will be inside the stubs.   
  
2) To write chained operators (where the input type and return type of each operator is an “XjObject”), please take note of the following steps.  
  
Text

Description automatically generated

Additional Generator Stub for python

3. return priority

2. var code

1.Auxiliary function

|  |  |
| --- | --- |
| 1) Auxiliary Functions | This function will appear as a newly defined function inside your script but will be invisible to the user. This allows you to method-chain your inputs while providing complicated functionality. Each comma leads to a new line. Tab and spaces is important in this step. Essentially a string hardcode. Refer to official guide under “Utility functions” ([Caching Arguments](https://developers.google.com/blockly/guides/create-custom-blocks/caching-arguments)). |
| 2) var code | Hard code a call to your auxiliary function with the block’s input parameters as the auxiliary function’s argument. |
| 3) return priority | Set return priority (argument 2 of return) to “Blockly.JavaScript.ORDER\_FUNCTION\_CALL” (for JavaScript stub) to ensure that the generator provides the right ordering for chained operators. |

Please Note: Your python stub will require even more attention to spacing and tab spaces as python language is indent sensitive.  
  
Text

Description automatically generated  
  
2.3 Generating your python file (TO UPDATE)

Simply click on the “**Python**” tab in your index.html file and a python\_runnable.py file should be automatically downloaded in to you download folder.

Text

Description automatically generated  
  
Inside code.js (line 401 – 414) is where the `saveTextToFile()` function is executed at specifically for only the python tab (line 406 - 409). The `saveTextToFile()` function should be updated in future to allow for **1) prompt user for download path, 2) save file according to user provided name**.