Events System: Scripting API

Assembly

```
CarterGames.Cart.Core.Runtime
```

Namespace

```
CarterGames.Core.Events
```

Definition

```
private readonly Evt MyEvent = new Evt();
```

As Evt is a class you'll need to create an instance of it for use, otherwise it'll return null and do nothing.

Methods

Add

Adds an listener to the evt instance. Note that it must have matching parameters to correctly subscribe.

```
public void Add(Action listener);
public void Add(Action<T> listener);
public void Add(Action<T1,T2> listener);
public void Add(Action<T1,T2,T3> listener);
public void Add(Action<T1,T2,T3,T4> listener);
public void Add(Action<T1,T2,T3,T4,T5> listener);
public void Add(Action<T1,T2,T3,T4,T5,T6> listener);
public void Add(Action<T1,T2,T3,T4,T5,T6,T7> listener);
public void Add(Action<T1,T2,T3,T4,T5,T6,T7,T8> listener);
```

```
Evt MyEvent = new Evt();

private void OnEnable()
{
    MyEvent.Add(MyMethod);
}
```

```
private void MyMethod()
{
    // My logic here...
}
```

AddAnonymous

Adds an anonymous listener to the evt instance. The parameters do not need to match when adding anonymous actions to run on an event. Though it is still advised to match them where possible.

```
public void AddAnonymous(string id, Action listener);
public void AddAnonymous(string id, Action<T> listener);
public void AddAnonymous(string id, Action<T1,T2> listener);
public void AddAnonymous(string id, Action<T1,T2,T3> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4,T5> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4,T5,T6> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4,T5,T6,T7> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4,T5,T6,T7,T8> listener);
```

```
Evt MyEvent = new Evt();

private void OnEnable()
{
    MyEvent.AddAnonymous("MyMethod", MyMisMatchedMethod(100));
}

private void MyMisMatchedMethod(int health)
{
    // My logic here...
}
```

Remove

Removes a listener to the evt instance. Note that it must have matching parameters to correctly unsubscribe.

```
public void Remove(Action listener);
public void Remove(Action<T> listener);
public void Remove(Action<T1,T2> listener);
public void Remove(Action<T1,T2,T3> listener);
public void Remove(Action<T1,T2,T3,T4> listener);
public void Remove(Action<T1,T2,T3,T4,T5> listener);
```

```
public void Remove(Action<T1,T2,T3,T4,T5,T6> listener);
public void Remove(Action<T1,T2,T3,T4,T5,T6,T7> listener);
public void Remove(Action<T1,T2,T3,T4,T5,T6,T7,T8> listener);
```

```
Evt MyEvent = new Evt();

private void OnEnable()
{
    MyEvent.Remove(MyMethod);
}

private void MyMethod()
{
    // My logic here...
}
```

RemoveAnonymous

Removes an anonymous listener to the evt instance. You only need to pass the key you assigned to the anonymous event listener to remove it.

```
public void RemoveAnonymous(string id);
```

```
Evt MyEvent = new Evt();

private void OnEnable()
{
    MyEvent.RemoveAnonymous("MyMethod");
}
```

Raise

Raises/Invokes the evt which will run all the listeners currently subscribed to it. If your event takes parameters you'll need to pass through their values here.

```
public void Raise();
public void Raise(T param);
public void Raise(T1 param1, T2 param2);
public void Raise(T1 param1, T2 param2, T3 param3);
public void Raise(T1 param1, T2 param2, T3 param3, T4 param4);
public void Raise(T1 param1, T2 param2, T3 param3, T4 param4, T5 param5);
public void Raise(T1 param1, T2 param2, T3 param3, T4 param4, T5 param5, T6 param6
```

```
public void Raise(T1 param1, T2 param2, T3 param3, T4 param4, T5 param5, T6 param6
public void Raise(T1 param1, T2 param2, T3 param3, T4 param4, T5 param5, T6 param6
```

```
Evt MyEvent = new Evt();

private void OnEnable()
{
    MyEvent.Raise();
}
```

Clear

Clears all the listeners from this evt.

```
public void Clear();
```

```
Evt MyEvent = new Evt();

private void OnEnable()
{
    MyEvent.Clear();
}
```