

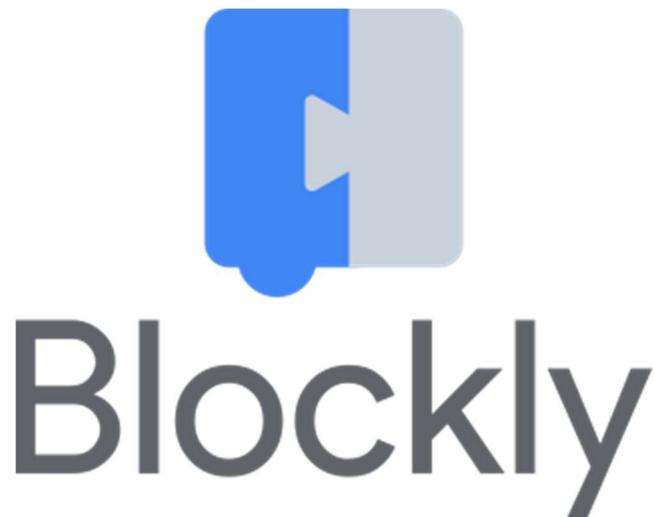


POLITÉCNICO  
DE LEIRIA

iModDom



## Tutorial - Criar Blocos no *Blockly*



**Elaborado por:**

Marco Pereira - 2190516

**Orientado por:**

Luís Bento

Carlos Neves

---

## Índice

<b>1.</b>	<b>INTRODUÇÃO .....</b>	<b>1</b>
1.1.	OBJETIVOS.....	1
<b>2.</b>	<b>CRIAR BLOCOS .....</b>	<b>2</b>
<b>3.</b>	<b>EXPORTAR BLOCOS .....</b>	<b>15</b>
<b>4.</b>	<b>CRIAR UM <i>WORKSPACE</i> .....</b>	<b>18</b>
<b>5.</b>	<b>CONSTRUÇÃO DA PÁGINA HTML .....</b>	<b>22</b>

## 1. Introdução

O *Blockly* é uma biblioteca da *Google* e adiciona um editor visual de código às páginas *web* e a aplicações de *smartphones*, é gratuito e *open source*. Permite ao utilizador gerar qualquer linguagem de código através de blocos, o que se torna bastante intuitivo para aqueles que estão a dar os primeiros passos na programação.

### 1.1. Objetivos

Com este tutorial, o objetivo é ensinar a criar blocos, um *workspace* e integrar tudo isso numa página html.

## 2. Criar Blocos

1. Aceda ao seguinte link: <https://blockly-demo.appspot.com/static/demos/blockfactory/index.html>
2. Como pode verificar, foi direcionado para uma página onde é possível criar blocos (Figura 1). Na lateral esquerda será onde se adicionará os vários tipos de componentes aos blocos, no campo “Preview”, tal como o nome indica é onde é possível pré-visualizar o aspeto do bloco, no campo “Block Definition” é onde é apresentado o código da parte visual do bloco e no último campo, “Generator stub”, surge o código que o bloco irá utilizar para gerar código.

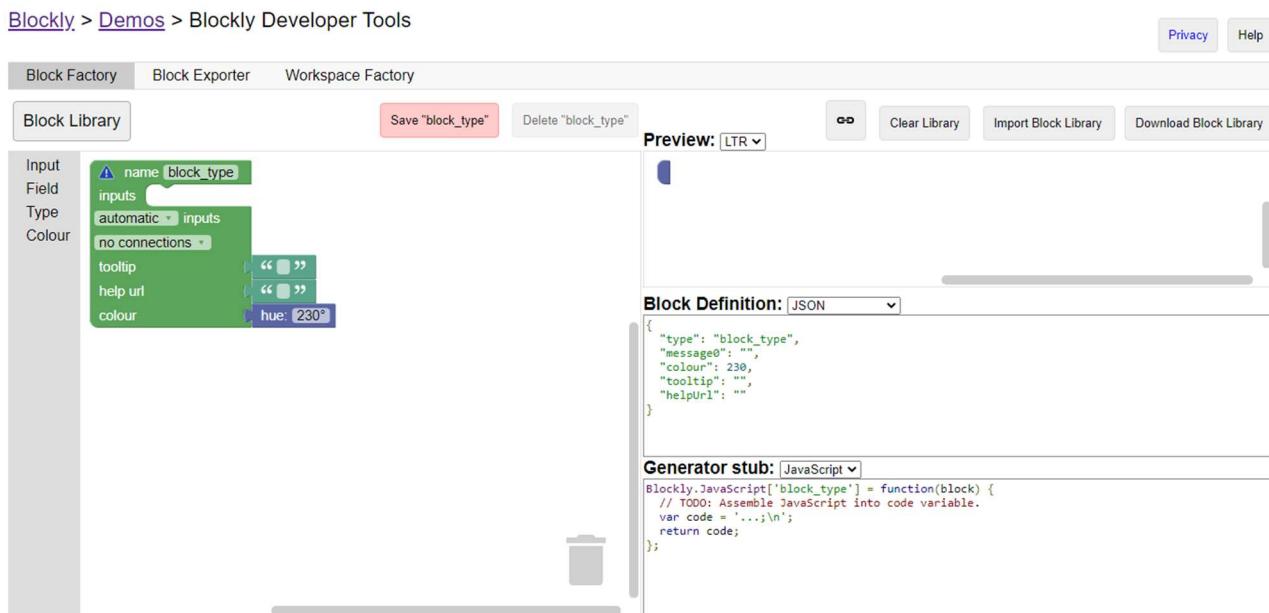


Figura 1 – Página de Criação de Blocos

3. Para começar, atribua um nome ao seu bloco no campo “name”, este nome será apenas utilizado no construtor de blocos (Figura 2).

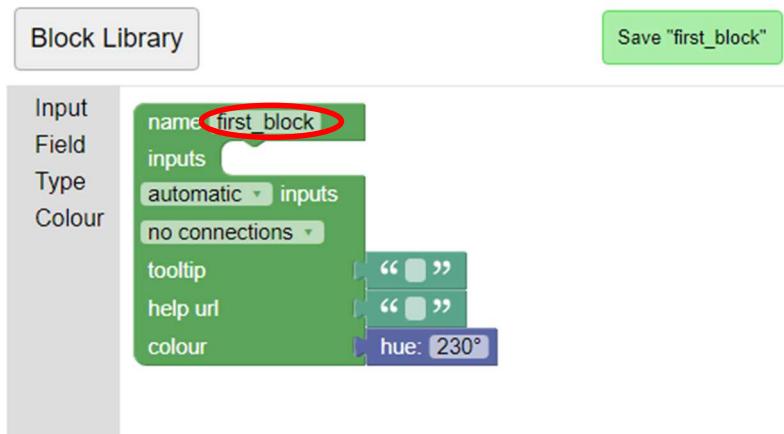
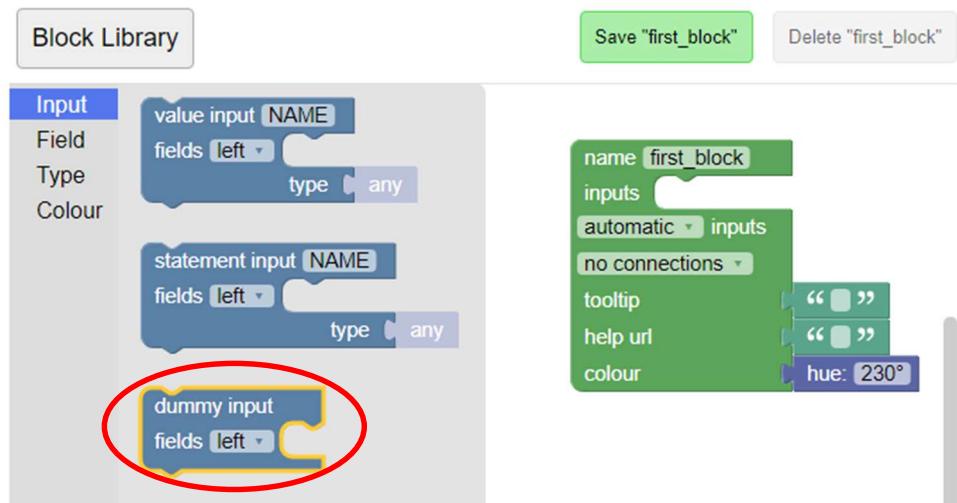
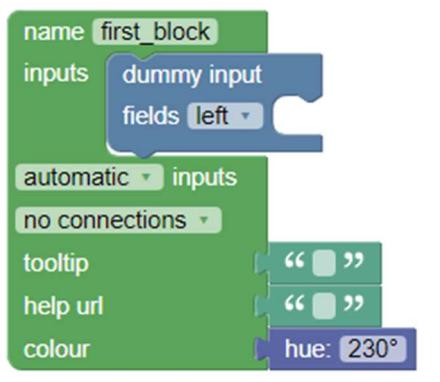


Figura 2 – Atribuir Nome

4. Na aba lateral esquerda, clique em “*Input*” (Figura 3) e de seguida arraste o bloco “*dummy input*” para o campo “*inputs*” do seu bloco, de modo a ficar com o aspeto da Figura 4. Sempre que adicionar um bloco do tipo *input*, uma nova “linha” será criada no bloco final.



**Figura 3 – Clicar em “*Input*”**



**Figura 4 - Aspetto do Bloco**

5. De seguida, na mesma aba lateral, aceda a “*Field*” e arraste o primeiro bloco “*text*” para o dentro do bloco “*dummy input*” anteriormente adicionado (Figura 5).

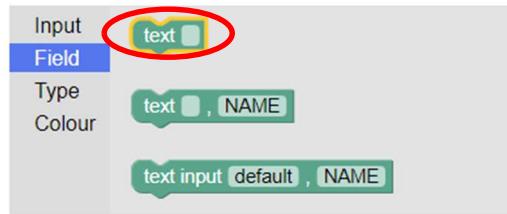


Figura 5 - Seleção do Bloco "text"

6. Escreva algo no campo do bloco “*text*” que acabou de adicionar. Pode verificar que o aspetto do bloco alterou e que agora aparece o que escreveu, note que no campo “*Block Definition*” também houve alterações (Figura 6).

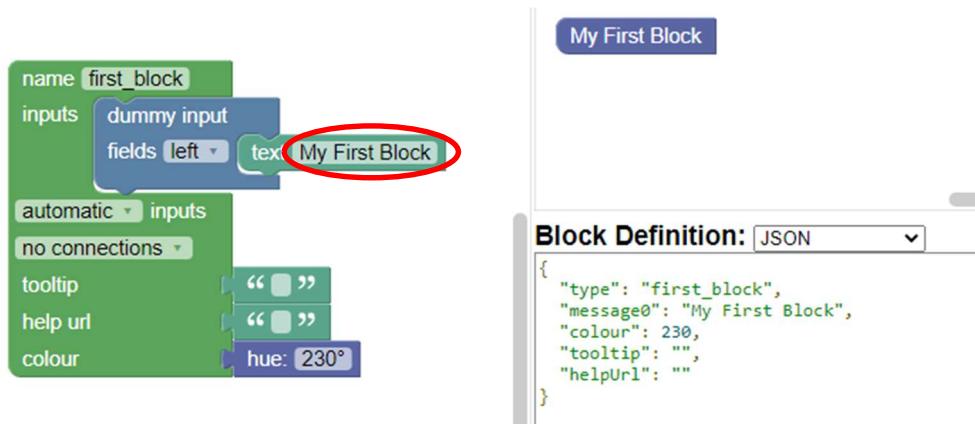


Figura 6 - Bloco de Texto

7. Adicione um outro bloco “*dummy input*” ao seu bloco e de seguida adicione um bloco “*text input*” ao seu novo bloco “*dummy input*” (Figura 7). O aspeto final deverá ser semelhante à Figura 8.

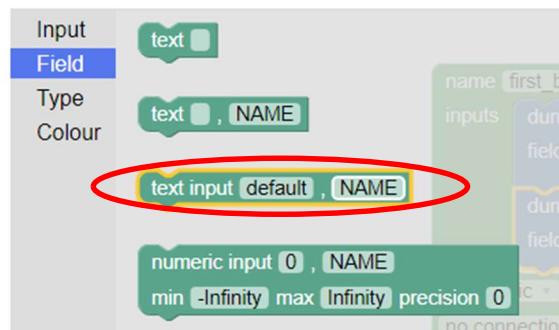
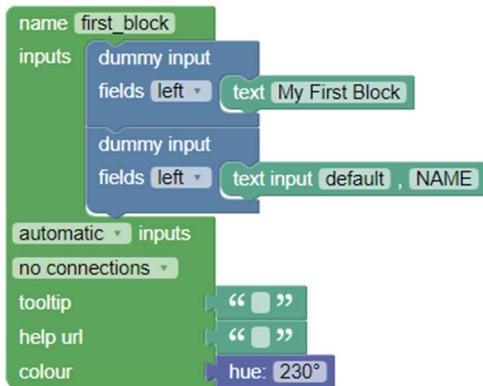
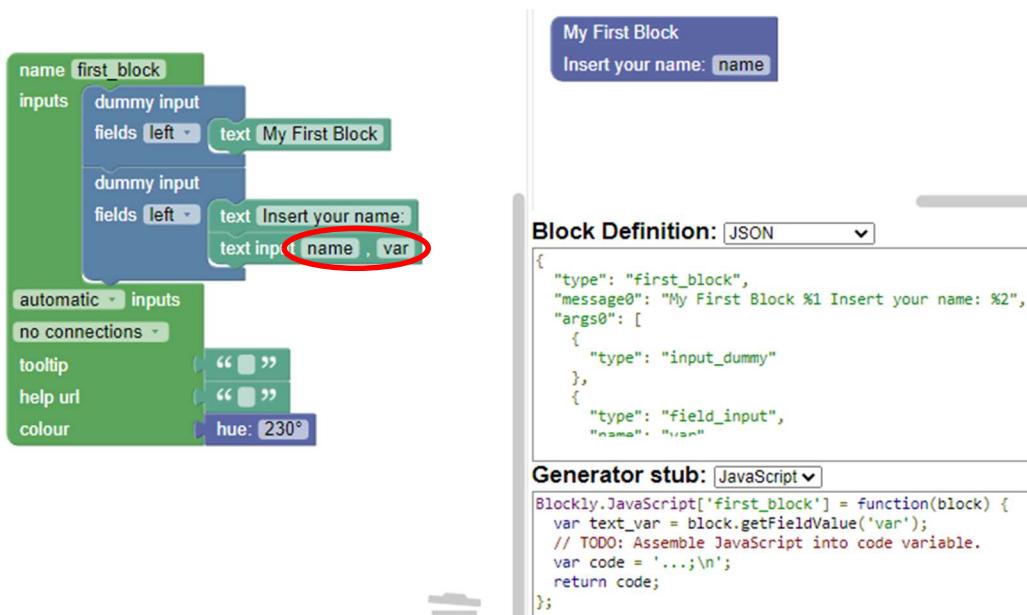


Figura 7 – Bloco “text input”



**Figura 8 - Aspetto do Bloco**

8. O bloco que acabou de adicionar, tal como o nome indica, tem a função de armazenar numa variável o conteúdo introduzido pelo utilizador. O primeiro campo desse bloco é o que aparece por predefinição no bloco final e o segundo campo será o nome da variável onde é armazenado o conteúdo do primeiro campo. Posto isto, escreva algo para aparecer por predefinição no bloco final, e atribua um nome à variável, também pode adicionar um bloco “text” para contextualizar o bloco “text input”. Como pode verificar, o aspeto do bloco final e os campos do código do bloco sofreram alterações, nomeadamente no campo “Generator stub” existe agora uma variável com o nome “text\_” seguido do nome que lhe atribuiu (Figura 9).



**Figura 9 - Aspetto do Bloco**

9. Adicione novamente um bloco “*dummy input*” e dentro desse bloco coloque um bloco “*dropdown*” acedendo à aba “*Field*”. Este bloco, permite ao utilizador selecionar uma de várias opções (Figura 10).



Figura 10 - Bloco "dropdown"

10. No primeiro campo do bloco que acabou de adicionar, atribua o nome da variável que será utilizada para guardar a opção selecionada pelo utilizador, nos outros campos, à esquerda escreva o nome da opção que deseja que apareça no bloco final e à direita o que deseja que seja guardado na variável do bloco. Note que se todas as opções tiverem o texto inicial ou final igual, no bloco final esse mesmo texto não surge nas opções (Figura 11).

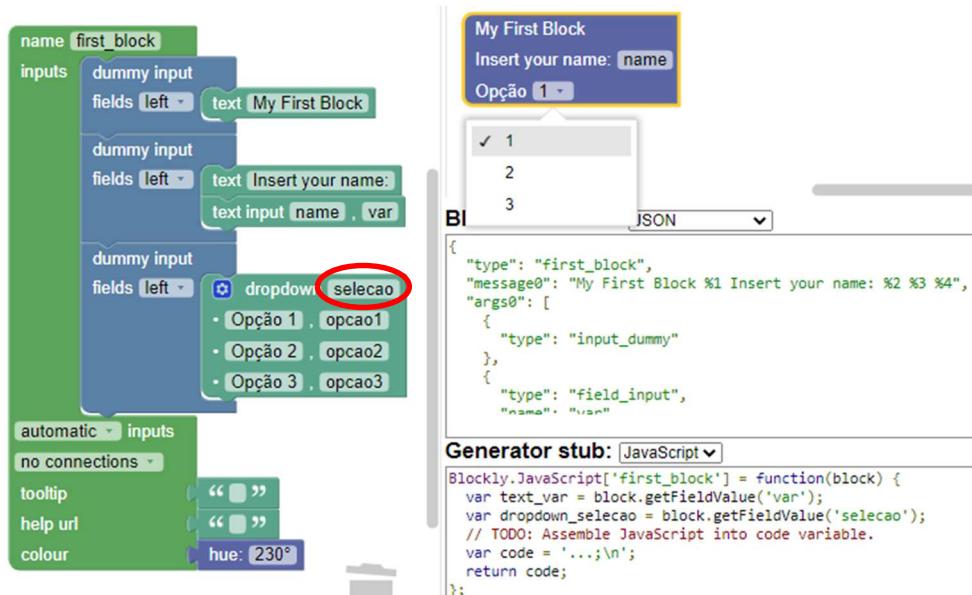


Figura 11 - Aspetto do Bloco

Pode adicionar ou remover campos de opções clicando nas definições do bloco e arrastando mais um bloco para o bloco “*add options*” ou removendo blocos deste último (Figura 12).

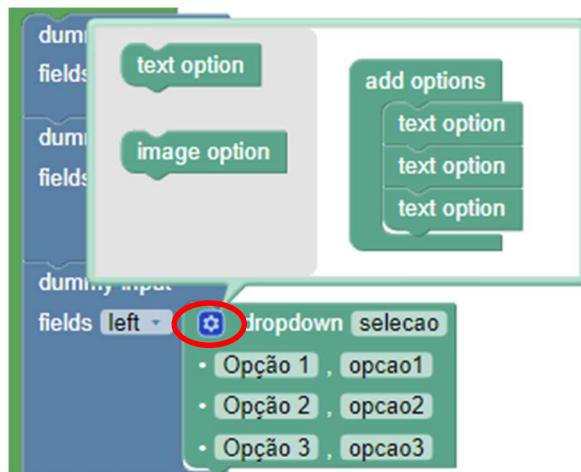


Figura 12 - Adicionar ou Remover Campos de Opção

11. Para adicionar uma caixa de seleção, aceda a “*Field*” e selecione o bloco “*checkbox*”, colocando-o dentro de um bloco do tipo *input*, como é o caso do bloco “*dummy input*” (Figura 13).

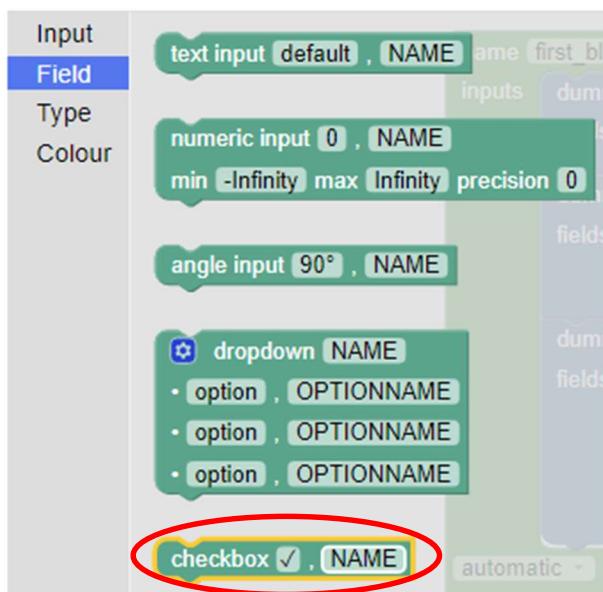


Figura 13 - Bloco "checkbox"

12. No campo do bloco acabado de adicionar, atribua um nome à variável que guardará o estado da caixa de seleção, pode ainda alterar o estado de predefinição da mesma clicando na caixa de seleção localizada no bloco “checkbox” (Figura 14).

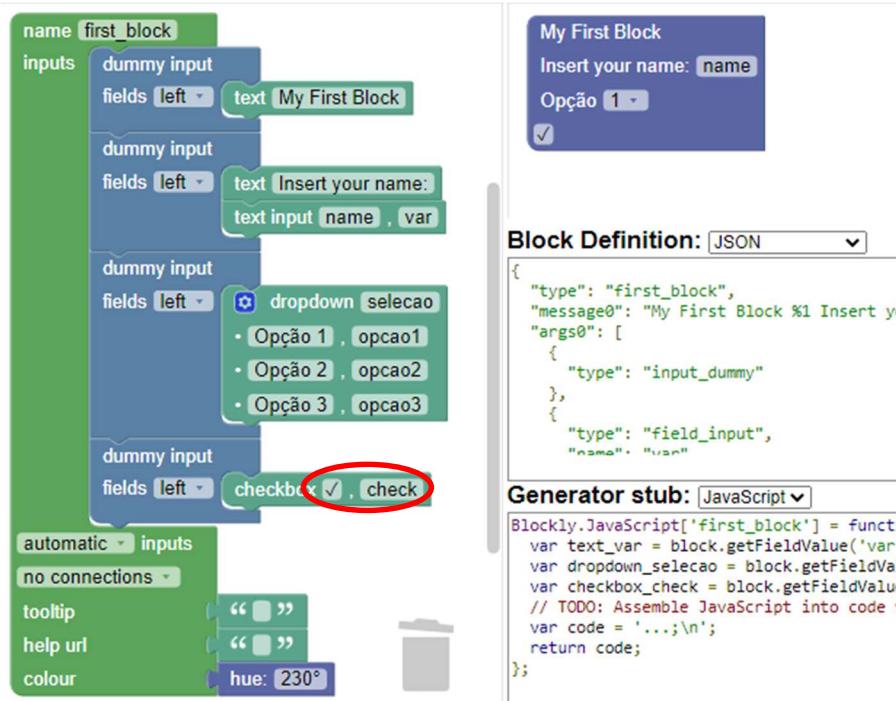


Figura 14 - Aspetto do Bloco

13. Agora que já sabe trabalhar com alguns dos blocos do tipo “Field”, acceda aos blocos do tipo “Input” e adicione um bloco “value input” ao bloco principal, este bloco permite que sejam encaixados outros blocos (Figura 15).



Figura 15 - Bloco "value input"

14. Atribua um nome à variável do bloco que acabou de adicionar e repare que, agora, existe um encaixe na lateral do bloco final (Figura 16).

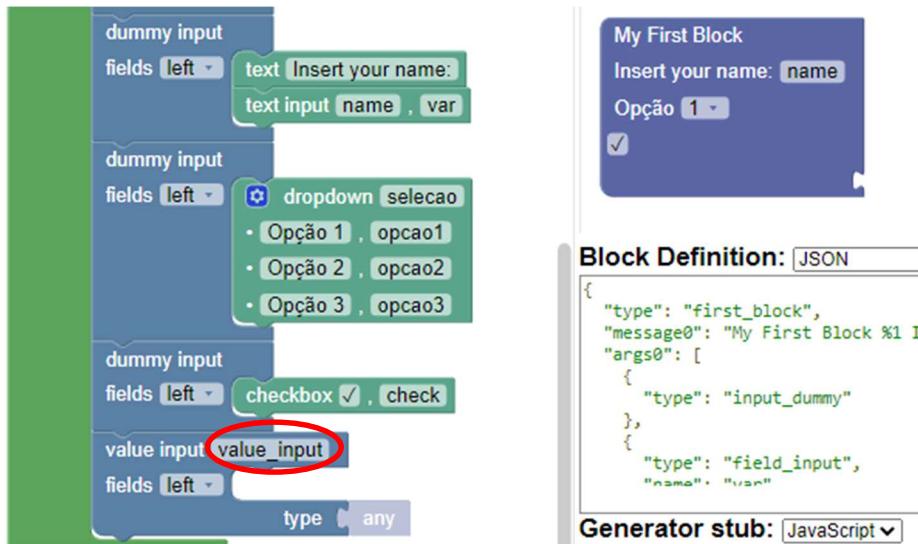


Figura 16 - Aspetto do Bloco

Pode restringir os blocos que são permitidos encaixarem nesse mesmo encaixe colocando um bloco do tipo “*type*” no campo “*type*”, e assim apenas os blocos que tenham encaixes do mesmo tipo poderão ser encaixados. O bloco “*any of*” permite que sejam encaixados qualquer tipo de blocos que sejam definidos neste bloco, já o bloco “*any*” permite que qualquer bloco seja encaixado, o tipo “*other*” apenas permite que sejam encaixados blocos do mesmo tipo que é definido neste bloco (Figura 17).



Figura 17 - Tipos de Atributos dos Blocos

15. Coloque agora um bloco “*statement input*” no seu bloco principal, este bloco, à semelhança do bloco “*value input*” permite que sejam encaixados outros blocos (Figura 18).



Figura 18 - Bloco "statement input"

16. Atribua um nome à variável do bloco que acabou de adicionar e repare que, agora, existe um outro encaixe no bloco final (Figura 19).

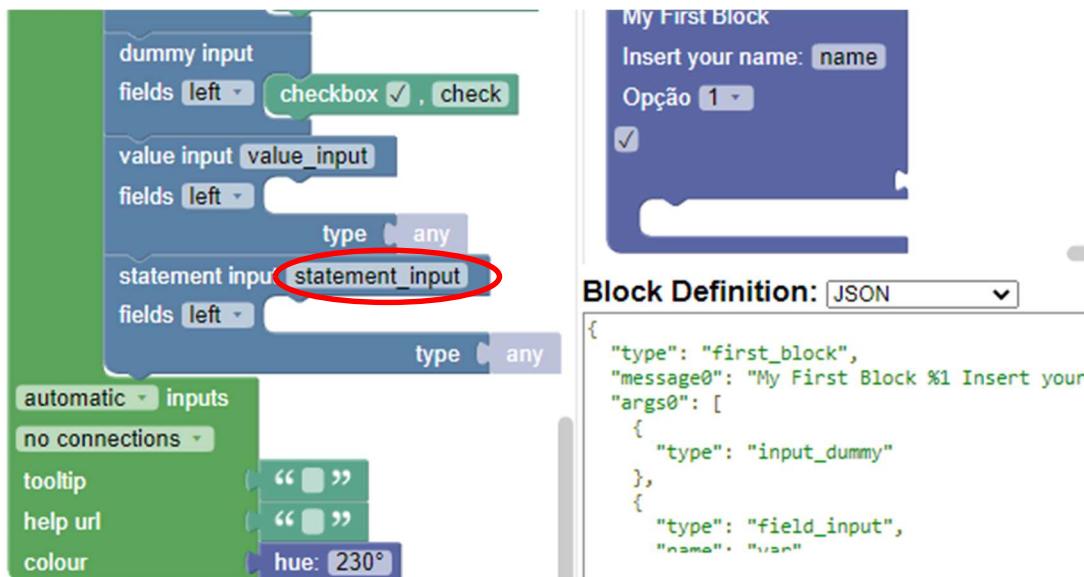


Figura 19 - Aspetto do Bloco

17. Um dos últimos campos do bloco principal é relativo às conexões, sendo que é possível selecionar “*no connections*” e o bloco final não irá ter nenhuma conexão, “*left output*” e neste caso o bloco irá ter uma conexão no lado esquerdo, “*top+bottom connections*” onde o bloco terá conexões na parte de cima e na parte de baixo, “*top connection*” e o bloco só apresentará conexão na parte de cima e ainda “*bottom connection*” onde o bloco apenas terá conexão na parte de baixo. Selecione a opção “*top+bottom connections*” (Figura 20).

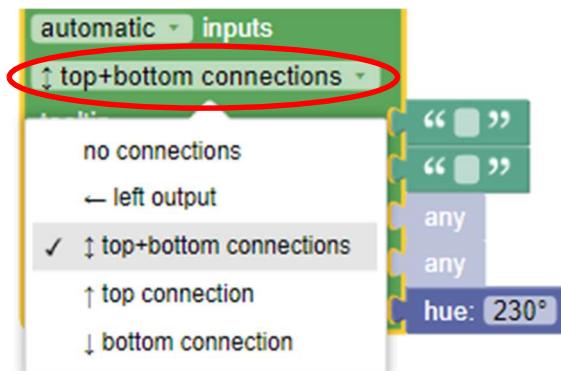


Figura 20 - Adicionar Conexões ao Bloco

18. Verifique que o seu bloco final agora apresenta uma conexão na parte de cima e também na parte de baixo, e também surgiram mais dois campos no seu bloco principal, estes têm a função de restringir os blocos que poderão ser encaixados tanto no encaixe de cima como no de baixo (Figura 21).

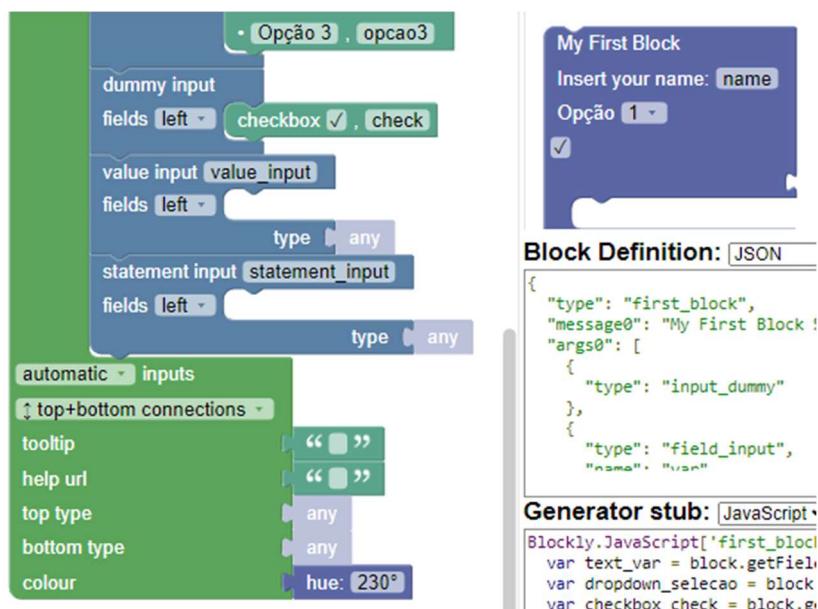


Figura 21 - Aspetto do Bloco

19. Pode ainda adicionar uma dica de utilização do bloco no campo “*tooltip*” que aparecerá quando passar o rato sobre o bloco final (Figura 22 e Figura 23).

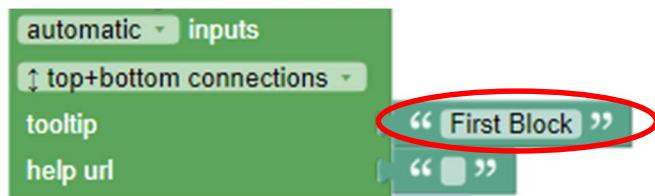


Figura 22 – *Tooltip*



Figura 23 - Demonstração da *Tooltip*

20. Na mesma zona onde adicionou uma dica de utilização, pode também adicionar um *link* de ajuda que poderá ser acedido clicando sobre o bloco final com o botão direito do rato e de seguida clicando com o botão esquerdo do rato em “*Help*” (Figura 24 e Figura 25).



Figura 24 - *Help url*



Figura 25 - Demonstração do *Help url*

21. Na aba lateral, pode ainda selecionar a cor que pretende do seu bloco final (Figura 26). Esta seleção pode também ser feita no último campo do bloco principal, clicando no número do último campo e selecionar a cor pretendida (Figura 27).

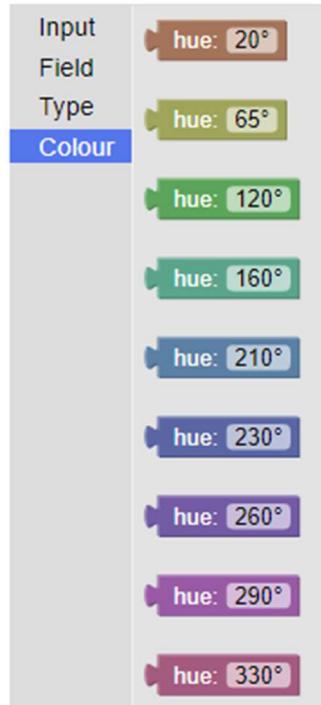


Figura 26 - Seleção da Cor do Bloco

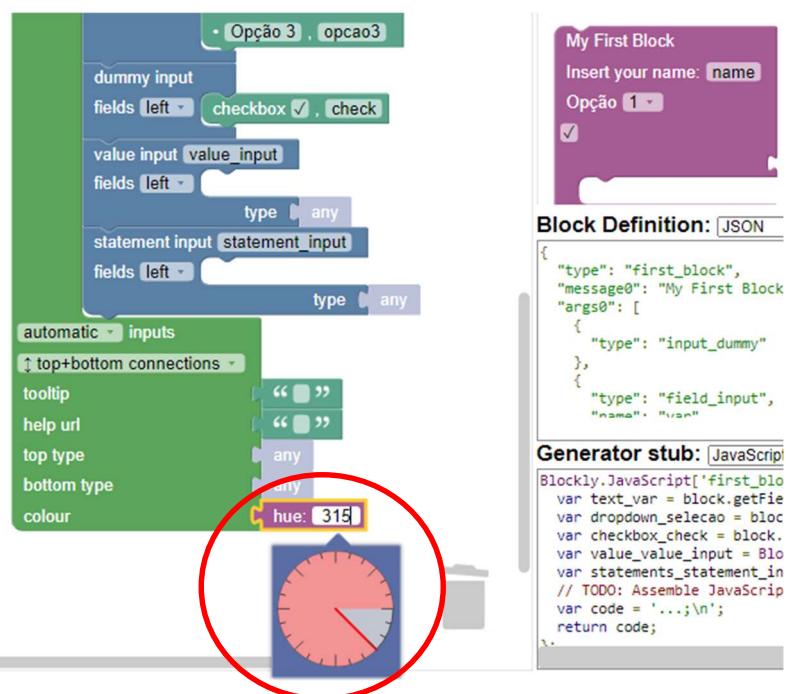
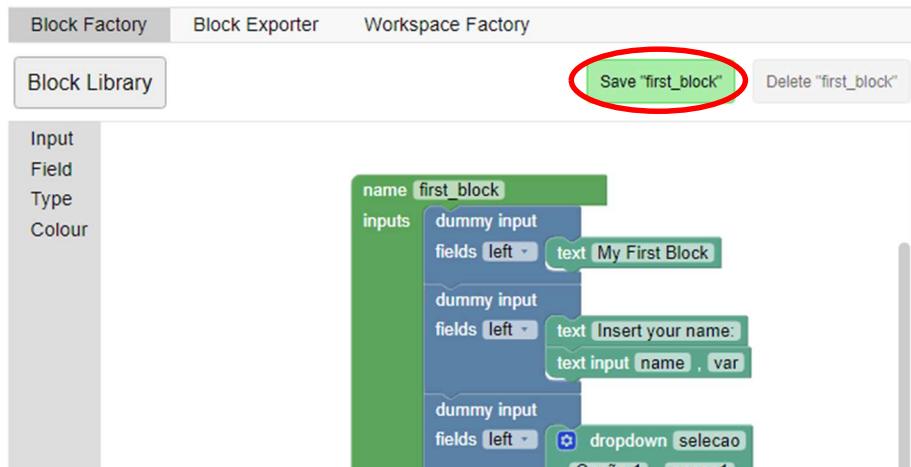


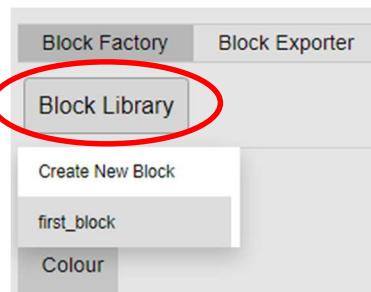
Figura 27 - Seleção da Cor do Bloco

22. Agora que já sabe como criar um bloco, pode e deve guardá-lo, para isso basta clicar no botão “Save” seguido do nome que atribuiu ao seu bloco (Figura 28).



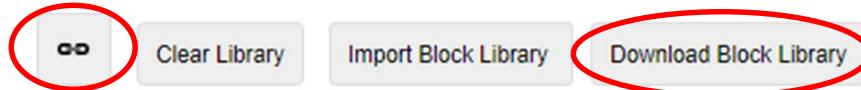
**Figura 28 - Guardar o Bloco**

23. Para criar um novo bloco ou visualizar os blocos existentes pode aceder a “Block Library” (Figura 29).



**Figura 29 - Visualizar Blocos Existentes**

24. Para guardar a sua biblioteca (conjunto de blocos criados) para poder editar mais tarde, pode criar um *link* de acesso à mesma clicando no ícone apresentado na Figura 30. Pode também clicar em “Download Block Library” para fazer o *download* da biblioteca e posteriormente fazer o seu *upload*. Se desejar editar os blocos de programação do projeto, pode fazê-lo importando o ficheiro *library.xml* que se encontra no *GitHub* do projeto em [1\\_Documentacao/3\\_Software/library.xml](#).



**Figura 30 - Guardar a Biblioteca**

### 3. Exportar Blocos

Agora que já sabe criar blocos, é necessário exportá-los. Para isso siga os próximos passos.

1. Na mesma página onde estava, clique em “*Block Exporter*” (Figura 31).

[Blockly > Demos > Blockly Developer Tools](#)



**Figura 31 - Block Exporter**

2. Será direcionado para uma página onde pode exportar os seus blocos (Figura 32).

[Blockly > Demos > Blockly Developer Tools](#)

The screenshot shows the 'Block Exporter' page with three main sections: 'Block Selector', 'Export Settings', and 'Export Preview'.

- Block Selector:** Shows a list of selected blocks. One block, 'first\_block', is highlighted with a purple background. It has a dropdown menu with options: 'Insert your name: name', 'Opção 1', and a checked checkbox.
- Export Settings:** Contains settings for exporting selected blocks:
  - Block Definition(s):** Checked, Format: JSON, File Name:
  - Generator Stub(s):** Checked, Language: JavaScript, File Name:
- Export Preview:** Displays the generated code in two sections:
  - Block Definitions:** Shows the JSON definition for the selected block.
  - Generator Stubs:** Shows the generated JavaScript stub code.

**Figura 32 - Exportar Blocos**

3. Para poder exportar os seus blocos, necessita de selecionar aqueles que deseja exportar, para isso, clique em “Select” e de seguida em “All Stored in Block Library” para selecionar todos os blocos da biblioteca (Figura 33). Se não pretender selecionar todos os blocos, pode apenas selecionar um por um clicando sobre os mesmos.

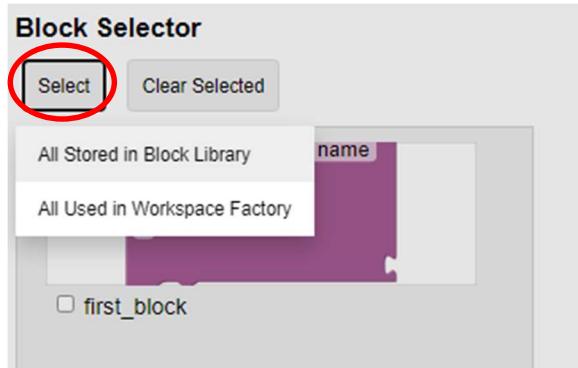


Figura 33 - Selecionar os Blocos

4. Agora que selecionou os blocos que pretende exportar, pode verificar que é apresentada uma pré-visualização do código relativo a todos os blocos selecionados, tanto do código do aspeto do bloco como do código que o bloco irá utilizar para gerar código. Atribua um nome aos ficheiros que irá exportar e de seguida clique em “Export” (Figura 34).

**Block Selector**

Select Clear Selected

Insert your name: name  
Opção 1  
✓

first\_block

**Export Settings**

Currently Selected:  
first\_block

Block Definition(s)  
Format: JSON  
File Name: Block\_definition

Generator Stub(s)  
Language: JavaScript  
File Name: Generator

Export

**Export Preview**

Block Definitions:

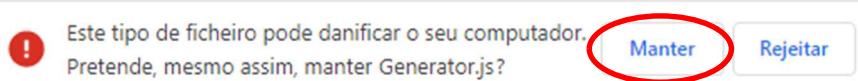
```
{
  "type": "first_block",
  "message0": "My First Block %1 Insert your name: %2 %3 %4 %5 %6 %7 %8 %9",
  "args0": [
    {
      "type": "input_dummy"
    },
    {
      "type": "field_input",
      "name": "var",
      "text": "name"
    }
  ]
}
```

Generator Stubs:

```
Blockly.JavaScript['first_block'] = function(block) {
  var text_var = block.getFieldValue('var');
  var dropdown_selecao = block.getFieldValue('selecao');
  var checkbox_check = block.getFieldValue('check') == 'TRUE';
  var value_value_input = Blockly.JavaScript.valueToCode(block, 'value_input', Blockly.JavaScript.statementToCode(block, 'statement_in'));
  // TODO: Assemble JavaScript into code variable.
  var code = '...';
  return code;
};
```

Figura 34 – Download dos Ficheiros

5. Poderá receber um alerta de segurança semelhante ao da Figura 35, caso isso aconteça basta clicar em “Manter” e o *download* será feito.



**Figura 35 - Alerta de Segurança**

## 4. Criar um *Workspace*

Neste momento já tem os blocos criados e os ficheiros dos mesmos, agora é necessário agrupar os blocos num *workspace* para que possam ser utilizados pelo utilizador.

1. Clique em “*Workspace Factory*” (Figura 36).

[Blockly](#) > [Demos](#) > Blockly Developer Tools



Figura 36 - *Workspace Factory*

2. Na página para a qual foi direcionado, clique no ícone com o símbolo “+” e de seguida em “*New Category*” para adicionar uma nova categoria (Figura 37).

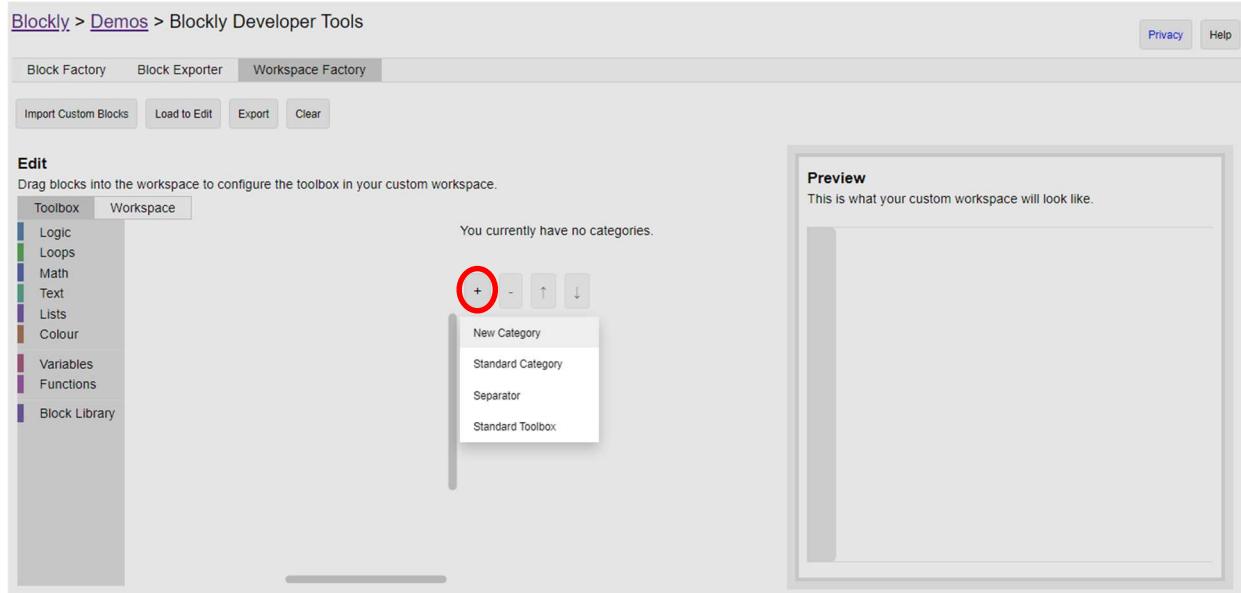
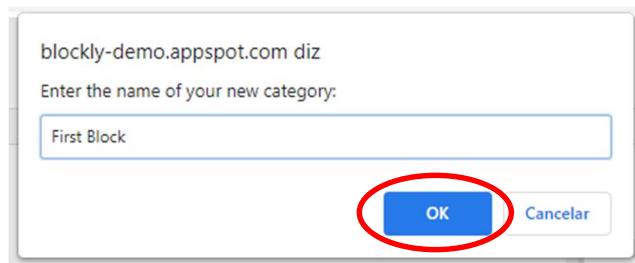


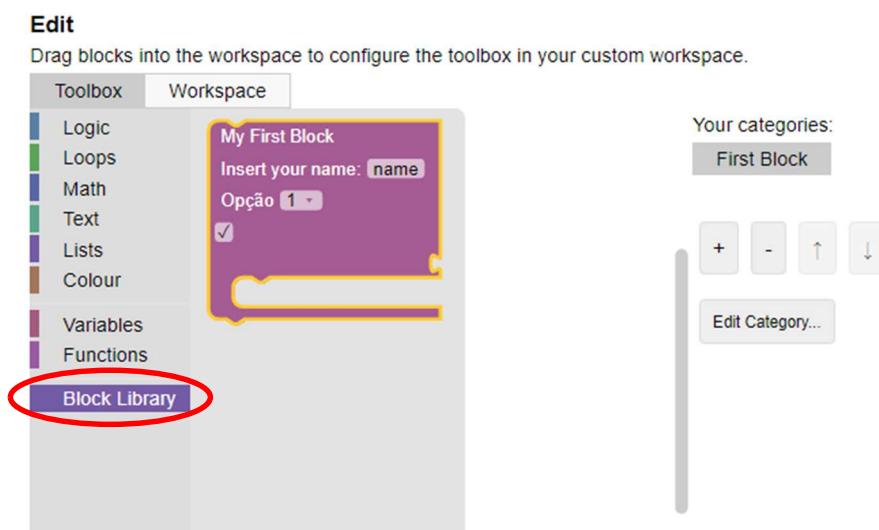
Figura 37 - *New Category*

3. Atribua um nome à categoria e de seguida clique em “OK” (Figura 38).

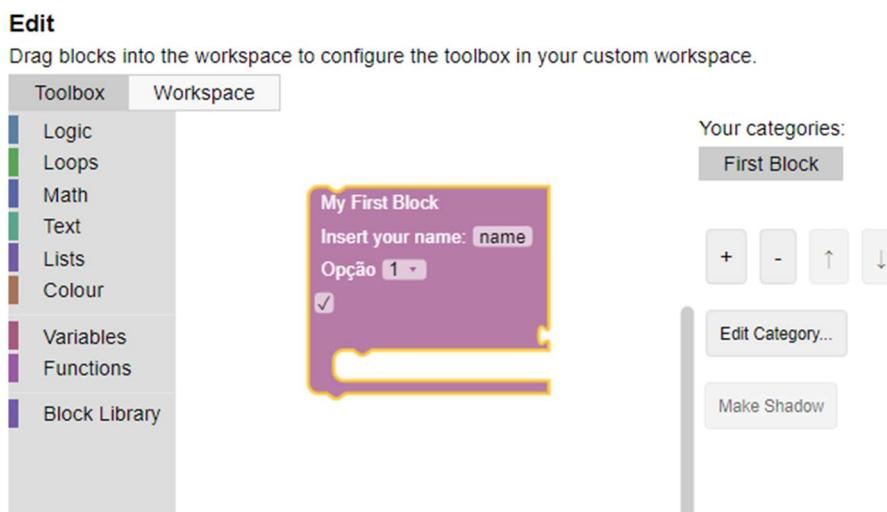


**Figura 38 - Atribuir Nome à Categoria**

4. Com a categoria criada, clique em “*Block Library*” para aceder a todos os blocos que estão na sua biblioteca, e arraste para o campo em branco os que deseja adicionar à categoria criada (Figura 39 e Figura 40).



**Figura 39 - Block Library**



**Figura 40 - Arrastar o Bloco para o Campo em Branco**

5. No lado direito da janela é apresentada uma pré-visualização do seu *workspace*, se clicar na categoria criada poderá verificar que aparece o bloco criado (Figura 41).

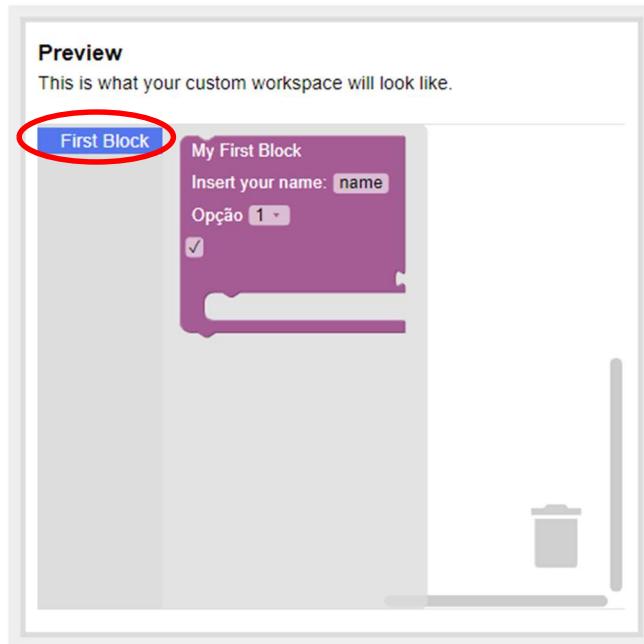


Figura 41 - Pré-visualização do *Workspace*

6. Quando finalizar o seu *workspace* pode transferi-lo clicando em “*Export*” e de seguida em “*Toolbox*”, posteriormente também poderá fazer o seu *upload* através de “*Load to Edit*” (Figura 42)

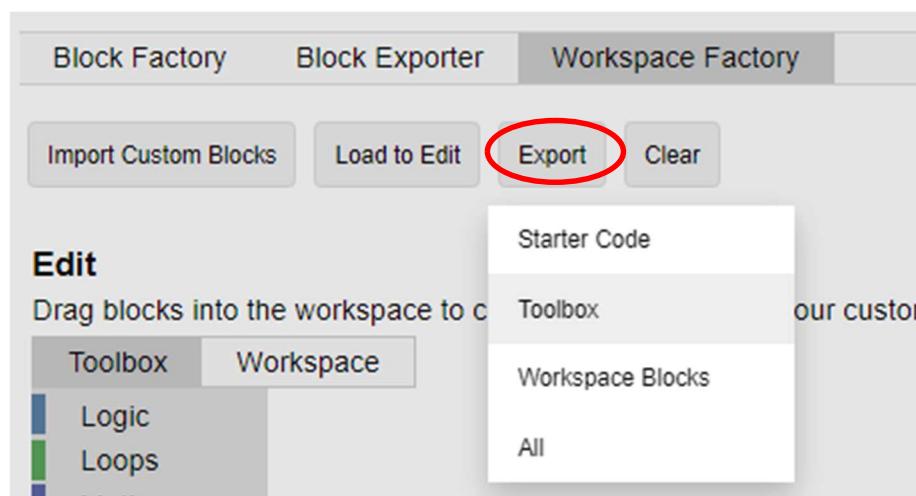
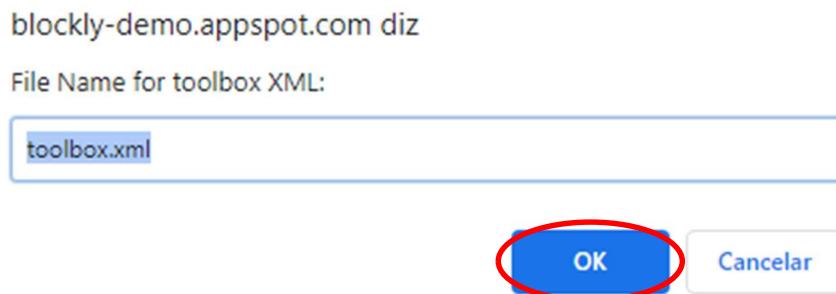


Figura 42 - Export *Workspace*

7. Atribua um nome ao ficheiro do seu *workspace* e clique em “OK” para fazer o seu *download* (Figura 43).



**Figura 43 - Download do Workspace**

## 5. Construção da Página html

Com os blocos e o *workspace* criados, o próximo passo é a integração desses mesmos blocos numa página html, para isso e a título de exemplo, nos próximos passos irá editar uma página contida no projeto do *Blockly*.

1. Comece por aceder ao *GitHub* do projeto através do seguinte *link*:  
<https://github.com/google/blockly>.
2. Faça o *download* do repositório acedendo a “Code” e de seguida a “Download ZIP” (Figura 44).

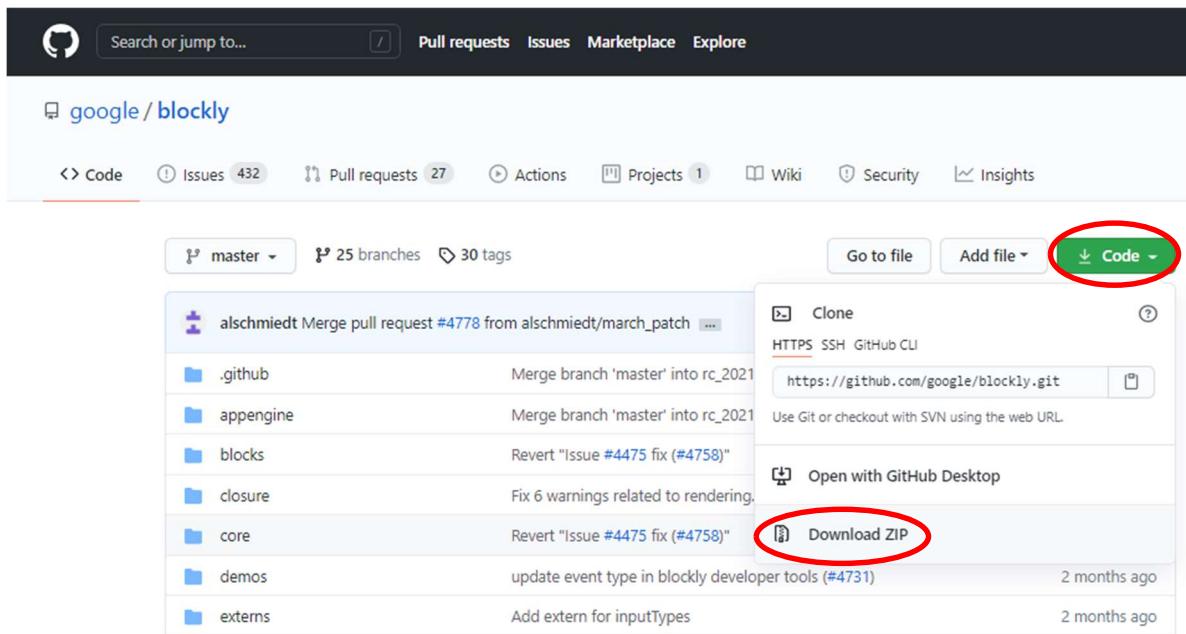


Figura 44 - *Download* do Repositório

3. Quando o *download* estiver concluído, descompacte a pasta descarregada.

4. Aceda à pasta descompactada, de seguida à pasta “*generators*” e depois à pasta “*javascript*” e coloque nesta pasta o ficheiro que descarregou anteriormente com o código que o bloco irá gerar (Figura 45).

Este PC > Transferências > blockly-master > generators > javascript		
	Nome	Data de modificação
Árvore	colour.js	18/05/2021 11:25
Meu trabalho	Generator.js	18/05/2021 11:26
Transferências	lists.js	13/04/2021 21:09
Downloads	logic.js	13/04/2021 21:09
	loops.js	13/04/2021 21:09
	math.js	13/04/2021 21:09
	procedures.js	13/04/2021 21:09
	text.js	13/04/2021 21:09
	variables.js	13/04/2021 21:09
	variables_dynamic.js	13/04/2021 21:09

Figura 45 - Pasta "generators"

5. Recue para a pasta do repositório e na pasta “*blocks*” coloque o outro ficheiro que descarregou anteriormente com as definições dos blocos criados (Figura 46).

Este PC > Transferências > blockly-master > blocks		
	Nome	Data de modificação
		Tipo
Árvore	Block_definition.js	17/05/2021 18:48
Meu trabalho	colour.js	Ficheiro JavaScript
Transferências	lists.js	Ficheiro JavaScript
Downloads	logic.js	Ficheiro JavaScript
	loops.js	Ficheiro JavaScript
	math.js	Ficheiro JavaScript
	procedures.js	Ficheiro JavaScript
	text.js	Ficheiro JavaScript
	variables.js	Ficheiro JavaScript
	variables_dynamic.js	Ficheiro JavaScript

Figura 46 - Pasta "blocks"

6. Abra esse mesmo ficheiro com o seu editor de texto e, insira uma nova linha no início do ficheiro e coloque o código da Caixa de Texto 1. Insira uma nova linha no final do ficheiro e coloque o seguinte texto “);” de modo a ficar com o aspeto da Figura 47 e da Figura 48.

```
Blockly.defineBlocksWithJsonArray(
```

#### Caixa de Texto 1 - Código a Adicionar

```
playground.html      Generator.js      Block_definition.js X
blocks > Block_definition.js > args0 > options
1 Blockly.defineBlocksWithJsonArray(
2 [
3   {
4     "type": "first_block",
5     "message0": "My First Block %1 Insert your name: %2 %3 %4 %5 %6 %7 %8 %9",
6     "args0": [
7       {
8         "type": "input_dummy"
9       },
10      {
11        "type": "field_input",
12        "name": "var",
13        "text": "name"
```

Figura 47 - Início do Ficheiro

```
playground.html      Generator.js      Block_definition.js X
blocks > Block_definition.js > args0 > options
46   {
47     "type": "input_value",
48     "name": "value_input"
49   },
50   {
51     "type": "input_statement",
52     "name": "statement_input"
53   }
54 ],
55 "previousStatement": null,
56 "nextStatement": null,
57 "colour": 315,
58 "tooltip": "",
59 "helpUrl": ""
60 };
61 );
```

Figura 48 - Final do Ficheiro

7. Vá novamente para a pasta do repositório e agora aceda à pasta “*tests*” e de seguida à pasta “*xml*” e substitua o ficheiro “*toolbox.xml*” por o ficheiro anteriormente descarregado com as informações do seu *workspace*.
8. Agora, na pasta “*tests*” abra o ficheiro “*playground.html*” com o seu editor de texto preferido. Nas linhas iniciais, logo após “*<script src=“..../generators/javascript.js”></script>*” coloque o código da Caixa de Texto 2 de modo a ficar com o aspeto da Figura 49. Deve alterar o nome do ficheiro para o nome que atribuiu anteriormente.

```
<script src="..../generators/Generator.js"></script>
```

#### Caixa de Texto 2 - Código a Adicionar

```
playground.html ×
tests > playground.html > html > head > script
1   <!DOCTYPE html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Blockly Playground</title>
6     <script src="..../blockly_uncompressed.js"></script>
7     <script src="..../generators/javascript.js"></script>
8     <script src="..../generators/javascript/Generator.js"></script> (Line 8)
9     <script src="..../generators/javascript/logic.js"></script>
10    <script src="..../generators/javascript/loops.js"></script>
11    <script src="..../generators/javascript/math.js"></script>
```

Figura 49 - Adição de *Script*

9. Ainda na mesma zona do ficheiro “*playground.html*”, coloque o código da Caixa de Texto 3 logo após ““*<script src=“..../msg/messages.js”></script>*” de forma a ficar como aspeto da Figura 50. Deve alterar o nome do ficheiro para o nome que atribuiu anteriormente.

```
<script src="..../blocks/Block_definition.js"></script>
```

#### Caixa de Texto 3 - Código a Adicionar

```

tests > playground.html > html > head > script
  56   <script src="../generators/dart/variables_dynamic.js"></script>
  57   <script src="../generators/dart/procedures.js"></script>
  58   <script src="../msg/messages.js"></script>
  59   <script src="../blocks/Block_definition.js"></script> (highlighted)
  60   <script src="../blocks/logic.js"></script>
  61   <script src="../blocks/loops.js"></script>

```

**Figura 50 – Adição de Script**

- Ainda no ficheiro “playground.html”, faça scroll até ao final do ficheiro (linha 464 aproximadamente) e substitua todas as linhas de código, desde a linha que contém o código da Figura 51 até à linha que contém o código apresentado na Figura 52 (neste caso em concreto será da linha 464 à linha 1172), pelo código contido no ficheiro que descarregou com as informações do workspace que criou. Ficando, no final, com o aspetto da Figura 53.

```

464 | <xml xmlns="https://developers.google.com/blockly/xml" id="toolbox-simple" style="display: none">
465 |   <block type="controls_ifelse"></block>
466 |   <block type="logic_compare"></block>

```

**Figura 51 - Código Inicial**

```

1169 |   <sep></sep>
1170 |   <category name="Variables" categorystyle="variable_category" custom="VARIABLE_DYNAMIC"></category>
1171 |   <category name="Functions" categorystyle="procedure_category" custom="PROCEDURE"></category>
1172 | </xml>

```

**Figura 52 - Código Final**

```

467 | <!-- toolbox-simple is an always-open flyout with no category menu.
468 | Always-open flyouts are a good idea if you have a small number of blocks. -->
469 | <xml xmlns="https://developers.google.com/blockly/xml" id="toolbox" style="display: none">
470 |   <category name="First Block">
471 |     <block type="first_block">
472 |       <field name="var">name</field>
473 |       <field name="selecao">opcao1</field>
474 |       <field name="check">TRUE</field>
475 |     </block>
476 |   </category>
477 | </xml>
478 | </body>
479 | </html>

```

**Figura 53 - Aspetto Final**

11. No mesmo ficheiro, procure pela função “*function start()*”, deverá estar aproximadamente na linha 84 e elimine as linhas de código selecionadas e guarde o ficheiro (Figura 54).

```

playground.html X
tests > playground.html > html > head > script > start
84  function start() {
85      setBackgroundColour();
86      // Parse the URL arguments.
87      var match = location.search.match(/dir=([^&]+)/);
88      var rtl = match && match[1] == 'rtl';
89      document.forms.options.elements.dir.selectedIndex = Number(rtl);
90      var toolbox = getToolboxElement();
91      setToolboxDropdown();
92      match = location.search.match(/side=([^&]+)/);
93      var autoimport = !location.search.match(/autoimport=([^&]+)/);
94      // Create main workspace.
95      var match = 'ltr';
96      var rtl = match && match[1] == 'rtl';
97      var autoimport = 'ltr';
98      workspace = Blockly.inject('blocklyDiv',

```

Figura 54 – “*function start()*”

12. Abra o ficheiro “*playground.html*” no seu *browser* e repare que agora contém o bloco que criou (Figura 55).

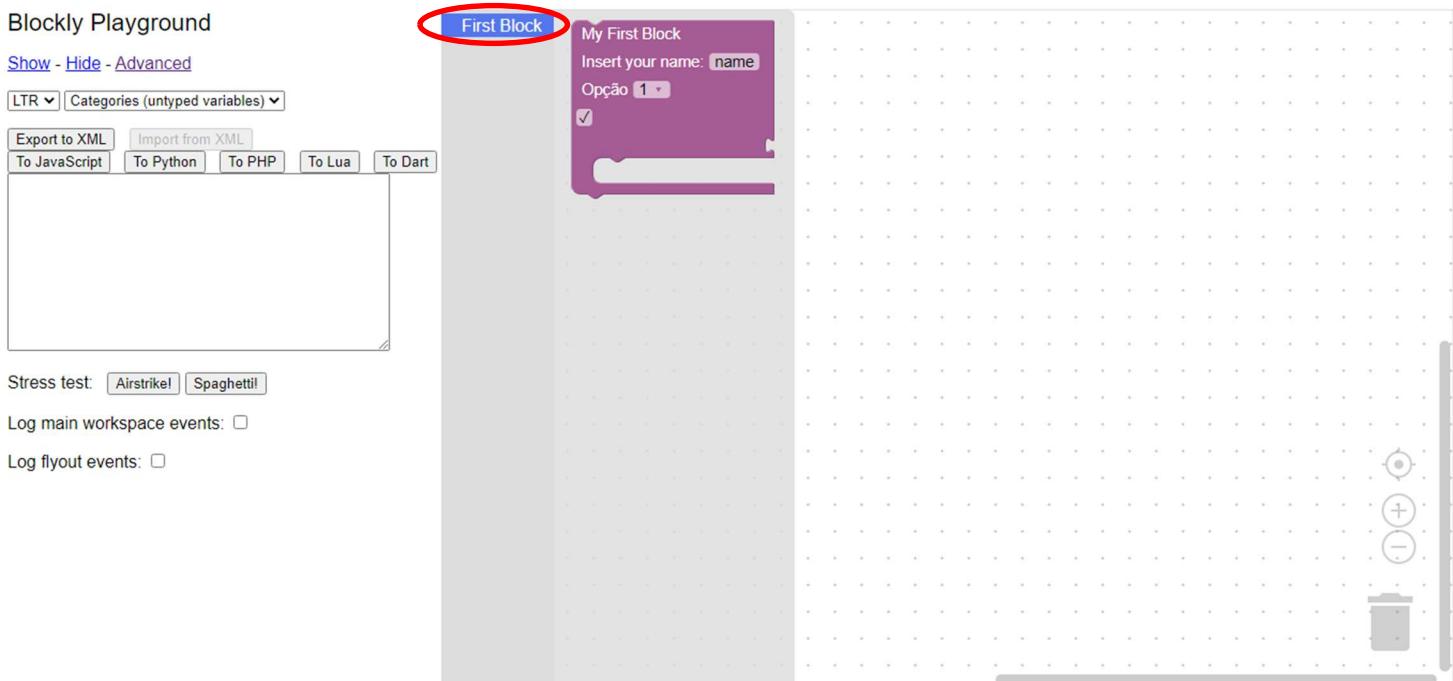


Figura 55 - Aspetto da Página

13. Neste momento, os blocos que criou já surgem na página html, agora é necessário adicionar o código que pretende gerar, para isso, abra o ficheiro que contém o código a gerar pelo bloco criado, colocado anteriormente em “`|blockly-master|generators|javascript`”. Todo o código que desejar que o bloco gere deverá ser colocado na variável “`code`” (Figura 56). A título de exemplo irá adicionar o código que permitirá visualizar as variáveis do próprio bloco, para que o conteúdo de uma variável seja escrito deve colocar a variável no seguinte formato “`‘+nome_da_variavel+’`”, para verificar se a `checkbox` está acionada basta utilizar a função “`if`”. Posto isto, adicione o código da Caixa de Texto 4 à variável “`code`” do seu bloco de modo a ficar com o aspeto da Figura 57 e guarde o ficheiro. Note que deverá alterar o nome das variáveis para o nome que atribuiu anteriormente.

```

playground.html      Generator.js X  Block_definition.js
generators > javascript > Generator.js ...
1  Blockly.JavaScript['first_block'] = function(block) {
2    var text_var = block.getFieldValue('var');
3    var dropdown_selecao = block.getFieldValue('selecao');
4    var checkbox_check = block.getFieldValue('check') == 'TRUE';
5    var value_value_input = Blockly.JavaScript.valueToCode(block, 'value_input', Blockly.JavaScript.ORDER_ATOMIC);
6    var statements_statement_input = Blockly.JavaScript.statementToCode(block, 'statement_input');
7    // TODO: Assemble JavaScript into code variable.
8    var code = '...;\n';
9    return code;
10 };

```

Figura 56 - Variável “code”

```

<'Texto Inserido: '+text_var+'\n'
+'Opção Selecionada: '+dropdown_selecao+'\n';
if (checkbox_check){
  code = code + 'Checkbox: Checked\n';
}
else{
  code = code + 'Checkbox: Unchecked\n';
}
code = code + ''+value_value_input+'\n'
+'+statements_statement_input+';
```

Caixa de Texto 4 - Código a Adicionar

```

playground.html  Generator.js X  Block_definition.js
generators > javascript > Generator.js > ...
1  Blockly.JavaScript['first_block'] = function(block) {
2      var text_var = block.getFieldValue('var');
3      var dropdown_selecao = block.getFieldValue('selecao');
4      var checkbox_check = block.getFieldValue('check') == 'TRUE';
5      var value_value_input = Blockly.JavaScript.valueToCode(block, 'value_input', Blockly.JavaScript.ORDER_ATOMIC);
6      var statements_statement_input = Blockly.JavaScript.statementToCode(block, 'statement_input');
7      // TODO: Assemble JavaScript into code variable.
8      var code = 'Texto Inserido: '+text_var+'\n'
9      +'Opção Selecionada: '+dropdown_selecao+'\n';
10     if (checkbox_check){
11         code = code + 'Checkbox: Checked\n';
12     }
13     else{
14         code = code + 'Checkbox: Unchecked\n';
15     }
16     code = code + ''+value_value_input+'\n'
17     +' '+statements_statement_input+'';
18     return code;
19 };

```

Figura 57 - Código Final

14. Abra novamente no seu *browser* o ficheiro “*playground.html*”, arraste o bloco criado para o *workspace*, altere os campos do bloco e clique no botão “*To JavaScript*” de forma a gerar o código (Figura 58).

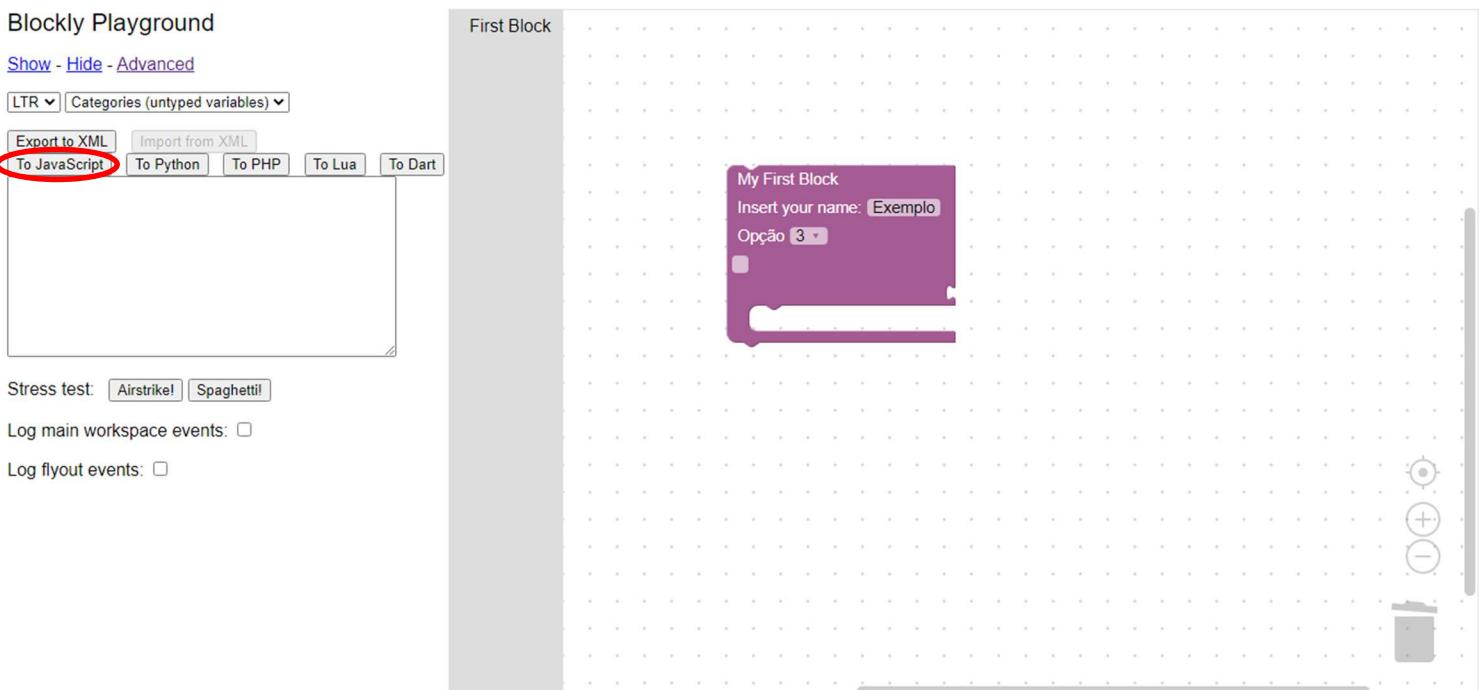
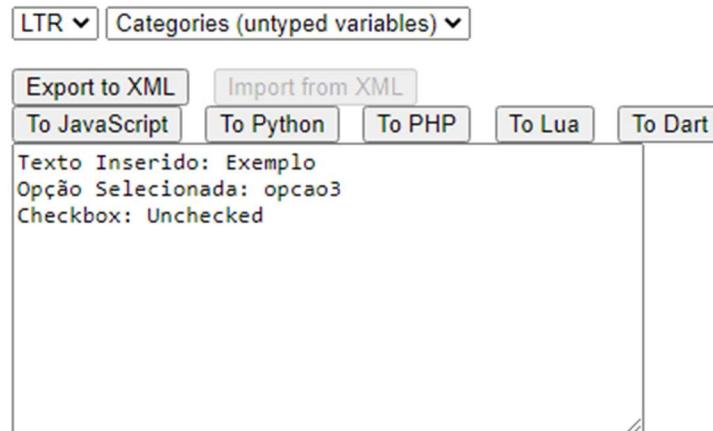


Figura 58 - Gerar Código

15. Como pode verificar no código gerado é apresentado o conteúdo que colocou nos campos do bloco, é apresentado o conteúdo de todas as variáveis do bloco à exceção das variáveis dos encaixes, já que não existem nada encaixado no bloco (Figura 59).

### Blockly Playground

[Show](#) - [Hide](#) - [Advanced](#)



**Figura 59 - Código Gerado**