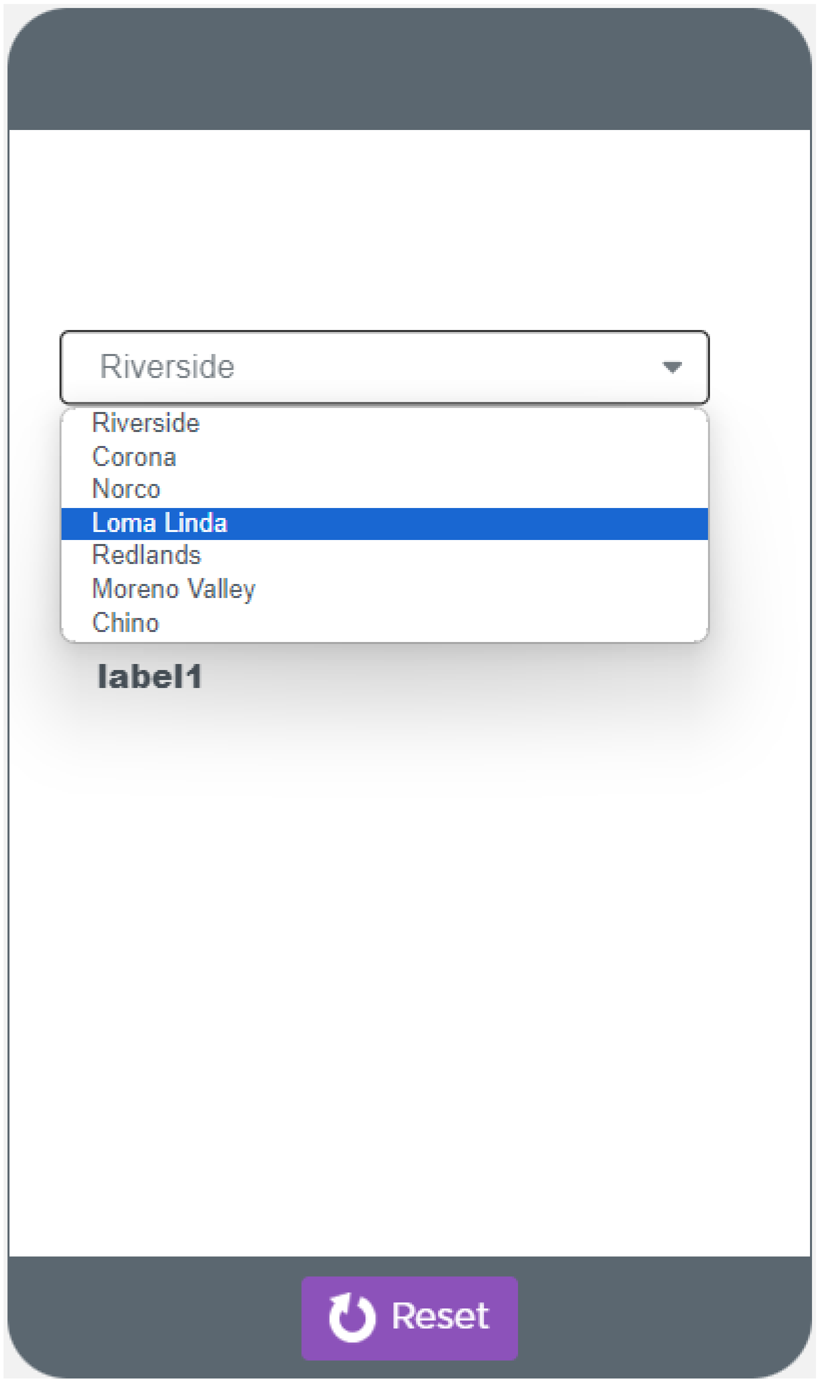
List

1. Introduction

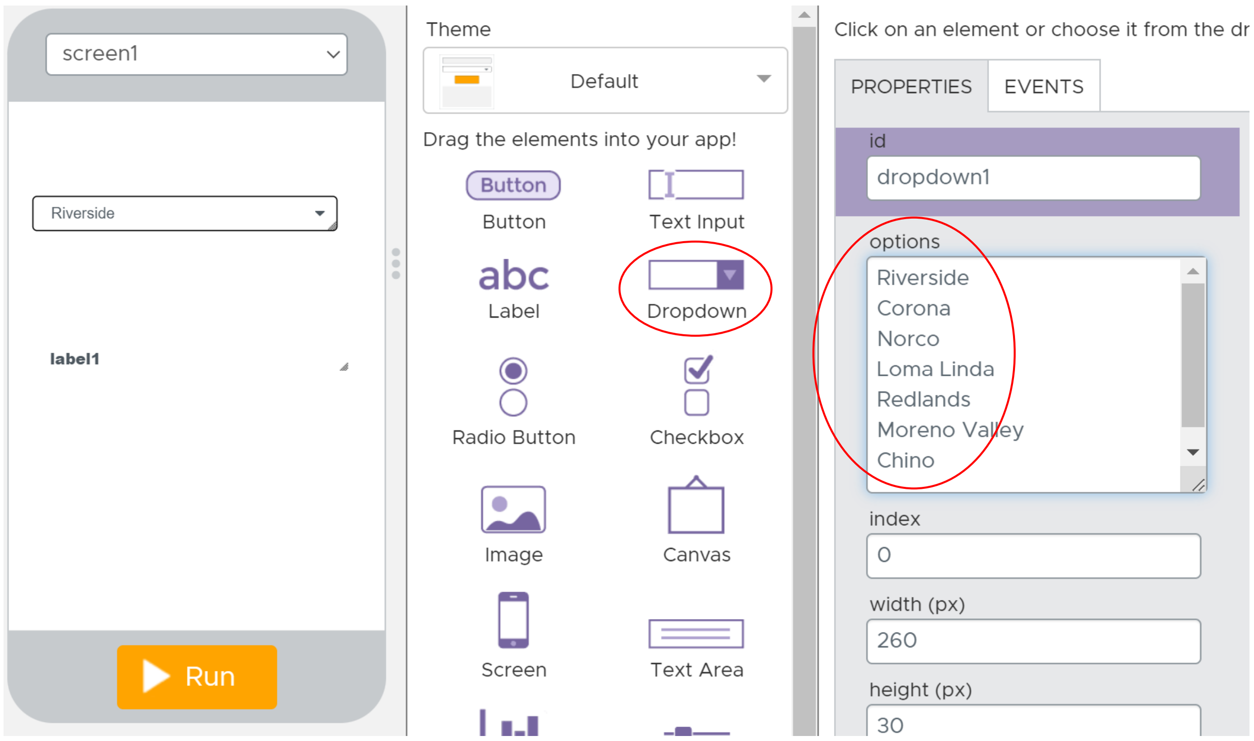
Lists are for displaying lots of similar items for the user to select from.



2. Static list

A static list is one where the items in the list are fixed during the initial design and does not change during the program run.

To create a static list, you just type in the items that you want for the list in the *options* property of the Dropdown list.



id: dropdown1

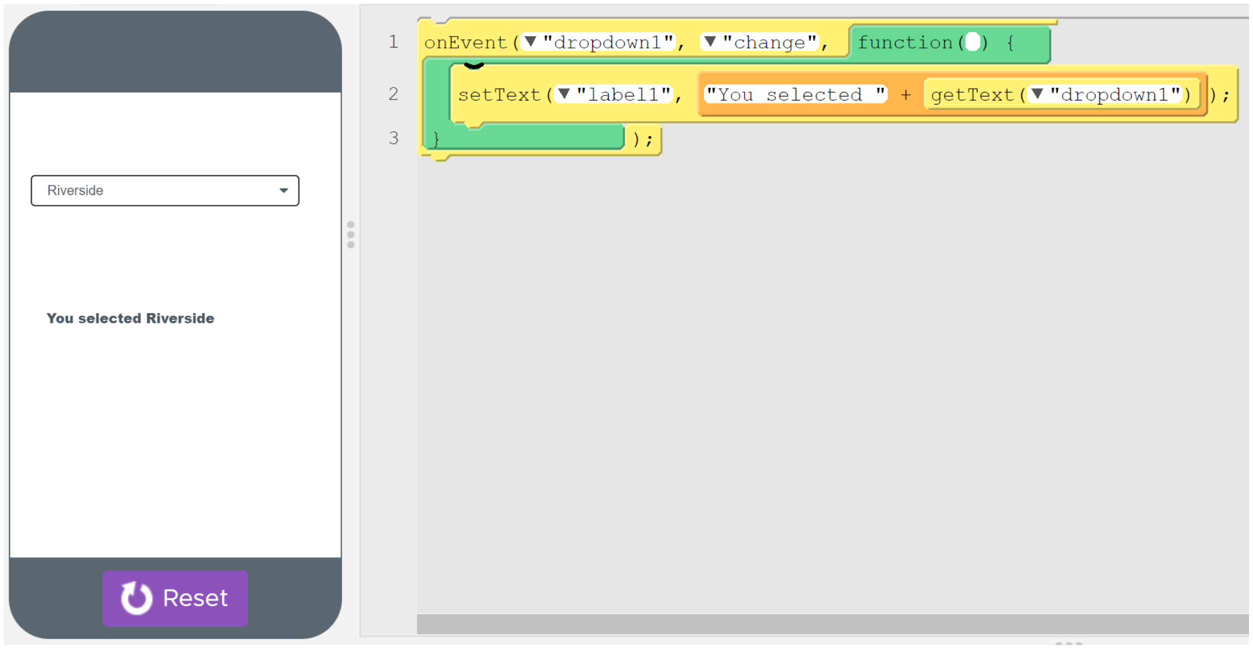
options:
Riverside
Corona
Norco
Loma Linda
Redlands
Moreno Valley
Chino

index: 0

width (px): 260

height (px): 30

Use the **getText** command to see what the user has selected as shown next.

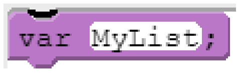


```
onEvent("dropdown1", "change", function( ) {
  setText("label1", "You selected " + getText("dropdown1"));
});
```

3. Dynamic list

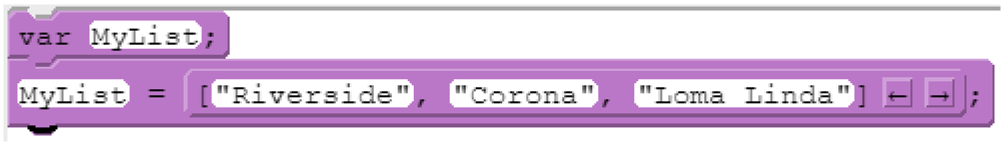
A dynamic list is one where the items in the list can change during the program run. Since the list is not fixed, you'll need to use a variable to remember the items in the list. Then you'll use the contents stored in this variable to populate the items in the Dropdown list. Note that you have two different "lists", so don't get confused about which is which. The first is the list variable for storing the items, and the second is the Dropdown list for displaying the items.

To manipulate a list variable, you first create a variable. Give it a meaningful name.



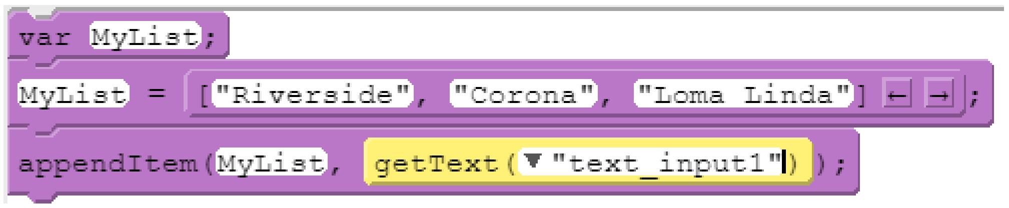
```
var MyList;
```

You can optionally initialize the list with some fixed items.



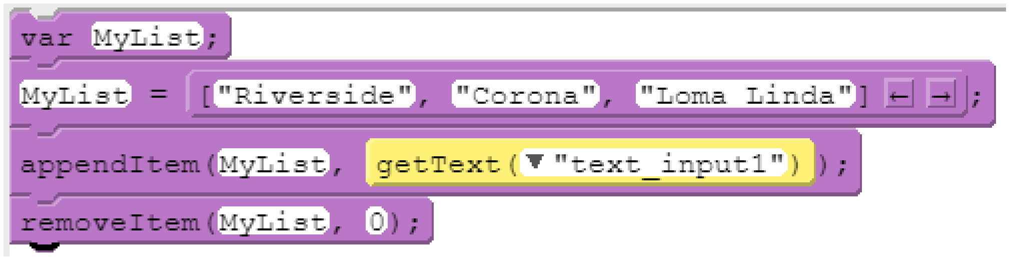
```
var MyList;
MyList = ["Riverside", "Corona", "Loma Linda"];
```

You can append more items to the list using the **appendItem** command. You typically would get the text that you want to append from a text input box. Furthermore, instead of the example below, you should put the **appendItem** command inside an **onEvent** block for a button click.



```
var MyList;
MyList = ["Riverside", "Corona", "Loma Linda"];
appendItem(MyList, getText("text_input1"));
```

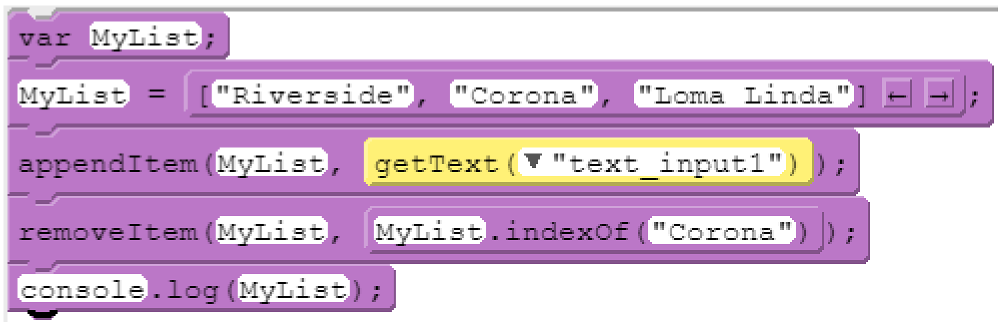
You can remove an item from the list using the **removeItem** command. Again, you should put the **removeItem** command inside an **onEvent** block for a button click instead of what's shown below.



```
var MyList;
MyList = ["Riverside", "Corona", "Loma Linda"];
appendItem(MyList, getText("text_input1"));
removeItem(MyList, 0);
```

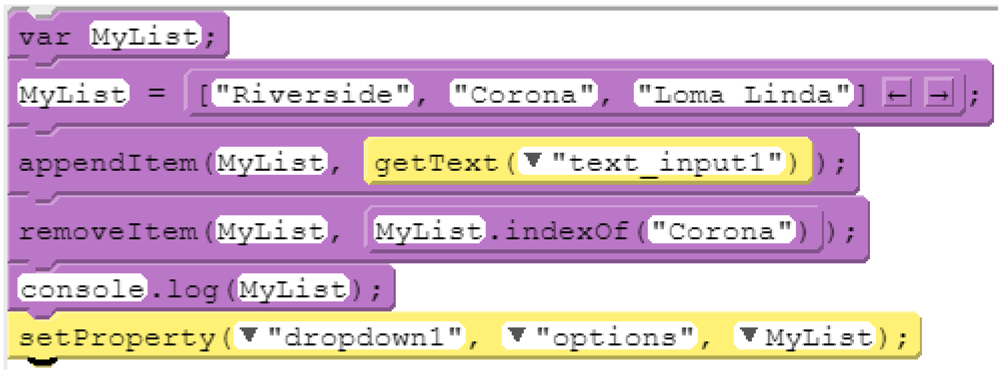
The command requires you to provide the index of the item that you want to remove. The above example removes item 0 which is Riverside.

If you do not know the index number of the item that you want to remove then you can use the **indexOf** command to find the index. This command searches for the given string in the list. If it is found, it returns the index of where that string is. If it is not found then it returns -1. The next example removes Corona from the list and then prints out the resulting list in the Debug console using the **console.log** command.



```
var MyList;
MyList = ["Riverside", "Corona", "Loma Linda"];
appendItem(MyList, getText("text_input1"));
removeItem(MyList, MyList.indexOf("Corona"));
console.log(MyList);
```

Finally, to populate and display the Dropdown list with the items that are stored in the list variable, you use the **setProperty** command to set the **options** property using the list variable as shown next.

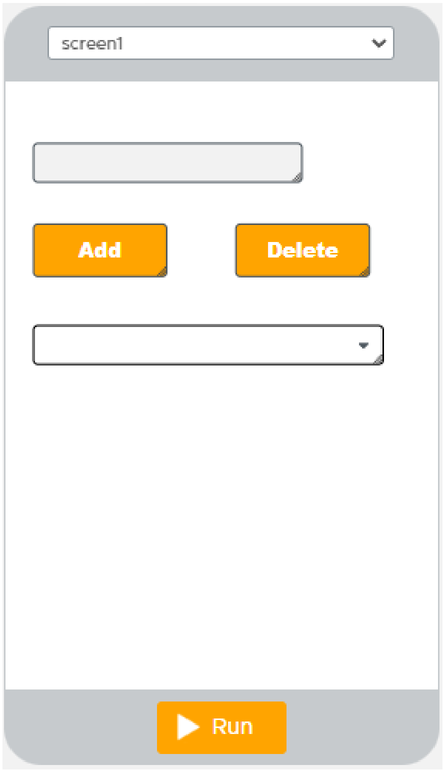


```
var MyList;
MyList = ["Riverside", "Corona", "Loma Linda"];
appendItem(MyList, getText("text_input1"));
removeItem(MyList, MyList.indexOf("Corona"));
console.log(MyList);
setProperty("dropdown1", "options", MyList);
```

To keep the Dropdown list updated after each operation (append or remove), you'll need to execute the **setProperty** command every time after you add or remove an item from the list variable.

Problems (Questions with an * are more difficult)

1) Create the following user interface with a text input box, two buttons, and a dropdown list.



2) Add in the functionality for adding items to the list. The user first types an item name in the textbox and then press the Add button. The item from the textbox is then added to the list. Items added to the list must be displayed correctly in the dropdown list.

3) Add in the functionality for deleting items from the list. The user first types an item name in the textbox and then press the Delete button. The item from the textbox is then deleted from the list. The dropdown list must show the resulting list with the item removed.

4) Continuing with problem 3), if the item is not found in the current list, then display a message saying "Item not in list" and no deletion occurs.

5) * In addition to adding items to the list from problem 2), add them also into a database table.

6) * In addition to deleting items from the list from problem 3), delete them also from the database table.

7) ** Populate the dropdown list with the records stored in the database table when you first start up the app.

8) Modify problem 3) to delete the item from the selected name in the dropdown list instead of the name typed in the textbox.

9) ** Add a change operation. The user first selects an item from the dropdown list, and types a new name in the textbox. A change button click will replace the selected item from the dropdown list with the new name typed in.