Toast For Unity Documentation

Hi, Welcome To Toast For Unity Documentation! ☀️

WebGL Demo | **Discord** | **Asset Store Link**

Online Documentation (Prefer, this will update more faster than offline documentation)

Toast For Unity is a message alert pop-out plugin for unity.

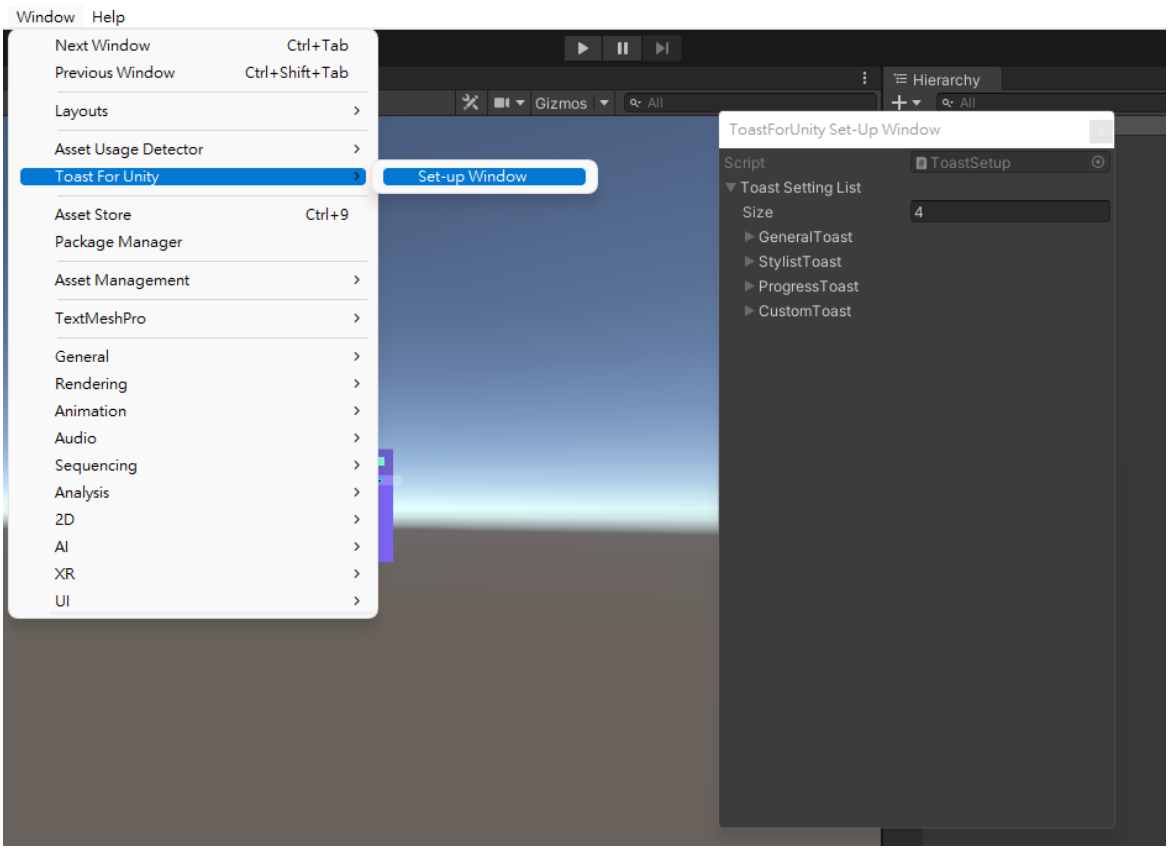
Our purposes is let you quickly pop out a beautiful message box in everywhere and in a second.

Since the core class is write as a **static** function, you can call it everywhere as the example code below:

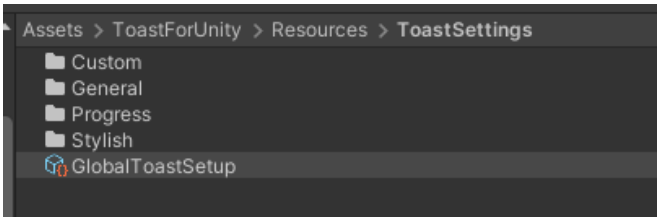
```
using ToastForUnity.Script.Core;

private void Start(){
   Toast.PopOut("Hello World");
}
```

▼ How it works?

1. Go to [Window/Toast For Unity/ Set-up Window] to open the Toast For Unity Set-up Window.

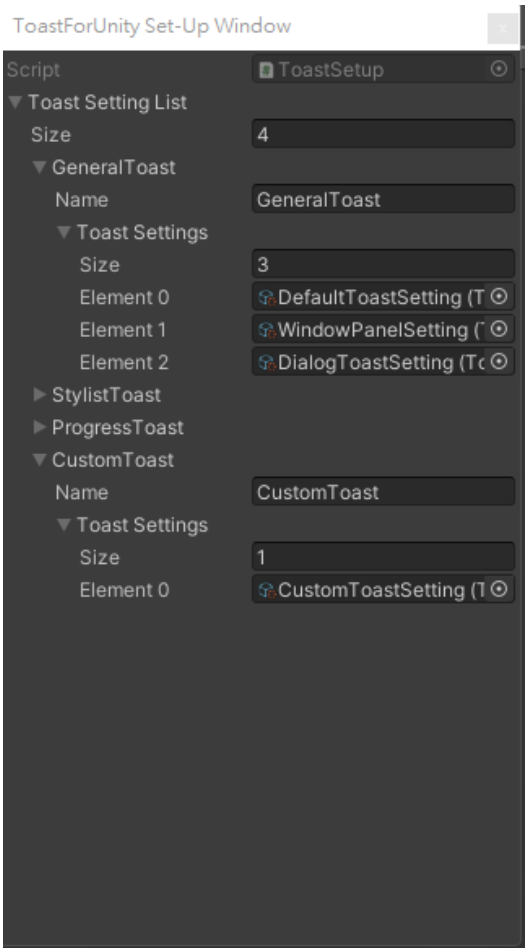


You can also locate this at [ToastForUnity/Resources/ToastSettings] for the version before 1.0.2

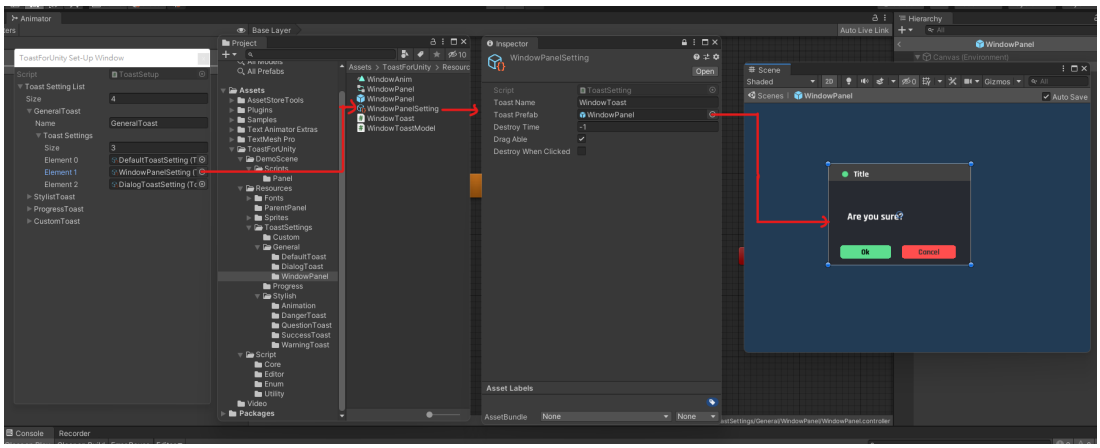


3. The Toast Set-up Window already contain some pre-made toast to directly use with.

   You can also insert your own custom toast by manually increase the Size.

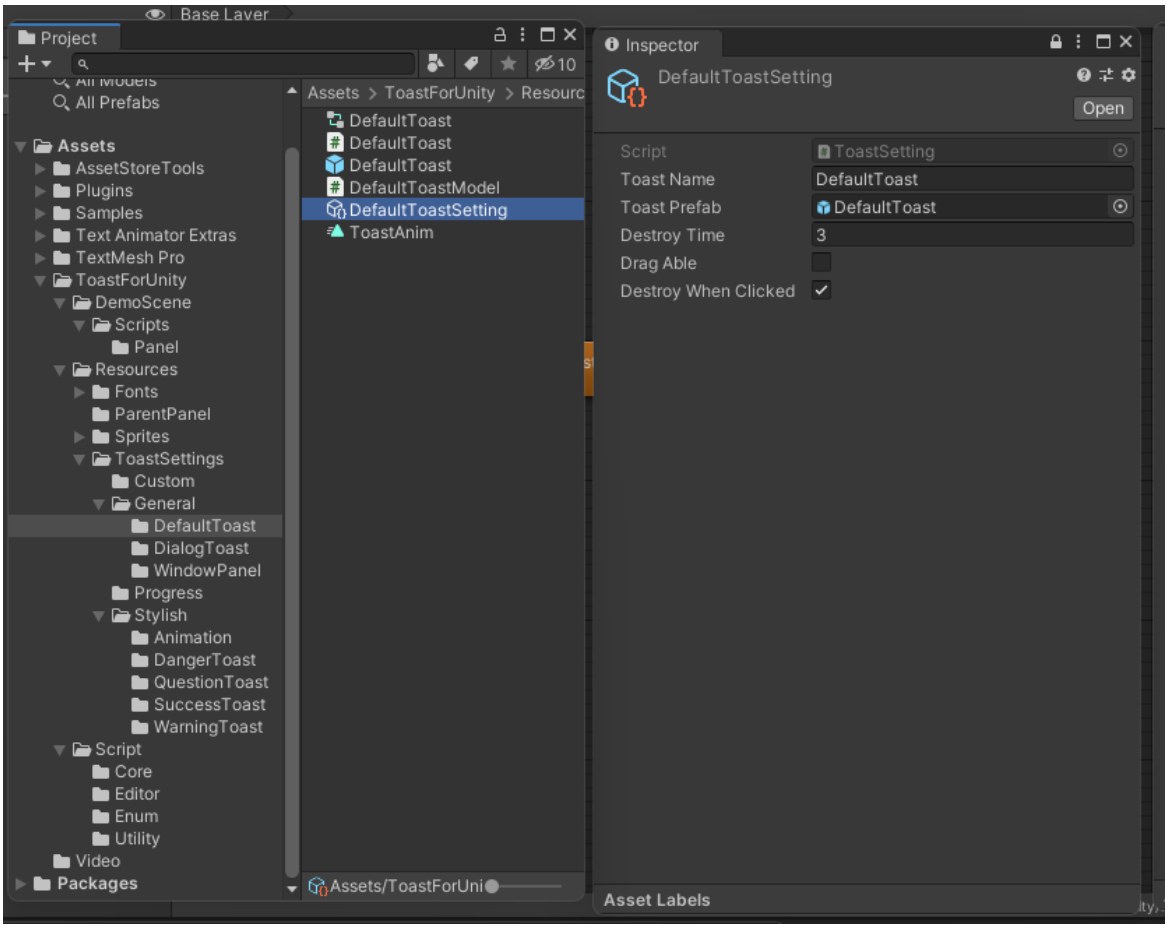


2. In the Toast Set-up window, you can easily find and locate all the reference toast object.

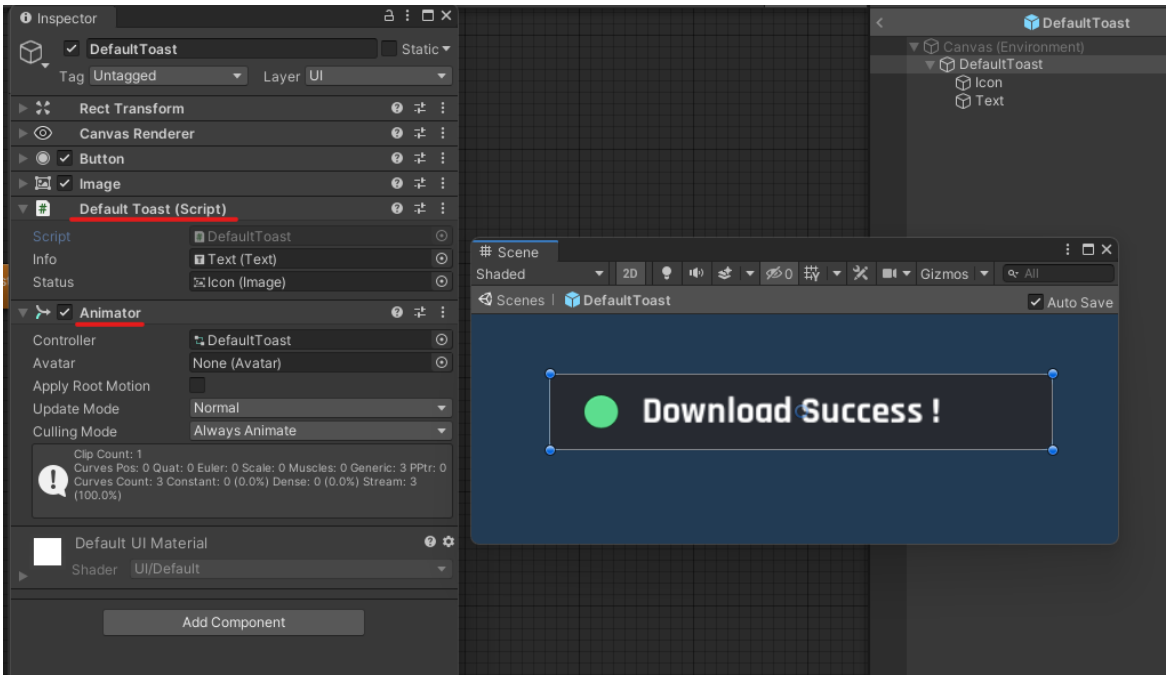


3. Each of the **Toast Setting** is contain the structure below:

```
- Toast Name ⇒ The name of this toast (it should be an unique name)
- Toast Prefab ⇒ The prefab of this toast (prefab that need to spawn)
- Destroy Time ⇒ Destroy this gameObject after second (type -1 for not destroy)
- Drag Able ⇒ Trigger for this toast is draggable
- Destroy When Clicked ⇒ Destroy this toast when mouse clicked.
```



3. **Toast Prefab**



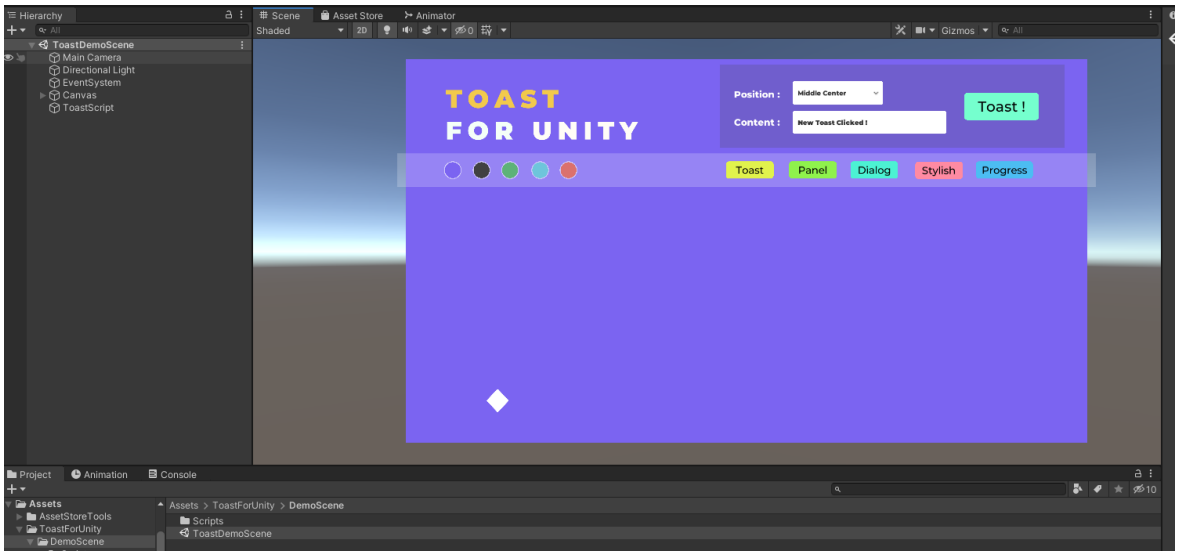
Toast prefab is contain a toast view script which inherent by `ToastPrefabBase` class, and a custom animator that will be show out when prefab instantiated. That's all for the basic knowledge, you are able to change the prefab style just like changing uGUI.

Check **here** For more information of how to create a custom toast.

▼ Demo Scene:

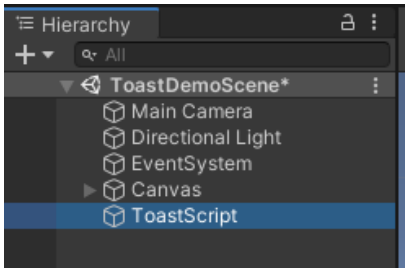
You can try the demo scene to see the toast result.

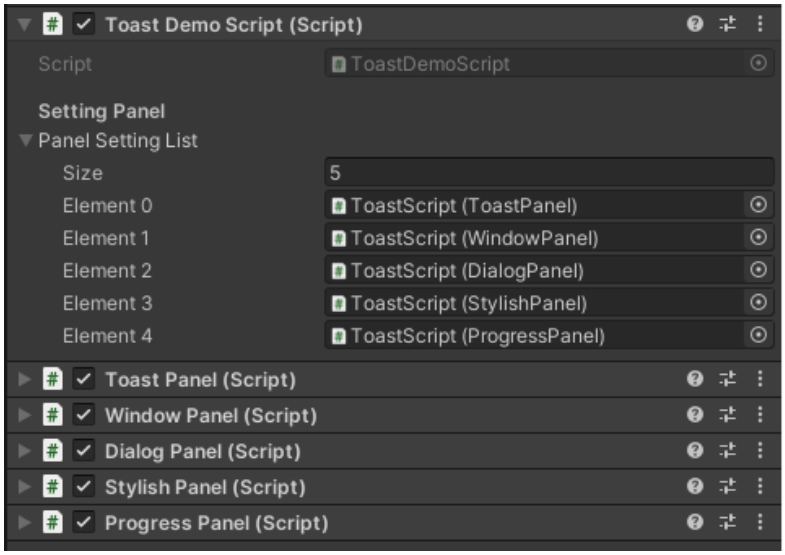
Also understand how to call the Toast function here.



[ToastForUnity/DemoScene/ToastDemoScene]

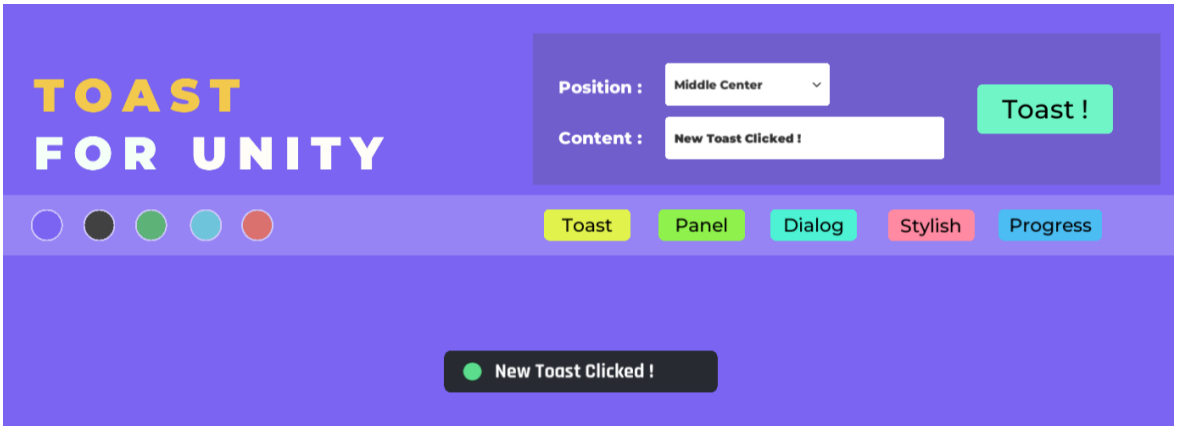
The main script is place on **ToastScript** game object, which contains 5 different example.





5 Different Official Toast Example:

▼ 1. Toast Panel



Example Demo Code:

```
public class ToastPanel: MonoBehavior
{
    // UI Variable
    public InputField ContentInput;
    public Button ToastBtn;

    private void Start()
    {
        // Add Button Click Listener
        ToastBtn.onClick.AddListener(ToastPop);
    }

    // Show Toast Display
    private void ToastPop()
    {
        // The Simplest Function To Work
        Toast.PopOut(ContentInput.text);

        // With More Parameters Options
        Toast.PopOut(ContentInput.text, ToastStatus.Success,
            ParentPrefab.GetParent((ToastPosition)PositionDropdown.value));
    }
}
```

▼ API Usage Summary:

Basic API:

```
Toast.PopOut(string message);
```

API With Status:

```
Toast.PopOut(string message, ToastStatus status);
```

API With Status and Parent Position:

```
Toast.PopOut(string message, ToastStatus status, Transform parentTransform);
```

API With Color:

```
Toast.PopOut(string message, Color color);
```

API With Color and Parent Position:

```
Toast.PopOut(string message, Color color, Transform parentTransform);
```

API Setting Default Parent:

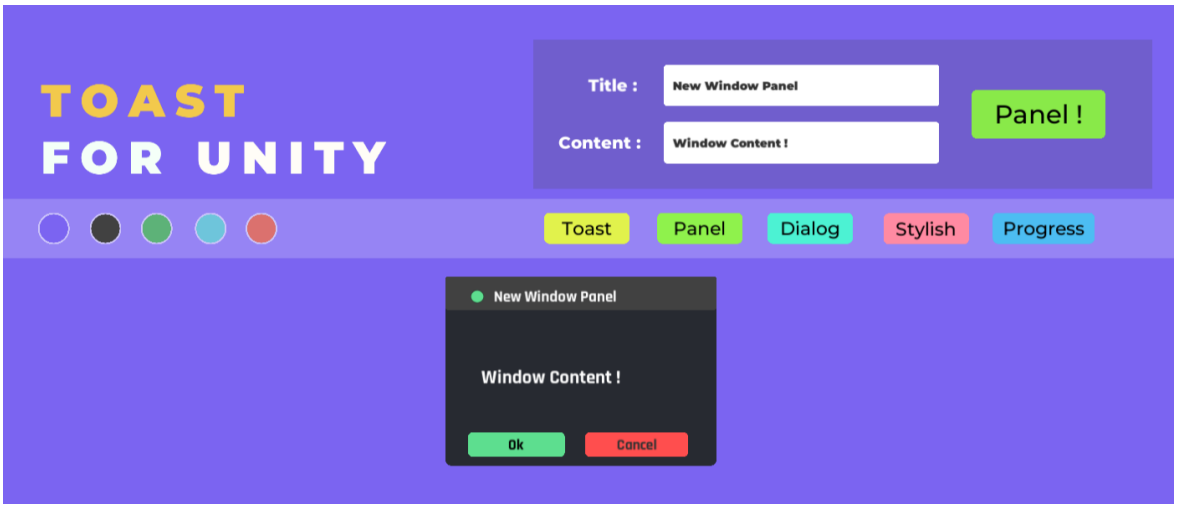
(In-case you don’t want to set the parent every time, you can set this for the first time)

```
Toast.SetDefaultParent(Transform Parent);
```

▼ **Example Code Snippets:**

```
using ToastForUnity.Script.Core;

private void SomeFunction(){

  // Set Default Parent
  Toast.SetDefaultParent(Parent);

  //Most Simplest Toast Function
  Toast.PopOut("Hello");

  //Toast With Given Parent
  Toast.PopOut("Hello", Parent.transform);

  //Toast With Different Status
  Toast.PopOut("Hello", status.Success, Parent.transform);

  //Toast With Custom Color
  Toast.PopOut("Hello", Color.red, Parent.transform);
}
```

▼ 2. Window Panel



Example Demo Code:

```
public class WindowPanel : MonoBehaviour
{
   //UI Variable
   public InputField TitleInput;
   public InputField ContentInput;
   public Button WinPanelBtn;

   private void Start(){
     //Button onClick Listener
     WinPanelBtn.onClick.AddListener(PanelPop);
   }

    private void PanelPop(){

      // Usage of Window Pop Out, the parameter is a Window Toast Model
      Toast.WindowPopOut(new WindowToastModel(){
                Title = TitleInput.text, // Title of window
                Content = ContentInput.text, // Content of Window
                OkBtnEvent = () =>
                {
                    //OK Delegate Event Function
                    Debug.Log("Ok is Clicked");
                },
                CancelBtnEvent = () =>
                {
                    //Cancel Delegate Event Function
                    Debug.Log("Cancel is Clicked");
                }
        });
    }
  }
```

WindowToastModel is just a simple model class inherent by a empty **ToastModelBase** class**:**

```
public class WindowToastModel : ToastModelBase
{
    public string Title;
```

```
    public string Content;
    public Action OkBtnEvent;
    public Action CancelBtnEvent;
}
```

▼ API Usage Summary:

Basic API:

```
Toast.WindowPopOut(WindowToastModel model);
```

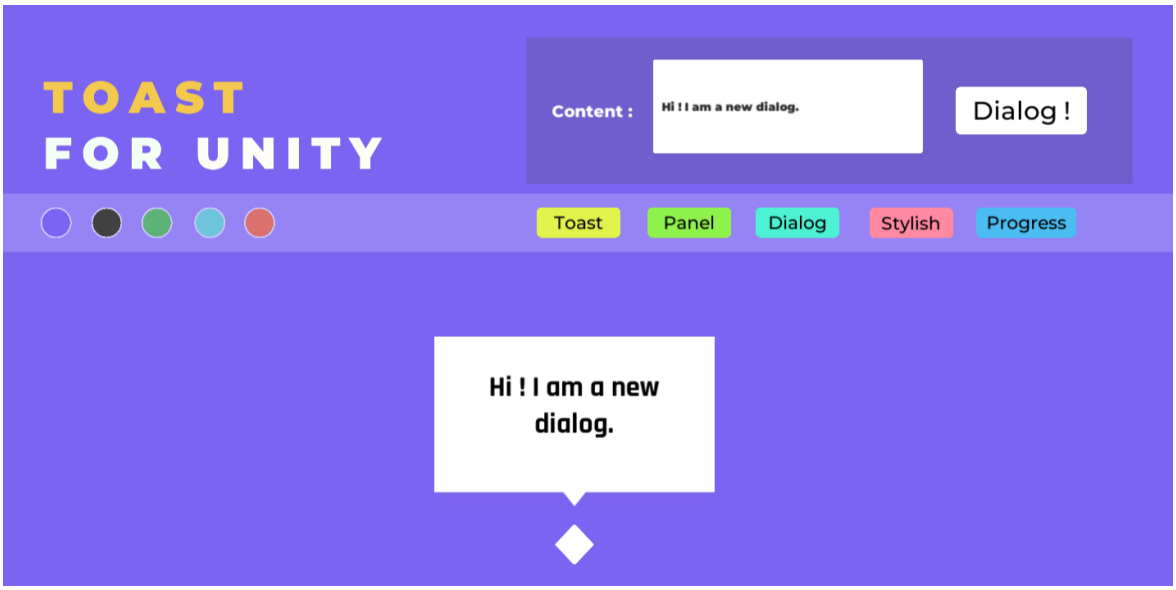
API With Parent Position:

```
Toast.WindowPopOut(WindowToastModel model, Transform parentTransform);
```

▼ Example Code Snippets:

```
using ToastForUnity.Script.Core;

private void SomeFunction(){

  Toast.WindowPopOut(new WindowToastModel()
  {
    Title = TitleInput.text,
    Content = ContentInput.text,
    OkBtnEvent = () => {Toast.PopOut("OK"));},
    CancelBtnEvent = () => {Toast.PopOut("Cancel");}
  });

}
```

▼ 3. Dialog Panel



Example Demo Code:

```
public class DialogPanel : MonoBehaviour
{
    // UI Variable
    public GameObject Parent; // this is the white geometry game object.
    public InputField ContentInput;
    public Button DialogBtn;

    private void Start()
    {
        // Add Button OnClick Listener
        DialogBtn.onClick.AddListener(DialogPop);
    }

    private void DialogPop()
    {
        // Dialog Function
        Toast.DialogPopOut(ContentInput.text, Parent.transform);
    }
}
```

▼ API Usage Summary:

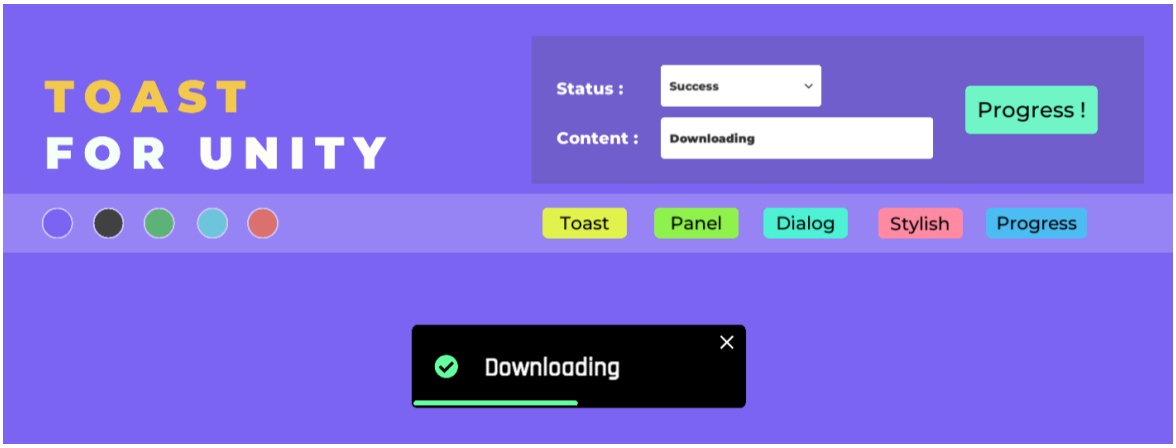
Basic API:

```
Toast.DialogPopOut( string message, Transform parentTransform);
```

▼ Example Code Snippets:

```
using ToastForUnity.Script.Core;

private void SomeFunction(){

  Toast.WindowPopOut(new WindowToastModel()
  {
    Title = TitleInput.text,
    Content = ContentInput.text,
    OkBtnEvent = () => {Toast.PopOut("OK"));},
    CancelBtnEvent = () => {Toast.PopOut("Cancel");}
  });

}
```

▼ 4. Progress Panel



Example Demo Code:

```
public class ProgressPanel : MonoBehaviour
{
    // UI Variable
    public Dropdown StatusDropdown;
    public InputField ContentInput;
    public Button ProgressBtn;

    private void Start(){
        // Add Button OnClick Listener
        ProgressBtn.onClick.AddListener(ProgressPop);
    }

    private void ProgressPop(){

        // Progress Toast Display Function
        Toast.PopOut<ProgressToastView>("ProgressToast",
            new ProgressToastModel(){
                DestroyWhenProgressComplete = true,
                ProgressDone = () =>
                {
                    Toast.PopOut("Progress Done");
                },
```

```
                ProgressRunOnStart = true,
                Status = (ProgressToastStatus)StatusDropdown.value,
                Title = ContentInput.text,
                ProgressValueChanged = null
            }
        );
    }
}
```

▼ API Usage Summary:

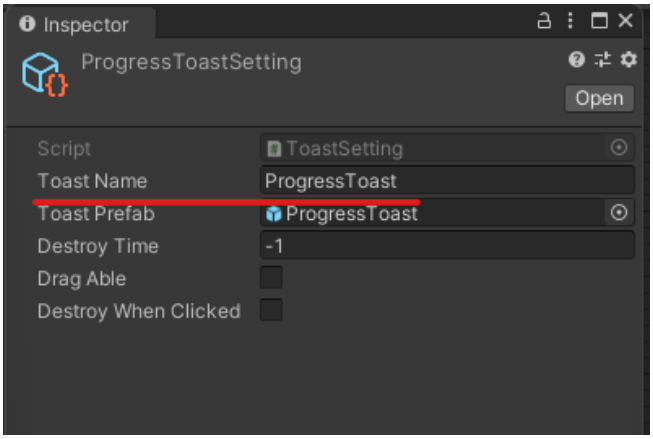
Pre-knowledge:

This is now showing how to use a common generic function to call custom toast class type, each of the toast function is declare below:

```
Toast.PopOut<ToastType>(string toastName, ToastTypeModel model);
```

Api For Progress Toast:

```
Toast.PopOut<ProgressToastView>("ProgressToast", ProgressToastModel progressModel);
```



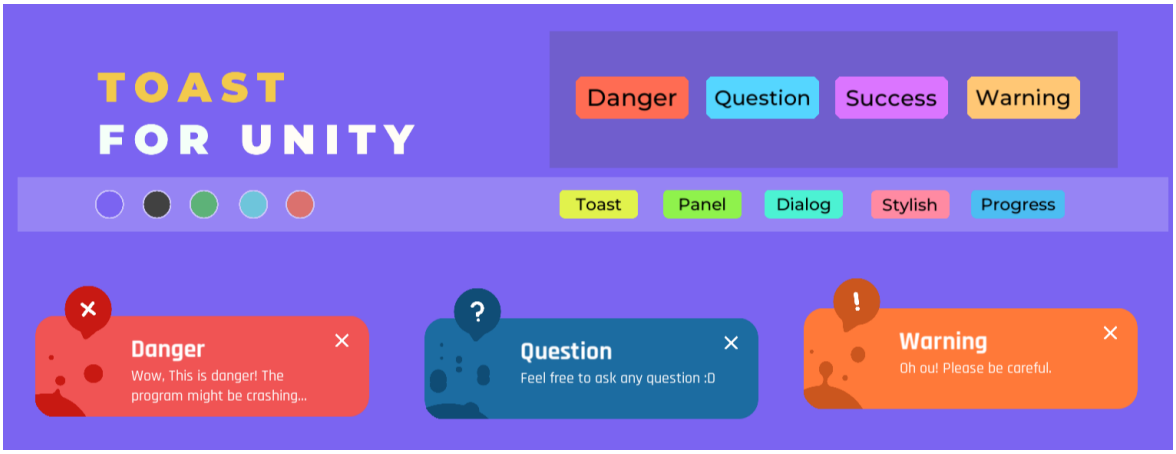
“ProgressToast” is the given Toast Name.

```
private void StylishPop(string stylishName, StylistToastModel toastModel)
{
    Toast.PopOut<StylistToastView>(stylishName, toastModel,
        ParentController.GetParent(ToastPosition.BottomCenter));
}
```

ProgressToastModel is a model class inherent by a empty **ToastModelBase** class**:**

```
public class ProgressToastModel : ToastModelBase
{
    public string Title;
    public bool ProgressRunOnStart;
    public bool DestroyWhenProgressComplete;
    public ProgressToastStatus Status;
    public UnityAction<float> ProgressValueChanged;
    public Action ProgressDone;
}
```

▼ 5. Stylish Panel



▼ API Usage Summary:

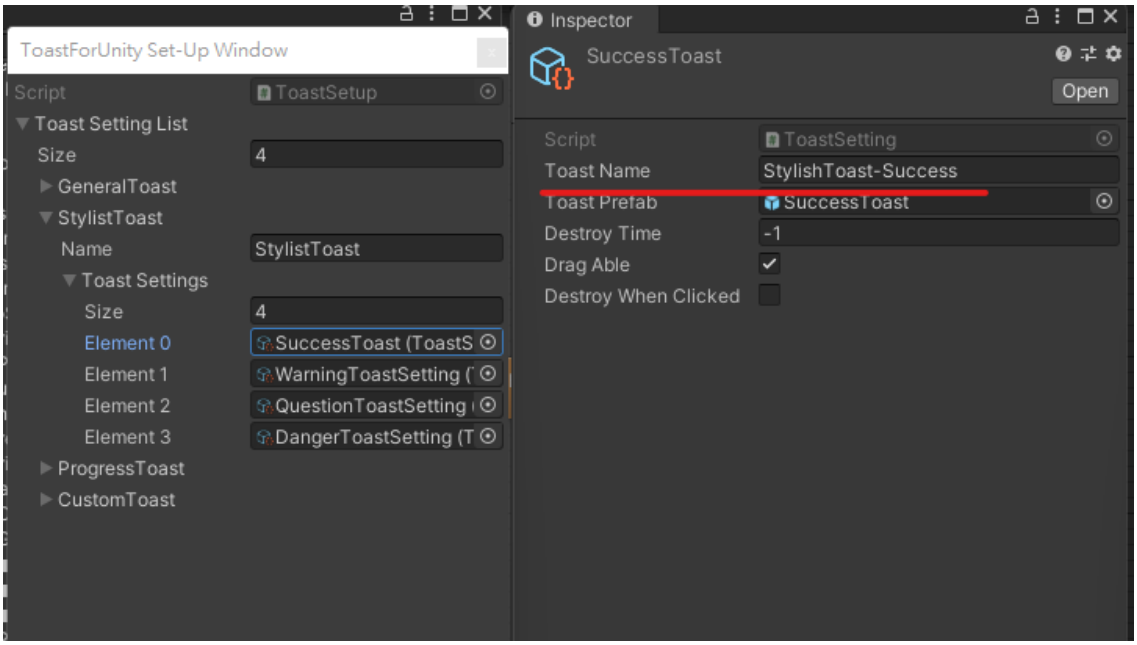
Basic API:

```
Toast.PopOut<StylistToastView>(string stylishName, StylistToastModel model);
```

Basic API With Parent Position:

```
Toast.PopOut<StylistToastView>(string stylishName, StylistToastModel model, Transform parentTransform);
```

stylishName is the **Toast Name at the ToastForUnity set-up window:**



StylistToastModel is a model class inherent by a empty **ToastModelBase** class**:**

```
public class StylistToastModel : ToastModelBase
{
    public string Title;
    public string Content;
}
```

▼ Example Code Snippets:

```
WarningBtn.onClick.AddListener(() =>
{
    StylishPop("StylishToast-Warning", new StylistToastModel()
    {
        Title = "Warning",
        Content = "Oh ou! Please be careful."
    });
});
```

```
SuccessBtn.onClick.AddListener(() =>
{
    StylishPop("StylishToast-Success", new StylistToastModel()
    {
        Title = "Success",
        Content = "Congratulation! You made it! You are just amazing!"
    });
});
```

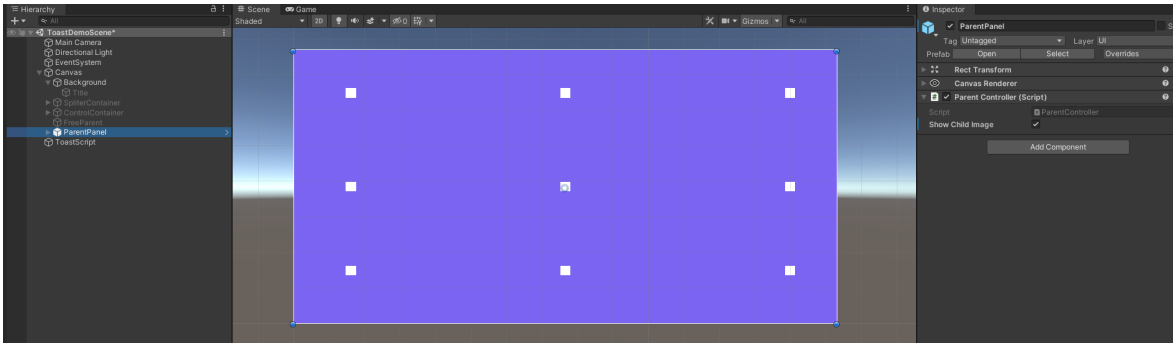
```
QuestionBtn.onClick.AddListener(() =>
{
    StylishPop("StylishToast-Question", new StylistToastModel()
    {
        Title = "Question",
        Content = "Feel free to ask any question :D"
    });
});
```

```
DangerBtn.onClick.AddListener(() =>
{
    StylishPop("StylishToast-Danger", new StylistToastModel()
    {
        Title = "Danger",
        Content = "Wow, This is danger! The program might be crashing..."
    });
});
```

▼ Parent Controller

We provide a **ParentPanel** Prefab as a position spawning helper, which can easily helps to spawn Toast at a different position.

To use it, drag the prefab to your scene as reference.



This prefab is provide 9 different point as a position reference.

Example Usage Code :

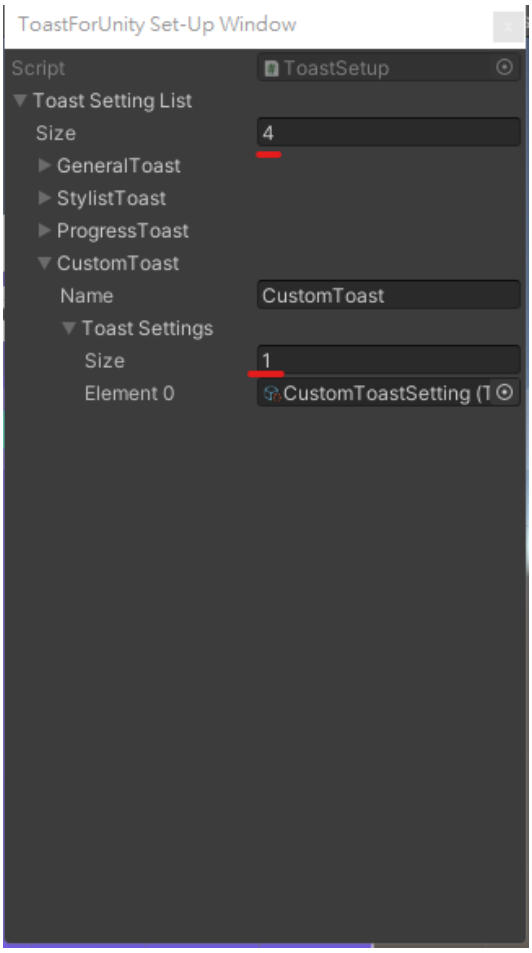
```
public class ToastPanel : MonoBehaviour
{
    public ParentController ParentPrefab; // This is the ParentPanel Reference.

    private void ToastPop()
    {
        Transform position = ParentPrefab.GetParent(ToastPosition.TopLeft);
        Toast.PopOut(ContentInput.text, ToastStatus.Success,position);
    }
}
```

ToastPosition is a enum of 9 position reference, and a random position.

```
public enum ToastPosition
{
  TopLeft,
  TopCenter,
  TopRight,
  MiddleLeft,
  MiddleCenter,
  MiddleRight,
  BottomLeft,
  BottomCenter,
  BottomRight,
  Random,
}
```

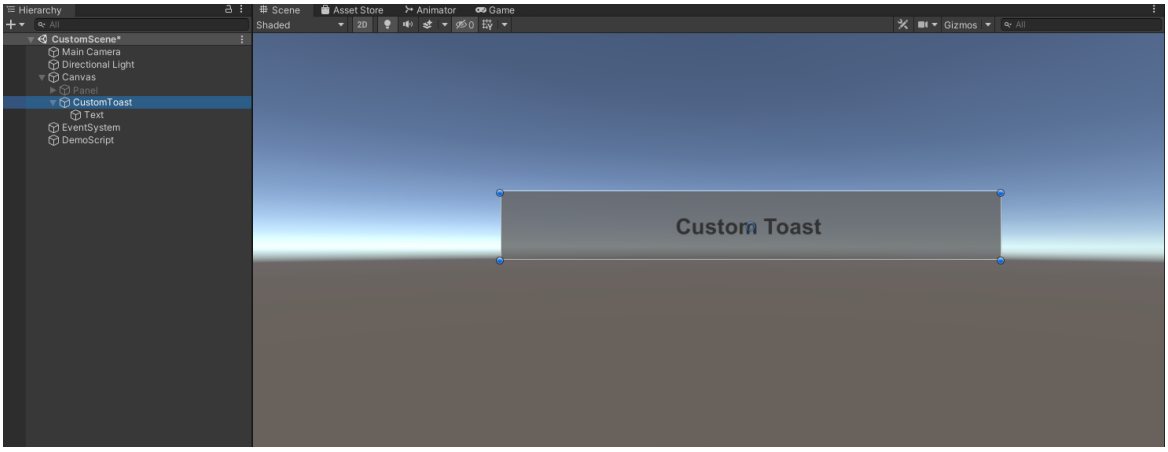
▼ Create a custom toast steps:

1. At the ToastForUnity Set-up Window, increase a new size of the first row if you want to create a new type of toast list, or you can use the pre-created CustomToast list to store your custom toast.



2. **Design Your Own Toast UI In Unity uGUI system.**

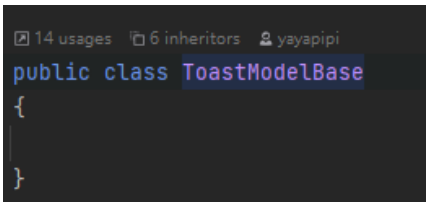
   Before adding a new toast to the set-up panel, you need to design your own toast style first



3. You need to add 2 new c# scripts for each custom toast.
    a. Create a new c# model script class inherent to the **ToastModelBase**

```
// This is to tell the toast which information need to receive.
public class CustomModel : ToastModelBase
{
   public string Content;

   // Add More Variable Depends On Your Usage
   // ex: public int Score;
}
```

Note: **ToastModelBase** is an empty class, inherent this class is use to make the toast core knowing this is a toast base model class.



    b. Create another new c# view script class inherent By **ToastPrefabBase**

```
// This is the script needs to drag into your toast prefab
public class CustomToastView: ToastPrefabBase
{
    // UI Variable Reference
    public Text ContentText;

    // This Function Will Execute When Toast Pop Out
    public override void Initialize(ToastModelBase toastModel)
    {
        //Convert The Model To Your Define CustomModel
        CustomModel customModel = toastModel as CustomModel;

        //Do Whatever You Want For Toast Initialization
        if (customModel != null)
            ContentText.text = customModel.Content;
    }
}
```

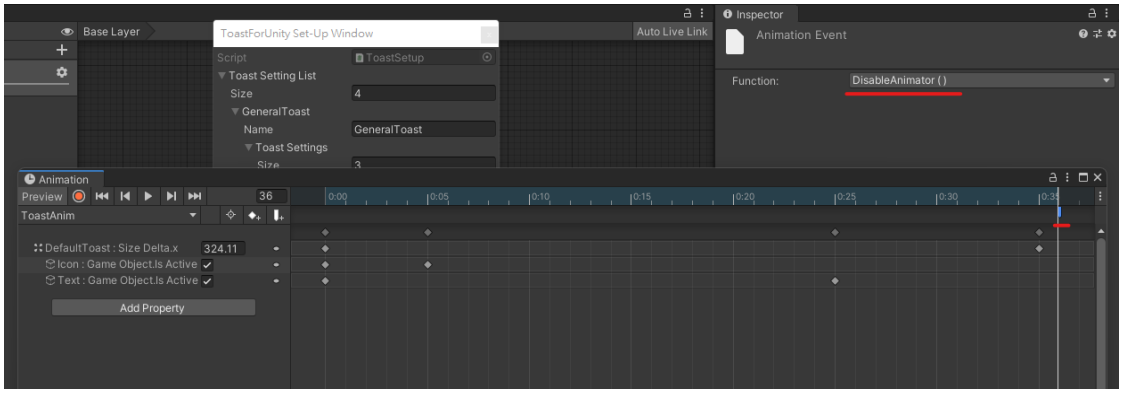
Note: **ToastPrefabBase** is an abstact class inherent by MonoBehaviour.

```
public abstract class ToastPrefabBase:MonoBehaviour
{
    public abstract void Initialize(ToastModelBase toastModel);

    public void DisableAnimator() {
        transform.gameObject.GetComponent<Animator>().enabled = false;
    }

}
```

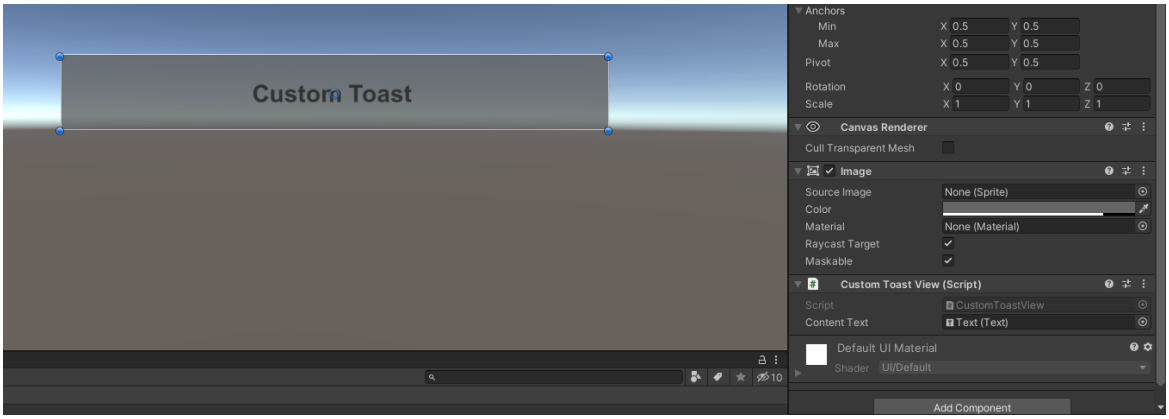
It contains 2 function.

Initialize - an abstact function you needs to implement when toast pop out.

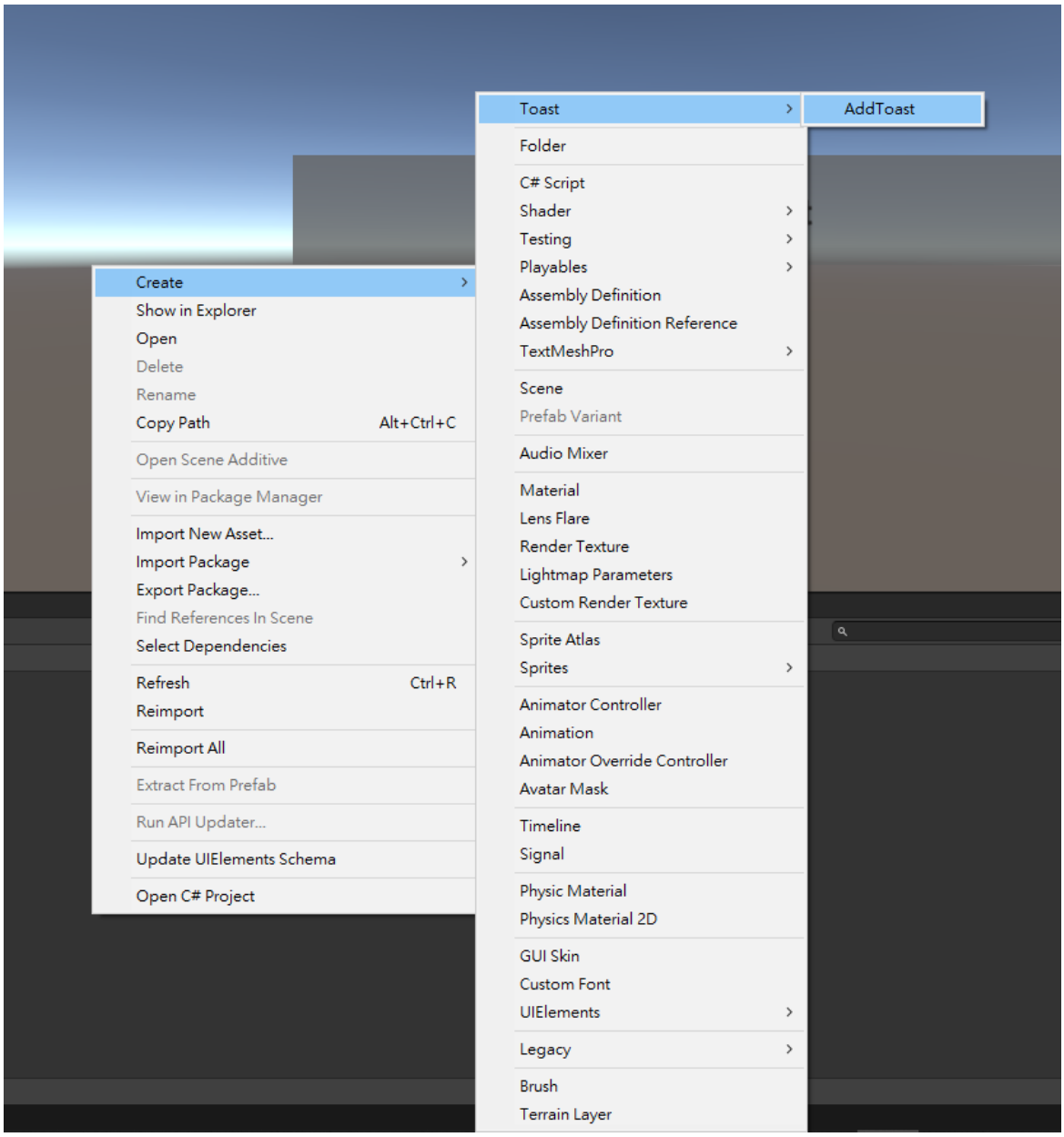
DisableAnimator - a public function that able to use at the animator, allows to animator stop playing when finished playing. You can also use it if you want.



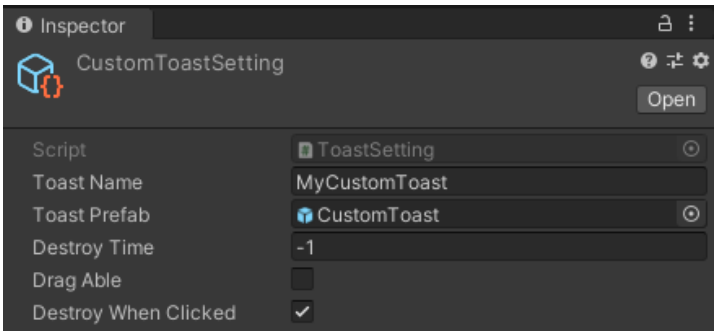
4. Remember To Drag the **CustomToastView.cs** To Your Toast Prefab, and assign your UI reference variable.



5. Add a **ToastSettings (Scriptable Object)** Inside Unity Editor

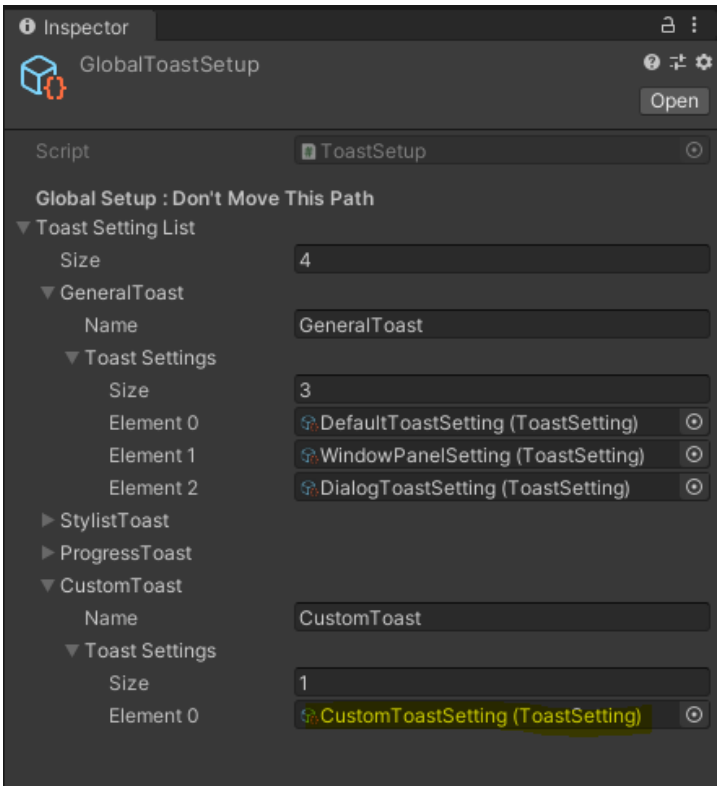


6. Setting Up Your ToastSettings Value




```
- Toast Name ⇒ The name of this toast (it should be an unique name)
- Toast Prefab ⇒ The prefab of this toast (drag in your prefab)
- Destroy Time ⇒ Destroy this gameObject after second (type -1 for not destroy)
- Drag Able ⇒ Trigger for this toast is draggable
- Destroy When Clicked ⇒ Destroy this toast when mouse clicked.
```

7. Now, add to the **Toast Set-up Window** Panel.



I add to CustomToast at Element 0 in my case, you may need to change the Size to 2, and assign your toast at Element 1.

8. You are done, now **Pop** Your Toast!

Example Usage Code:

```
public void CallCustomToast()
{
    Toast.PopOut<CustomToastView>("MyCustomToast", new CustomModel()
    {
        Content = "New Custom Toast"
    });

    // or

    Toast.PopOut<CustomToastView>("MyCustomToast", new CustomModel()
    {
        Content = "New Custom Toast"
    }, Parent.transform);
}
```