# UltimatePooling Namespace

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✨️ GenericPoolGroup</td>
<td>Represents a pool group that can accept either a game object or component prefab as its root prefab.</td>
</tr>
<tr>
<td>✨️ PoolBehaviour</td>
<td>Intermediate behaviour script that allows spawn and despawn events to be received. The events are broadcast to the object that is being re-used so the script can be at any level on the objects hierarchy.</td>
</tr>
<tr>
<td>✨️ PoolGroup</td>
<td>Represents a spawn pool for a specific type of prefab. Valid types are game objects and components.</td>
</tr>
<tr>
<td>✨️ PoolManager</td>
<td>The manager that is responsible for all pool groups and handles the creation and destruction of pools at runtime.</td>
</tr>
<tr>
<td>✨️ ResourcesPoolGroup</td>
<td>Represents a pool group that manages a prefab object located within the resources folder.</td>
</tr>
</tbody>
</table>
UltimatePool

The main class or interacting with the UltimatePooling API. All spawning and despawning methods are found in this class however you can use the individual spawn method on pools if required.

## Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPoolReceiver</td>
<td>Implement this interface when you want to receive spawned and despawned events sent to the pooled object. This interface will typically be implemented by a mono behaviour script that is attached to a pooled object which will then receive the appropriate event when it is spawned or despawned. Alternatively you can inherit from PoolBehaviour which provides default overridable behaviour for these events. (Modifies the objects enabled state to show or hide the object).</td>
</tr>
</tbody>
</table>

## Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogLevel</td>
<td>The amount of detail to include in logged messages.</td>
</tr>
<tr>
<td>PoolEventType</td>
<td>The method that is used to inform pooled objects about their current spawn state.</td>
</tr>
</tbody>
</table>
GenericPoolGroup Class

Represents a pool group that can accept either a game object or component prefab as its root prefab.

Inheritance Hierarchy

```
System
   Object
   Component
      Behaviour
         MonoBehaviour
            UltimatePoolingPoolGroup
            UltimatePoolingGenericPoolGroup
            UltimatePoolingResourcesPoolGroup
```

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```
public class GenericPoolGroup : PoolGroup
```

The GenericPoolGroup type exposes the following members.

Constructors

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚫ GenericPoolGroup</td>
<td>Initializes a new instance of the GenericPoolGroup class</td>
</tr>
</tbody>
</table>
```

Top
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onInstanceDespawned</td>
<td>Handle despawning of a pooled object. By default, this method disables the game object. (Overrides PoolGroup.onInstanceDespawned(Object, PoolEventType).)</td>
</tr>
<tr>
<td>onInstanceSpawned</td>
<td>Handle the spawning of a pooled object. By default, this method enabled the game object. (Overrides PoolGroup.onInstanceSpawned(Object, PoolEventType, Vector3, Quaternion).)</td>
</tr>
</tbody>
</table>

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefab</td>
<td>Access the component or game object prefab. (Overrides PoolGroupPrefab.)</td>
</tr>
</tbody>
</table>
GenericPoolGroup Constructor

Initializes a new instance of the `GenericPoolGroup` class

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

```csharp
public GenericPoolGroup()
```

### See Also

Reference
- `GenericPoolGroup Class`
- `UltimatePooling Namespace`
## GenericPoolGroup Methods

The **GenericPoolGroup** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onInstanceDespawned</td>
<td>Handle despawning of a pooled object. By default, this method disables the game object. (Overrides PoolGroup onInstanceDespawned(Object, PoolEventType).)</td>
</tr>
<tr>
<td>onInstanceSpawned</td>
<td>Handle the spawning of a pooled object. By default, this method enabled the object. (Overrides PoolGroup onInstanceSpawned(Object, PoolEventType, Vector3, Quaternion).)</td>
</tr>
</tbody>
</table>

---

**See Also**

Reference

- **GenericPoolGroup Class**
- **UltimatePooling Namespace**
GenericPoolGroup onInstanceDespawned Method

Handle despawning of a pooled object. By default, this method disables the game object.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```csharp
protected override void onInstanceDespawned(
    Object instance,
    PoolEventType type
)
```

Parameters

- `instance`  
  Type: `Object`  
  The instance to handle the despawning of

- `type`  
  Type: `UltimatePoolingPoolEventType`  
  The type of event used to inform the object of its spawned status

See Also

Reference

- GenericPoolGroup Class
- UltimatePooling Namespace
GenericPoolGroup::onInstanceSpawned Method

Handle the spawning of a pooled object. By default, this method enabled the game object.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

## Syntax

### C#  

```csharp
protected override void onInstanceSpawned(
    Object instance,
    PoolEventType type,
    Vector3 position,
    Quaternion rotation
)
```

### Parameters

- **instance**  
  Type: `Object`  
  The newly spawned instance to handle

- **type**  
  Type: `UltimatePoolingPoolEventType`  
  The type of event used to inform the object of its spawn status

- **position**  
  Type: `Vector3`  
  The position to spawn the object at

- **rotation**  
  Type: `Quaternion`  
  The rotation to spawn the object with
See Also

Reference
GenericPoolGroup Class
UltimatePooling Namespace
### GenericPoolGroup Properties

The `GenericPoolGroup` type exposes the following members.

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Prefab Access the component or game object prefab. (Overrides <code>PoolGroupPrefab</code>).</th>
</tr>
</thead>
</table>

### See Also

**Reference**
- `GenericPoolGroup Class`
- `UltimatePooling Namespace`
GenericPoolGroupPrefab Property

Access the component or game object prefab.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#  
```csharp
public override Object Prefab { get; set; }
```

JavaScript

Property Value

Type: Object

See Also

Reference

GenericPoolGroup Class
UltimatePooling Namespace
IPoolReceiver Interface

Implement this interface when you want to receive spawned and despawned events sent to the pooled object. This interface will typically be implemented by a mono behaviour script that is attached to a pooled object which will then receive the appropriate event when it is spawned or despawned. Alternatively you can inherit from PoolBehaviour which provides default overridable behaviour for these events. (Modifies the objects enabled state to show or hide the object).

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public interface IPoolReceiver</code></td>
<td></td>
</tr>
</tbody>
</table>

The IPoolReceiver type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌟</td>
<td>OnDespawned</td>
<td>Called when an object is about to be returned to the pool. Note that this method will not be called when the object is destroyed.</td>
</tr>
<tr>
<td>🌟</td>
<td>OnSpawned</td>
<td>Called when an object has been spawned from the pool. This event allows the state of the object to be reset so it can be treaded as a newly created object. Note that this method</td>
</tr>
</tbody>
</table>
will not be called when the object is first created.

See Also

Reference
UltimatePooling Namespace
IPoolReceiver Methods

The `IPoolReceiver` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnDespawned</td>
<td>Called when an object is about to be returned to the pool. Note that this method will not be called when the object is destroyed.</td>
</tr>
<tr>
<td>OnSpawned</td>
<td>Called when an object has been spawned from the pool. This event allows the state of the object to be reset so it can be treated as a newly created object. Note that this method will not be called when the object is first created.</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- `IPoolReceiver Interface`
- `UltimatePooling Namespace`
IPoolReceiverOnDespawned Method

Called when an object is about to be returned to the pool. Note that this method will not be called when the object is destroyed.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```csharp
void OnDespawned(
    PoolGroup pool
)
```

Parameters

pool
Type: UltimatePoolingPoolGroup

See Also

Reference
IPoolReceiver Interface
UltimatePooling Namespace
IPoolReceiverOnSpawned Method

Called when an object has been spawned from the pool. This event allows the state of the object to be reset so it can be treated as a newly created object. Note that this method will not be called when the object is first created.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```csharp
void OnSpawned(
    PoolGroup pool
)
```

Parameters

- **pool**
  Type: UltimatePoolingPoolGroup

See Also

Reference
- IPoolReceiver Interface
- UltimatePooling Namespace
LogLevel Enumeration

The amount of detail to include in logged messages.

**Namespace:** UltimatePooling

**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

```
public enum LogLevel
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
<td>Dont log anything to the console.</td>
</tr>
<tr>
<td>Error</td>
<td>1</td>
<td>Only log error messages to the console.</td>
</tr>
<tr>
<td>Warning</td>
<td>2</td>
<td>Log errors and warnings to the console.</td>
</tr>
<tr>
<td>Message</td>
<td>3</td>
<td>Log all message types to the console, including errors and warnings.</td>
</tr>
</tbody>
</table>

### See Also

Reference
UltimatePooling Namespace
PoolBehaviour Class

Intermediate behaviour script that allows spawn and despawn events to be received. The events are broadcast to the object that is being reused so the script can be at any level on the objects hierarchy.

Inheritance Hierarchy

System
  Object
    Component
      Behaviour
        MonoBehaviour
          UltimatePooling
            PoolBehaviour

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#       JavaScript

```
public class PoolBehaviour : MonoBehaviour,
  IPoolReceiver
```

The PoolBehaviour type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoolBehaviour</td>
<td>Initializes a new instance of the PoolBehaviour class</td>
</tr>
</tbody>
</table>

Top
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnDespawned</strong></td>
<td>Called by the managing pool to notify that this object is about to be returned to the pool. This method will not be called when the object is about to be destroyed.</td>
</tr>
<tr>
<td><strong>OnSpawned</strong></td>
<td>Called by the managing pool to notify that this object has just been recycled from the pool. This method will not be called when the object is created for the first time.</td>
</tr>
</tbody>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>** monoDespawnedEvent**</td>
<td>The name of the event that is called when an object is returned to the pool.</td>
</tr>
<tr>
<td>** monoSpawnedEvent**</td>
<td>The name of the event that is called when an object is spawned from the pool.</td>
</tr>
</tbody>
</table>

## See Also

Reference

UltimatePooling Namespace
PoolBehaviour Constructor

Initializes a new instance of the `PoolBehaviour` class

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

```csharp
public PoolBehaviour()
```

### See Also

Reference
- `PoolBehaviour Class`
- `UltimatePooling Namespace`
PoolBehaviour Fields

The `PoolBehaviour` type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>monoDespawnedEvent</code></td>
<td>The name of the event that is called when an object is returned to the pool.</td>
</tr>
<tr>
<td><code>monoSpawnedEvent</code></td>
<td>The name of the event that is called when an object is spawned from the pool.</td>
</tr>
</tbody>
</table>

## See Also

Reference

- `PoolBehaviour Class`
- `UltimatePooling Namespace`
PoolBehaviour::monoDespawnedEvent Field

The name of the event that is called when an object is returned to the pool.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

**C#**

```csharp
public static string monoDespawnedEvent
```

**JavaScript**

Field Value

Type: String

### See Also

Reference  
PoolBehaviour Class  
UltimatePooling Namespace
PoolBehaviour::monoSpawnedEvent Field

The name of the event that is called when an object is spawned from the pool.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

**Syntax**

C#  
```csharp
public static string monoSpawnedEvent
```

Field Value

Type: **String**

**See Also**

**Reference**

PoolBehaviour Class  
UltimatePooling Namespace
PoolBehaviour Methods

The PoolBehaviour type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnDespawned</td>
<td>Called by the managing pool to notify that this object is about to be returned to the pool. This method will not be called when the object is about to be destroyed.</td>
</tr>
<tr>
<td>OnSpawned</td>
<td>Called by the managing pool to notify that this object has just been recycled from the pool. This method will not be called when the object is created for the first time.</td>
</tr>
</tbody>
</table>

See Also

Reference
PoolBehaviour Class
UltimatePooling Namespace
PoolBehaviourOnDespawned Method

Called by the managing pool to notify that this object is about to be returned to the pool. This method will not be called when the object is about to be destroyed.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**

```csharp
public virtual void OnDespawned(PoolGroup pool)
```

**JavaScript**

```javascript

```

### Parameters

**pool**

Type: UltimatePoolingPoolGroup  


### Implements

IPoolReceiverOnDespawned(PoolGroup)

### See Also

Reference  
PoolBehaviour Class  
UltimatePooling Namespace
PoolBehaviourOnSpawned Method

Called by the managing pool to notify that this object has just been recycled from the pool. This method will not be called when the object is created for the first time.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

```csharp
public virtual void OnSpawned(
    PoolGroup pool
)
```

### Parameters

**pool**

Type: UltimatePoolingPoolGroup


### Implements

IPoolReceiverOnSpawned(PoolGroup)

### See Also

**Reference**

PoolBehaviour Class  
UltimatePooling Namespace
PoolEventType Enumeration

The method that is used to inform pooled objects about their current spawn state.

**Namespace:** UltimatePooling

**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

```csharp
public enum PoolEventType
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BroadcastMessage</td>
<td>0</td>
<td>Broadcast a message to the game object and all scripts with a matching listener method will be informed.</td>
</tr>
<tr>
<td>InterfaceCallback</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

Reference

UltimatePooling Namespace
PoolGroup Class

Represents a spawn pool for a specific type of prefab. Valid types are game objects and components.

Inheritance Hierarchy

System
  Object
  Component
    Behaviour
      MonoBehaviour
        UltimatePoolingPoolGroup
        UltimatePoolingGenericPoolGroup

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#  JavaScript

```csharp
public abstract class PoolGroup : MonoBehaviour
```

The PoolGroup type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌟 PoolGroup</td>
<td>Initializes a new instance of the PoolGroup class</td>
</tr>
</tbody>
</table>

Top
# Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>despawn(Object)</td>
<td>Indicates that the specified instance can be returned to the pool and re-used at a later time.</td>
</tr>
<tr>
<td>despawn(Object, Single)</td>
<td>Indicates that the specified instance can be returned to the pool and re-used at a later time.</td>
</tr>
<tr>
<td>despawnAll</td>
<td>Attempts to reclaim all instances spawned by this pool and return them to the pool. Any instances spawned by this pool will be forcefully returned without warning.</td>
</tr>
<tr>
<td>despawnAll(Single)</td>
<td>Attempts to reclaim all instances spawned by this pool after the specified time delay. Any instances spawned by this pool will be forcefully returned without warning.</td>
</tr>
<tr>
<td>destroy</td>
<td>Attempts to destroy a specific instance from the pool. Note that 'OnDespawn' will not be called on the instance. Instead you should handle any cleanup in 'OnDestroy'</td>
</tr>
<tr>
<td>destroyAll</td>
<td>Attempts to destroy all pooled objects effectively emptying the pool and resetting its state.</td>
</tr>
</tbody>
</table>
Note that 'OnDespawn' will not be called on the pooled objects. Instead you should handle any cleanup in 'OnDestroy'.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destroySelf</td>
<td>Attempts to destroy all pooled objects effectively emptying the pool as well as destroying the pool instance. This is the preferred way of destroying an object pool as it allows the spawned objects to remain in the scene if required as opposed to being destroyed along with the pool. Note that 'OnDespawn' will not be called on any of the pooled objects. Instead you should handle any cleanup in 'OnDestroy'.</td>
</tr>
<tr>
<td>didSpawn</td>
<td>Returns true if this spawn group created the instance specified. Useful for spawn validation to make sure multiple pools are not attempting to manage the same instance.</td>
</tr>
<tr>
<td>onInstanceDespawned</td>
<td>Should be implemented by the inheriting class. Called when the object is about to be returned to the pool.</td>
</tr>
<tr>
<td>onInstanceSpawned</td>
<td>Should be implemented by the inheriting class. Called when the object has been taken from the pool and will be re-used.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spawn</td>
<td>Spawn an instance from the pool.</td>
</tr>
<tr>
<td>spawn(Vector3, Quaternion)</td>
<td>Spawn an instance from the pool using the specified position and rotation.</td>
</tr>
<tr>
<td>Start</td>
<td>Called by Unity on the first frame.</td>
</tr>
<tr>
<td>ToString</td>
<td>Override the string value to return detailed state information about the pool. (Overrides <code>Object.ToString</code>.)</td>
</tr>
</tbody>
</table>

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventType</td>
<td>The method used to inform a spawned instance when it is added to or removed from the pool.</td>
</tr>
<tr>
<td>maxAmount</td>
<td>The max amount of instances that the pool can contain. If this amount is exceeded then the pool will need to destroy some objects.</td>
</tr>
<tr>
<td>parentInstances</td>
<td>When true, all spawned instances will be added as child objects to the managing pool group.</td>
</tr>
<tr>
<td>pooled</td>
<td>A collection of objects that are ready to be spawned.</td>
</tr>
<tr>
<td>prewarmAmount</td>
<td>The amount of instances to preload.</td>
</tr>
</tbody>
</table>
% prewarmPerFrame  The max amount of instances to preload per frame.
% prewarmPool  Should the pool preload a set number of objects at startup. This can avoid frame spikes cause by calls to 'Instantiate' but may increase loading time.
% tracked  A collection of objects that have been spawned by this pool.

See Also

Reference  UltimatePooling Namespace
PoolGroup Constructor

Initializes a new instance of the `PoolGroup` class

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

C#  
```
protected PoolGroup()
```

### See Also

Reference  
`PoolGroup Class`  
`UltimatePooling Namespace`
PoolGroup Fields

The **PoolGroup** type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventType</td>
<td>The method used to inform a spawned instance when it is added to or removed from the pool.</td>
</tr>
<tr>
<td>maxAmount</td>
<td>The max amount of instances that the pool can contain. If this amount is exceeded then the pool will need to destroy some objects.</td>
</tr>
<tr>
<td>parentInstances</td>
<td>When true, all spawned instances will be added as child objects to the managing pool group.</td>
</tr>
<tr>
<td>pooled</td>
<td>A collection of objects that are ready to be spawned.</td>
</tr>
<tr>
<td>prewarmAmount</td>
<td>The amount of instances to preload.</td>
</tr>
<tr>
<td>prewarmPerFrame</td>
<td>The max amount of instances to preload per frame.</td>
</tr>
<tr>
<td>prewarmPool</td>
<td>Should the pool preload a set number of objects at startup. This can avoid frame spikes cause by calls to 'Instantiate' but may increase loading time.</td>
</tr>
</tbody>
</table>
A collection of objects that have been spawned by this pool.

### See Also

**Reference**
- PoolGroup Class
- UltimatePooling Namespace
PoolGroupEventType Field

The method used to inform a spawned instance when it is added to or removed from the pool.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**
```
public PoolEventType eventType
```

**JavaScript**

### Field Value

Type: `PoolEventType`

### See Also

Reference
- *PoolGroup Class*
- *UltimatePooling Namespace*
PoolGroup maxAmount Field

The max amount of instances that the pool can contain. If this amount is exceeded then the pool will need to destroy some objects.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>public int maxAmount</strong></td>
</tr>
</tbody>
</table>

**Field Value**  
**Type:** Int32

### See Also

**Reference**  
PoolGroup Class  
UltimatePooling Namespace
PoolGroup.parentInstances Field

When true, all spawned instances will be added as child objects to the managing pool group.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool parentInstances</td>
<td></td>
</tr>
</tbody>
</table>

**Field Value**  
Type: Boolean

### See Also

Reference  
PoolGroup Class  
UltimatePooling Namespace
PoolGrouppooled Field

A collection of objects that are ready to be spawned.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>Stack&lt;Object&gt;</td>
</tr>
<tr>
<td>pooled</td>
<td></td>
</tr>
</tbody>
</table>

Field Value

**Type:** Stack<Object>

### See Also

Reference

PoolGroup Class  
UltimatePooling Namespace
PoolGroup::prewarmAmount Field

The amount of instances to preload.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public int prewarmAmount</code></td>
<td></td>
</tr>
</tbody>
</table>

**Field Value**  
**Type:** Int32

### See Also

**Reference**  
PoolGroup Class  
UltimatePooling Namespace
PoolGroup prewarmPerFrame Field

The max amount of instances to preload per frame.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public int prewarmPerFrame</code></td>
<td></td>
</tr>
</tbody>
</table>

### Field Value

Type: `Int32`

### See Also

**Reference**
- PoolGroup Class
- UltimatePooling Namespace
PoolGroup

prewarmPool Field

Should the pool preload a set number of objects at startup. This can avoid frame spikes cause by calls to 'Instantiate' but may increase loading time.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>public</strong> bool prewarmPool</td>
<td></td>
</tr>
</tbody>
</table>

### Field Value

**Type:** Boolean

### See Also

**Reference**
- PoolGroup Class
- UltimatePooling Namespace
PoolGroup tracked Field

A collection of objects that have been spawned by this pool.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected HashSet&lt;Object&gt; tracked</code></td>
<td></td>
</tr>
</tbody>
</table>

### Field Value

Type: `HashSet<Object>`

### See Also

Reference  
PoolGroup Class  
UltimatePooling Namespace
PoolGroup Methods

The **PoolGroup** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>despawn(Object)</td>
<td>Indicates that the specified instance can be returned to the pool and re-used at a later time.</td>
</tr>
<tr>
<td>despawn(Object, Single)</td>
<td>Indicates that the specified instance can be returned to the pool and re-used at a later time.</td>
</tr>
<tr>
<td>despawnAll</td>
<td>Attempts to reclaim all instances spawned by this pool and return them to the pool. Any instances spawned by this pool will be forcefully returned without warning.</td>
</tr>
<tr>
<td>despawnAll(Single)</td>
<td>Attempts to reclaim all instances spawned by this pool after the specified time delay. Any instances spawned by this pool will be forcefully returned without warning.</td>
</tr>
<tr>
<td>destroy</td>
<td>Attempts to destroy a specific instance from the pool. Note that 'OnDespawn' will not be called on the instance. Instead</td>
</tr>
</tbody>
</table>
you should handle any cleanup in 'OnDestroy'

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destroyAll</td>
<td>Attempts to destroy all pooled objects effectively emptying the pool and resetting its state. Note that 'OnDespawn' will not be called on the pooled objects. Instead you should handle any cleanup in 'OnDestroy'</td>
</tr>
<tr>
<td>destroySelf</td>
<td>Attempts to destroy all pooled objects effectively emptying the pool as well as destroying the pool instance. This is the preferred way of destroying an object pool as it allows the spawned objects to remain in the scene if required as opposed to being destroyed along with the pool. Note that 'OnDespawn' will not be called on any of the pooled objects. Instead you should handle any cleanup in 'OnDestroy'</td>
</tr>
<tr>
<td>didSpawn</td>
<td>Returns true if this spawn group created the instance specified. Useful for spawn validation to make sure multiple pools are not attempting to manage the same instance.</td>
</tr>
<tr>
<td>onInstanceDespawned</td>
<td>Should be implemented by the inheriting class. Called when the object is about to be returned to the pool.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>onInstanceSpawned</td>
<td>Should be implemented by the inheriting class. Called when the object has been taken from the pool and will be re-used.</td>
</tr>
<tr>
<td>spawn</td>
<td>Spawn an instance from the pool.</td>
</tr>
<tr>
<td>spawn(Vector3, Quaternion)</td>
<td>Spawn an instance from the pool using the specified position and rotation.</td>
</tr>
<tr>
<td>Start</td>
<td>Called by Unity on the first frame.</td>
</tr>
<tr>
<td>ToString</td>
<td>Override the string value to return detailed state information about the pool. (Overrides Object.ToString.)</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- PoolGroup Class
- UltimatePooling Namespace
PoolGroup::despawn Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>despawn(Object)</code></td>
<td>Indicates that the specified instance can be returned to the pool and reused at a later time.</td>
</tr>
<tr>
<td><code>despawn(Object, Single)</code></td>
<td>Indicates that the specified instance can be returned to the pool and reused at a later time.</td>
</tr>
</tbody>
</table>

See Also

Reference
- PoolGroup Class
- UltimatePooling Namespace
PoolGroup despawn Method (Object)

Indicates that the specified instance can be returned to the pool and re-used at a later time.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```
public void despawn(
    Object instance
)
```

Parameters

`instance`

Type: Object
The instance to despawn

See Also

Reference

PoolGroup Class
despawn Overload
UltimatePooling Namespace
PoolGroup despawn Method (Object, Single)

Indicates that the specified instance can be returned to the pool and re-used at a later time.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

```csharp
public void despawn(
    Object instance,
    float time
)
```

### Parameters

*instance*
- Type: **Object**  
  The instance to despawn

*time*
- Type: **SystemSingle**  
  The amount of time to wait before the object is despawned

### See Also

**Reference**
- PoolGroup Class
- despawn Overload
- UltimatePooling Namespace
PoolGroup despawnAll Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>despawnAll</td>
<td>Attempts to reclaim all instances spawned by this pool and return them to the pool. Any instances spawned by this pool will be forcefully returned without warning.</td>
</tr>
<tr>
<td>despawnAll(Single)</td>
<td>Attempts to reclaim all instances spawned by this pool after the specified time delay. Any instances spawned by this pool will be forcefully returned without warning.</td>
</tr>
</tbody>
</table>

See Also

Reference
PoolGroup Class
UltimatePooling Namespace
PoolGroup despawnAll Method

Attempts to reclaim all instances spawned by this pool and return them to the pool. Any instances spawned by this pool will be forcefully returned without warning.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

⚠️ *Syntax*

```csharp
public void despawnAll()
```

⚠️ *See Also*

Reference  
PoolGroup Class  
despawnAll Overload  
UltimatePooling Namespace
PoolGroup.despawnAll Method (Single)

Attempts to reclaim all instances spawned by this pool after the specified time delay. Any instances spawned by this pool will be forcefully returned without warning.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**

```csharp
public void despawnAll(
    float time
)
```

**JavaScript**

```javascript
// Not supported in this context
```

### Parameters

- **time**
  - Type: System.Single
  - The amount of time to wait before despawning all instances

### See Also

- Reference
  - PoolGroup Class
  - despawnAll Overload
  - UltimatePooling Namespace
PoolGroup destroy Method

Attempts to destroy a specific instance from the pool. Note that 'OnDespawn' will not be called on the instance. Instead you should handle any cleanup in 'OnDestroy'.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

C#  
```csharp
public void destroy(
    Object instance,
    bool keepSpawnedInstances = true
)
```

### Parameters

- **instance**  
  Type: `Object`  
  The instance to remove from the pool

- **keepSpawnedInstances (Optional)**
  Type: `System.Boolean`  
  If true, the pool will also try to locate this instance in its spawned list

### See Also

- Reference  
  - PoolGroup Class  
  - UltimatePooling Namespace
PoolGroup destroyAll Method

Attempts to destroy all pooled objects effectively emptying the pool and resetting its state. Note that 'OnDespawn' will not be called on the pooled objects. Instead you should handle any cleanup in 'OnDestroy'

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**

```csharp
public void destroyAll(
    bool keepSpawnedInstances = true
)
```

**JavaScript**

```javascript
// Not applicable
```

### Parameters

- **keepSpawnedInstances** *(Optional)*  
  Type: System.Boolean  
  If true, all spawned objects created by this pool will also be destroyed

### See Also

**Reference**  
- PoolGroup Class  
- UltimatePooling Namespace
PoolGroupdestroySelf Method

Attempts to destroy all pooled objects effectively emptying the pool as well as destroying the pool instance. This is the preferred way of destroying an object pool as it allows the spawned objects to remain in the scene if required as opposed to being destroyed along with the pool. Note that 'OnDespawn' will not be called on any of the pooled objects. Instead you should handle any cleanup in 'OnDestroy'

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```csharp
public void destroySelf(
    bool keepSpawnedInstances = true
)
```

Parameters

*keepSpawnedInstances (Optional)*

Type: SystemBoolean

When true, the pool will avoid destroying its parent object if the objects spawned by this pool are parented to it. This allows them to remain in the scene even though the pool will be destroyed

See Also

Reference

PoolGroup Class
UltimatePooling Namespace
PoolGroup didSpawn Method

Returns true if this spawn group created the instance specified. Useful for spawn validation to make sure multiple pools are not attempting to manage the same instance.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

```
public bool didSpawn(
    Object instance
)
```

### Parameters

*instance*  
Type: **Object**  
The instance to check

### Return Value

Type: **Boolean**  
True if this pool spawned the specified instance otherwise false

### See Also

Reference  
PoolGroup Class  
UltimatePooling Namespace
PoolGroupOnInstanceDespawned

Method

Should be implemented by the inheriting class. Called when the object is about to be returned to the pool.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

**C#**  
```csharp
protected abstract void onInstanceDespawned(
    Object instance,
    PoolEventType type
)
```

### Parameters

**instance**  
Type: `Object`  
The object that is about to be pooled  

**type**  
Type: `UltimatePooling.PoolEventType`  
The event type that should be used to inform the object that it is about to be despawned

### See Also

**Reference**  
PoolGroup Class  
UltimatePooling Namespace
PoolGroupOnInstanceSpawned Method

Should be implemented by the inheriting class. Called when the object has been taken from the pool and will be re-used.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

## Syntax

```csharp
protected abstract void onInstanceSpawned(
    Object instance,
    PoolEventType type,
    Vector3 position,
    Quaternion rotation
)
```

### Parameters

**instance**  
Type: **Object**  
The object that has been re-used

**type**  
Type: **UltimatePoolingPoolEventType**  
The event type that should be used to inform the object that it has been spawned

**position**  
Type: **Vector3**  
The position to spawn the object at

**rotation**  
Type: **Quaternion**
The rotation to spawn the object with

See Also

Reference
PoolGroup Class
UltimatePooling Namespace
PoolGroupspawn Method

▲ Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spawn</td>
<td>Spawn an instance from the pool.</td>
</tr>
<tr>
<td>spawn(Vector3, Quaternion)</td>
<td>Spawn an instance from the pool using the specified position and rotation.</td>
</tr>
</tbody>
</table>

See Also

Reference

PoolGroup Class
UltimatePooling Namespace
PoolGroupspawn Method

Spawn an instance from the pool.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

## Syntax

### C#  
```csharp
public Object spawn()
```

### JavaScript

**Return Value**

Type: **Object**  
An instance of a pooled object

## See Also

**Reference**  
PoolGroup Class  
spawn Overload  
UltimatePooling Namespace
PoolGroupspawn Method
(Vector3, Quaternion)

Spawn an instance from the pool using the specified position and rotation.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

```csharp
public Object spawn(
    Vector3 position,
    Quaternion rotation
)
```

### Parameters

**position**  
Type: **Vector3**  
The position in 3D space to place the spawned object

**rotation**  
Type: **Quaternion**  
The initial rotation to give the spawned object

### Return Value

Type: **Object**  
An instance of a pooled object

### See Also

Reference
PoolGroup Class
spawn Overload
UltimatePooling Namespace
PoolGroupStart Method

Called by Unity on the first frame.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

C#  
```csharp
protected virtual void Start()
```

### See Also

Reference  
PoolGroup Class  
UltimatePooling Namespace
PoolGroupToString Method

Override the string value to return detailed state information about the pool.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**

```csharp
public override string ToString()
```

**JavaScript**

```javascript

```

### Return Value

**Type:** String  
A string representation of the current pool state

### See Also

**Reference**

PoolGroup Class  
UltimatePooling Namespace
PoolGroup Properties

The `PoolGroup` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsFull</td>
<td>Returns true if the pool is unable to store anymore pooled instances.</td>
</tr>
<tr>
<td>IsPrewarming</td>
<td>Returns true if the pool is currently prewarming.</td>
</tr>
<tr>
<td>Prefab</td>
<td>Should be implemented by the inheriting class. Should return the specific prefab type, For example 'GameObject'.</td>
</tr>
</tbody>
</table>

See Also

Reference

- PoolGroup Class
- UltimatePooling Namespace
PoolGroupIsFull Property

Returns true if the pool is unable to store anymore pooled instances.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

```csharp
public bool IsFull { get; }
```

**Property Value**

Type: `Boolean`

### See Also

Reference

- PoolGroup Class
- UltimatePooling Namespace
PoolGroup.IsPrewarming Property

Returns true if the pool is currently prewarming.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#  JavaScript

```csharp
public bool IsPrewarming { get; }
```

Property Value
Type: Boolean

See Also

Reference
PoolGroup Class
UltimatePooling Namespace
PoolGroupPrefab Property

Should be implemented by the inheriting class. Should return the specific prefab type, for example 'GameObject'.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

**Syntax**

```csharp
public abstract Object Prefab { get; set; }
```

Property Value  
Type: **Object**

**See Also**

Reference  
PoolGroup Class  
UltimatePooling Namespace
PoolManager Class

The manager that is responsible for all pool groups and handles the creation and destruction of pools at runtime.

Inheritance Hierarchy

System Object UltimatePooling PoolManager

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#      JavaScript

```
public sealed class PoolManager
```

The PoolManager type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![createPool(String)](</td>
<td>Attempts to create a new object pool for a prefab located in the resources folder. If a pool already exists for the specified prefab name then this method will simply return the existing pool.</td>
</tr>
<tr>
<td>![createPool(Component, String)](</td>
<td>Attempts to create a new object pool for the component prefab type. If a</td>
</tr>
</tbody>
</table>
pool already exists for the specified prefab then this method will simply return the existing pool.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>createPool(GameObject, String)</code></td>
<td>Attempt to create a new object pool for prefab type. If a pool already exists for the specified prefab then this method will simply return the existing pool.</td>
</tr>
<tr>
<td><code>destroyPool</code></td>
<td>Destroys a pool group and call of its pooled instances.</td>
</tr>
<tr>
<td><code>findPool(String)</code></td>
<td>Find the pool for the prefab with the specified name.</td>
</tr>
<tr>
<td><code>findPool(Object)</code></td>
<td>Find the pool for the specified prefab.</td>
</tr>
<tr>
<td><code>findPoolWithInstance</code></td>
<td>Find the pool that initially spawned the specified instance. This method will fail if the pool that spawned this instance has been destroyed. In this case it will be up to the user to destroy the object manually, or call <code>despawn(Object)</code> which will result in the same thing.</td>
</tr>
<tr>
<td><code>hasPool</code></td>
<td>Returns true if there is an existing pool for the specified prefab type.</td>
</tr>
</tbody>
</table>

See Also
Reference
UltimatePooling Namespace
PoolManager Methods

The PoolManager type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>createPool(String)</code></td>
<td>Attempts to create a new object pool for a prefab located in the resources folder. If a pool already exists for the specified prefab name then this method will simply return the existing pool.</td>
</tr>
<tr>
<td><code>createPool(Component, String)</code></td>
<td>Attempts to create a new object pool for the component prefab type. If a pool already exists for the specified prefab then this method will simply return the existing pool.</td>
</tr>
<tr>
<td><code>createPool(GameObject, String)</code></td>
<td>Attempt to create a new object pool for prefab type. If a pool already exists for the specified prefab then this method will simply return the existing pool.</td>
</tr>
<tr>
<td><code>destroyPool</code></td>
<td>Destroys a pool group and call of its pooled instances.</td>
</tr>
<tr>
<td><code>findPool(String)</code></td>
<td>Find the pool for the prefab with the specified name.</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>findPool(Object)</code></td>
<td>Find the pool for the specified prefab.</td>
</tr>
<tr>
<td><code>findPoolWithInstance</code></td>
<td>Find the pool that initially spawned the specified instance. This method will fail if the pool that spawned this instance has been destroyed. In this case it will be up to the user to destroy the object manually, or call <code>despawn(Object)</code> which will result in the same thing.</td>
</tr>
<tr>
<td><code>hasPool</code></td>
<td>Returns true if there is an existing pool for the specified prefab type.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

- PoolManager Class
- UltimatePooling Namespace
## PoolManager `createPool` Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>createPool(String)</code></td>
<td>Attempts to create a new object pool for a prefab located in the resources folder. If a pool already exists for the specified prefab name then this method will simply return the existing pool.</td>
</tr>
<tr>
<td><code>createPool(Component, String)</code></td>
<td>Attempts to create a new object pool for the component prefab type. If a pool already exists for the specified prefab then this method will simply return the existing pool.</td>
</tr>
<tr>
<td><code>createPool(GameObject, String)</code></td>
<td>Attempt to create a new object pool for prefab type. If a pool already exists for the specified prefab then this method will simply return the existing pool.</td>
</tr>
</tbody>
</table>

---

**See Also**

Reference

PoolManager Class
UltimatePooling Namespace
PoolManager.createPool Method (String)

Attempts to create a new object pool for a prefab located in the resources folder. If a pool already exists for the specified prefab name then this method will simply return the existing pool.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
Version: 0.0.0.0

**Syntax**

```csharp
public PoolGroup createPool(
    string prefabName
)
```

**Parameters**

*prefabName*

Type: **System.String**  
The name of the prefab in the resources folder

**Return Value**

Type: **PoolGroup**  
An instance of a pool group

**See Also**

Reference

*PoolManager Class*
*createPool Overload*
*UltimatePooling Namespace*
PoolManager createPool Method (Component, String)

Attempts to create a new object pool for the component prefab type. If a pool already exists for the specified prefab then this method will simply return the existing pool.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```csharp
public PoolGroup createPool(
    Component prefab,
    string name = ""
)
```

Parameters

- **prefab**
  Type: `Component`
  The prefab to create the pool for

- **name** *(Optional)*
  Type: `System.String`
  The name of the pool

Return Value

Type: `PoolGroup`
An instance of a pool group

See Also
Reference

PoolManager Class
createPool Overload
UltimatePooling Namespace
PoolManager createPool Method
(GameObject, String)

Attempt to create a new object pool for prefab type. If a pool already exists for the specified prefab then this method will simply return the existing pool.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```
public PoolGroup createPool(
    GameObject prefab,
    string name = ""
)
```

Parameters

prefab
Type: GameObject
The prefab to create the pool for

name (Optional)
Type: System.String
The name of the pool

Return Value
Type: PoolGroup
An instance of a pool group

See Also
Reference

PoolManager Class
createPool Overload
UltimatePooling Namespace
PoolManager destroyPool Method

Destroys a pool group and call of its pooled instances.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

**C#**
```
public void destroyPool(
    PoolGroup pool,
    bool keepSpawnedInstances = true
)
```

**JavaScript**

### Parameters

- **pool**  
  Type: `UltimatePooling.PoolGroup`  
  The pool to destroy

- **keepSpawnedInstances** *(Optional)*  
  Type: `System.Boolean`  
  When true, all spawned instances will be kept alive

### See Also

- **Reference**  
  PoolManager Class  
  UltimatePooling Namespace
## PoolManager findPool Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>findPool(String)</td>
<td>Find the pool for the prefab with the specified name.</td>
</tr>
<tr>
<td>findPool(Object)</td>
<td>Find the pool for the specified prefab.</td>
</tr>
</tbody>
</table>

### See Also

Reference
- **PoolManager Class**
- **UltimatePooling Namespace**
PoolManager findPool Method (String)

Find the pool for the prefab with the specified name.

**Namespace:** UltimatePooling
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**

```csharp
public PoolGroup findPool(
    string name
)
```

**JavaScript**

### Parameters

*name*

Type: `System.String`

The name of the prefab to find the pool for

### Return Value

Type: `PoolGroup`

An instance of the pool group responsible for the prefab with the specified name

### See Also

Reference

*PoolManager Class*

*findPool Overload*

*UltimatePooling Namespace*
PoolManager findPool Method (Object)

Find the pool for the specified prefab.

**Namespace:** UltimatePooling
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**
```
public PoolGroup findPool(
    Object prefab
)
```

**JavaScript**

### Parameters

**prefab**
- Type: Object
- The prefab to find the pool for

### Return Value

- Type: PoolGroup
- An instance of the pool group responsible for the specified prefab

### See Also

**Reference**
- PoolManager Class
- findPool Overload
- UltimatePooling Namespace
PoolManager findPoolWithInstance Method

Find the pool that initially spawned the specified instance. This method will fail if the pool that spawned this instance has been destroyed. In this case it will be up to the user to destroy the object manually, or call `despawn(Object)` which will result in the same thing.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

**Syntax**

```csharp
public PoolGroup findPoolWithInstance(
    Object instance
)
```

**Parameters**

`instance`  
Type: `Object`  
The instance to find the managing pool for

**Return Value**

Type: `PoolGroup`  
The managing pool group if found

**See Also**

**Reference**  
PoolManager Class  
UltimatePooling Namespace
PoolManager hasPool Method

Returns true if there is an existing pool for the specified prefab type.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#    

```csharp
public bool hasPool(
    Object prefab
)
```

Parameters

prefab
Type: Object
The prefab to check for

Return Value
Type: Boolean
True if the specified prefab is already associated with a pool group

See Also

Reference
PoolManager Class
UltimatePooling Namespace
ResourcesPoolGroup Class

Represents a pool group that manages a prefab object located within the resources folder.

Inheritance Hierarchy

System
  Object
  Component
    Behaviour
      MonoBehaviour
        UltimatePoolingPoolGroup
        UltimatePoolingGenericPoolGroup
        UltimatePoolingResourcesPoolGroup

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#  JavaScript

```csharp
public class ResourcesPoolGroup : GenericPoolGroup
```

The ResourcesPoolGroup type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourcesPoolGroup</td>
<td>Initializes a new instance of the ResourcesPoolGroup class</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Called by Unity when the pool is created. (Overrides PoolGroupStart.)</td>
</tr>
</tbody>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefabName</td>
<td>The name of the prefab to load from the resources folder.</td>
</tr>
</tbody>
</table>

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefab</td>
<td>We need to modify the way that the prefab is retrieved. (Overrides GenericPoolGroupPrefab.)</td>
</tr>
</tbody>
</table>

## See Also

Reference
UltimatePooling Namespace
ResourcesPoolGroup Constructor

Initializes a new instance of the ResourcesPoolGroup class

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>Language</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>C#</td>
<td><code>public ResourcesPoolGroup()</code></td>
</tr>
</tbody>
</table>

### See Also

Reference

- ResourcesPoolGroup Class
- UltimatePooling Namespace
ResourcesPoolGroup Fields

The ResourcesPoolGroup type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefabName</td>
<td>The name of the prefab to load from the resources folder.</td>
</tr>
</tbody>
</table>

See Also

Reference

ResourcesPoolGroup Class
UltimatePooling Namespace
ResourcesPoolGroup prefabName Field

The name of the prefab to load from the resources folder.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**

```csharp
public string prefabName
```

**JavaScript**

Field Value

Type: *String*

### See Also

**Reference**

- ResourcesPoolGroup Class
- UltimatePooling Namespace
ResourcesPoolGroup Methods

The ResourcesPoolGroup type exposes the following members.

- **Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🍁 Start</td>
<td>Called by Unity when the pool is created. (Overrides PoolGroupStart.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- ResourcesPoolGroup Class
- UltimatePooling Namespace
ResourcesPoolGroupStart Method

Called by Unity when the pool is created.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

**C#**

```csharp
protected override void Start()
```

**JavaScript**

```javascript
// Not applicable
```

### See Also

Reference  
**ResourcesPoolGroup Class**  
**UltimatePooling Namespace**
ResourcesPoolGroup Properties

The ResourcesPoolGroup type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefab</td>
<td>We need to modify the way that the prefab is retrieved. (Overrides GenericPoolGroupPrefab.)</td>
</tr>
</tbody>
</table>

See Also

Reference
- ResourcesPoolGroup Class
- UltimatePooling Namespace
ResourcesPoolGroupPrefab Property

We need to modify the way that the prefab is retrieved.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

**C#**

```csharp
public override Object Prefab { get; set; }
```

**JavaScript**

Copy

**Property Value**

Type: **Object**

### See Also

**Reference**

ResourcesPoolGroup Class  
UltimatePooling Namespace
Ultimate Pool Class

The main class or interacting with the UltimatePooling API. All spawning and despawning methods are found in this class however you can use the individual spawn method on pools if required.

Inheritance Hierarchy

SystemObject  UltimatePoolingUltimatePool

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#      public static class UltimatePool

The UltimatePool type exposes the following members.

Methods

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌺</td>
<td>batchDespawn(IEnumerableComponent)</td>
<td>Attempts to Despawn all objects and return them to their pool group. Important: All objects in the enumerable collection must have been spawned from the same pool group. If you attempt...</td>
</tr>
</tbody>
</table>
to return instances from multiple pool using this method then you will invalidate all associated pool groups.

```csharp
batchDespawn(IEnumerable<GameObject>)
```
Attempts to Despawn all objects and return them to their pool group. Important: All objects in the enumerable collection must have been spawned from the same pool group. If you attempt to return instances from multiple pools using this method then you will invalidate all associated pool groups.

```csharp
batchSpawn(String, Int32)
```
Spawn a number of instances of a prefab with the specified name from the appropriate pool. This method will only succeed if the pool has been created before hand. Batch spawning is quicker than multiple calls to `spawn(String)` because the pool
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>batchSpawn(\text{Component}, \text{Int32})</td>
<td>Spawn a number instances of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls \text{spawn(Component)} because the pool cached on the first spawn.</td>
</tr>
<tr>
<td>batchSpawn(\text{GameObject}, \text{Int32})</td>
<td>Spawn a number instances of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls \text{spawn(GameObject)} because the pool cached on the first spawn.</td>
</tr>
<tr>
<td>batchSpawn(\text{String}, \text{Object}, \text{Int32})</td>
<td>Spawn a number instances of a prefab with the specified name from the</td>
</tr>
</tbody>
</table>
appropriate pool. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. This method will succeed if the pool has been created before hand. Batch spawning is quicker than multiple calls to spawn(String) because the pool is cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array, otherwise an out of bounds exception may occur.

\[\text{batchSpawn(Component, Object, Int32)}\]

Spawn a number of instances of the specified prefab and place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically
batchSpawn(GameObject, GameObject, Int32)

Spawn a number of instances of the specified prefab and place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls spawn(GameObject) because the pool is cached on the first spawn. If amount specified then the value should be less than or equal to the length of the array, otherwise an out of bounds exception may occur.
value should be less than or equal to the length of the array, otherwise an out of bounds exception may occur.

<table>
<thead>
<tr>
<th>s</th>
<th>batchSpawnT(Component, Int32)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spawn a number of instances of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to spawnT(Component) because the pool is cached on the first spawn.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>s</th>
<th>batchSpawnT(Component, T, Int32)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spawn a number of instances of the specified prefab and place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch</td>
</tr>
</tbody>
</table>
spawning is quicker that multiple calls to `spawnT(Component)` because the pool is cached on the first spawn. If `amount` is specified then the value should be less than or equal to the length of the array, otherwise an out of bounds exception may occur.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>despawn(Object)</td>
<td>Direct replacement for <code>Object.Destroy</code> for pooling. Allows the specified instance to be returned to the pool and re-used at a later time.</td>
</tr>
<tr>
<td>despawn(Object, Single)</td>
<td>Direct overload replacement for <code>Object.Destroy</code> for pooling. Allows the specified instance to be returned to the pool and re-used at a later time.</td>
</tr>
<tr>
<td>despawnAll(GameObject)</td>
<td>Calls all spawned instances of the specified prefab back to their pool. This method allows you to pass a prefab such as a 'Bullet'</td>
</tr>
</tbody>
</table>
which will subsequently cause all 'Bullet' instances to be despawned.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>despawnAll(GameObject, Single)</td>
<td>Calls all spawned instances of the specified prefab back to their pool. This method allows you to pass a prefab such as a 'Bullet' which will subsequently cause all 'Bullet' instances to be despawned.</td>
</tr>
<tr>
<td>spawn(String)</td>
<td>Spawn an instance of a prefab with the specified name. The prefab name is the same name used when creating the pool, or if no name is used then the name of the prefab supplied is substituted. This method will only succeed if the pool has been created before hand.</td>
</tr>
<tr>
<td>spawn(Component)</td>
<td>Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is...</td>
</tr>
<tr>
<td>Spawn Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>spawn(GameObject)</td>
<td>Spawn an instance of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</td>
</tr>
<tr>
<td>spawn(String, Vector3, Quaternion)</td>
<td>Spawn an instance of a prefab with the specified name. This method will only succeed if the pool has been created beforehand.</td>
</tr>
<tr>
<td>spawn(Component, Vector3, Quaternion)</td>
<td>Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the component prefab type.</td>
</tr>
<tr>
<td>spawn(GameObject, Vector3, Quaternion)</td>
<td>Spawn an instance of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</td>
</tr>
</tbody>
</table>
spawnT(Component)  Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.

spawnT(Component, Vector3, Quaternion)  Spawn an instance of the specified component prefab from the appropriate pool. If no pools exist, then one is automatically created for the component prefab type.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logLevel</td>
<td>The amount level of logging that is allowed. Default is 'Message' - Full logging.</td>
</tr>
</tbody>
</table>

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pools</td>
<td>Access the pool manager which maintains all</td>
</tr>
</tbody>
</table>
existing pools. Allows pools to be created and destroyed.

See Also

Reference

UltimatePooling Namespace
Ultimate Pool Fields

The UltimatePool type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logLevel</td>
<td>The amount level of logging that is allowed. Default is 'Message' - Full logging.</td>
</tr>
</tbody>
</table>

See Also

Reference
UltimatePool Class
UltimatePooling Namespace
Ultimate Pool logLevel Field

The amount level of logging that is allowed. Default is 'Message' - Full logging.

**Namespace:** UltimatePooling
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

```csharp
public static LogLevel logLevel
```

### Field Value

Type: LogLevel

### See Also

Reference
UltimatePool Class
UltimatePooling Namespace
## UltimatePool Methods

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![s] batchDespawn(IEnumerableComponent)</td>
<td>Attempts to Despawn all objects and return them to their pool group. Important: All objects in the enumerable collection must have been spawned from the same pool group. If you attempt to return instances from multiple pools using this method then you will invalidate all associated pool groups.</td>
</tr>
<tr>
<td>![s] batchDespawn(IEnumerableGameObject)</td>
<td>Attempts to Despawn all objects and return them to their pool group. Important: All objects in the enumerable collection must have been spawned from the same pool group. If you attempt to return instances from multiple pools using this method then you will invalidate all associated pool groups.</td>
</tr>
</tbody>
</table>
from multiple pool using this method then you will invalidate all associated pool groups.

<table>
<thead>
<tr>
<th>batchSpawn(String, Int32)</th>
<th>Spawn a number instance of a prefab with the specified name from the appropriate pool. This method will succeed if the pool has been created before hand. Batch spawning is quicker than multiple calls <code>spawn(String)</code> because the pool cached on the first spawn.</th>
</tr>
</thead>
<tbody>
<tr>
<td>batchSpawn(Component, Int32)</td>
<td>Spawn a number instances of the specified component prefab from the appropriate pool. If no pool exists, the one is automatically created for the prefab type. Batch spawning is quicker than multiple calls <code>spawn(Component)</code> because the pool cached on the first spawn.</td>
</tr>
<tr>
<td><strong>batchSpawn(GameObject, Int32)</strong></td>
<td>Spawn a number instances of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls <code>spawn(GameObject)</code> because the pool cached on the first spawn.</td>
</tr>
<tr>
<td><strong>batchSpawn(String, Object, Int32)</strong></td>
<td>Spawn a number instances of a prefab with the specified name from the appropriate pool. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. This method will succeed if the pool has been created before hand. Batch spawning is quicker than multiple calls <code>spawn(String)</code> because the pool cached on the first spawn. If amount specified then the value should be less</td>
</tr>
</tbody>
</table>
batchSpawn(Component, Object, Int32)

Spawn a number instances of the specified prefab and place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to spawn(Component) because the pool cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array; otherwise an out of bounds exception may occur.

batchSpawn(GameObject, GameObject, Int32)

Spawn a number instances of the specified prefab a
place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls `spawn(GameObject)` because the pool is cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array, otherwise an out of bounds exception may occur.

```csharp
batchSpawnT(Component, Int32)
```

Spawn a number of instances of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls `spawnT(Component)`.
batchSpawnT(Component, T, Int32)

Spawn a number instance of the specified prefab a place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls of spawnT(Component) because the pool is cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array otherwise an out of bounds exception may occur.

despawn(Object)

Direct replacement for 'Object.Destroy' for pooling. Allows the specified instance to be
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>despawn(Object, Single)</td>
<td>Direct overload replacement for 'Object.Destroy' for pooling. Allows the specified instance to be returned to the pool and re-used at a later time.</td>
</tr>
<tr>
<td>despawnAll(GameObject)</td>
<td>Calls all spawned instances of the specified prefab back to their pool. This method allows you to pass a prefab such as a 'Bullet' which will subsequently cause all 'Bullet' instances to be despawned.</td>
</tr>
<tr>
<td>despawnAll(GameObject, Single)</td>
<td>Calls all spawned instances of the specified prefab back to their pool. This method allows you to pass a prefab such as a 'Bullet' which will subsequently cause all 'Bullet' instances to be despawned.</td>
</tr>
<tr>
<td>spawn(String)</td>
<td>Spawn an instance of a prefab with the given string as name.</td>
</tr>
</tbody>
</table>
The prefab name is the same name used when creating the pool, or if no name was used then the name of the prefab supplied is substituted. This method will only succeed if the pool has been created before hand.

<table>
<thead>
<tr>
<th>s</th>
<th>spawn(Component)</th>
<th>Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>spawn(GameObject)</td>
<td>Spawn an instance of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</td>
</tr>
<tr>
<td>s</td>
<td>spawn(String, Vector3, Quaternion)</td>
<td>Spawn an instance of a prefab with the specified name. This method will only succeed if the pool has been created.</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>spawn(\text{Component}, \text{Vector3}, \text{Quaternion})</th>
<th>Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the component prefab type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>spawn(\text{GameObject}, \text{Vector3}, \text{Quaternion})</td>
<td>Spawn an instance of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</td>
</tr>
<tr>
<td>spawnT(\text{Component})</td>
<td>Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</td>
</tr>
<tr>
<td>spawnT(\text{Component}, \text{Vector3}, \text{Quaternion})</td>
<td>Spawn an instance of the specified component prefab from the appropriate pool. If no pools exist, then one is automatically created for the prefab type.</td>
</tr>
</tbody>
</table>
created for the component prefab type.

**See Also**

Reference
- UltimatePool Class
- UltimatePooling Namespace
# UltimatePoolbatchDespawn Method

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![s] batchDespawn(IEnumerable&lt;Component&gt;)</td>
<td>Attempts to Despawn all objects and return them to their pool group. Important: All objects in the enumerable collection must have been spawned from the same pool group. If you attempt to return instances from multiple pools using this method then you will invalidate all associated</td>
</tr>
</tbody>
</table>
batchDespawn(IEnumerable<GameObject>)

Attempts to Despawn all objects and return them to their pool group.
Important: All objects in the enumerable collection must have been spawned from the same pool group. If you attempt to return instances from multiple pools using this method then you will invalidate all associated pool groups.

See Also

Reference

UltimatePool Class
UltimatePooling Namespace
UltimatePoolbatchDespawn
Method (IEnumerableComponent)

Attempts to Despawn all objects and return them to their pool group. Important: All objects in the enumerable collection must have been spawned from the same pool group. If you attempt to return instances from multiple pools using this method then you will invalidate all associated pool groups.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#    JavaScript

```csharp
public static void batchDespawn(
    IEnumerable<Component> objects
)
```

Parameters

objects
Type: System.Collections.GenericIEnumerableComponent
An enumerable collection of components that should be despawned

See Also

Reference
UltimatePool Class
batchDespawn Overload
UltimatePooling Namespace
UltimatePool batchDespawn Method
(IEnumerable<GameObject>)

Attempts to Despawn all objects and return them to their pool group. Important: All objects in the enumerable collection must have been spawned from the same pool group. If you attempt to return instances from multiple pools using this method then you will invalidate all associated pool groups.

**Namespace:** UltimatePooling
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**

```csharp
public static void batchDespawn(
    IEnumerable<GameObject> objects
)
```

### Parameters

**objects**
Type: `System.Collections.Generic(IEnumerable<GameObject> objects`
An enumerable collection of objects that should be despawned

### See Also

**Reference**

UltimatePool Class
batchDespawn Overload
UltimatePooling Namespace
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>batchSpawn(String, Int32)</td>
<td>Spawn a number of instance of a prefab with the specified name from the appropriate pool. This method will only succeed if the pool has been created before hand. Batch spawning is quicker than multiple calls to spawn(String) because the pool is cached on the first spawn.</td>
</tr>
<tr>
<td>batchSpawn(Component, Int32)</td>
<td>Spawn a number of instances of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to spawn(Component) because the pool is cached on the first spawn.</td>
</tr>
<tr>
<td>batchSpawnT(Component, Int32)</td>
<td>Spawn a number of instances of the specified component prefab from the</td>
</tr>
</tbody>
</table>
appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to `spawnT(Component)` because the pool is cached on the first spawn.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>batchSpawn(GameObject, Int32)</code></td>
<td>Spawn a number of instances of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to <code>spawn(GameObject)</code> because the pool is cached on the first spawn.</td>
</tr>
<tr>
<td><code>batchSpawn(String, Object, Int32)</code></td>
<td>Spawn a number of instances of a prefab with the specified name from the appropriate pool. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. This method will only succeed if the pool has been created before hand. Batch spawning is quicker than multiple calls to <code>spawn(String)</code> because the pool is cached on the first spawn. If amount is specified then the value...</td>
</tr>
<tr>
<td><strong>batchSpawn(T, Int32)</strong></td>
<td>Spawn a number of instance of the specified prefab and place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker that multiple calls to <code>spawnT(Component)</code> because the pool is cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array otherwise an out of bounds exception may occur.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>batchSpawnT(Component, T, Int32)</strong></td>
<td>Spawn a number of instances of the specified instances of the specified prefab and place the results in the specified array. This overload allows the user to manager the array where the objects will be spawned to avoid garbage generation. If no</td>
</tr>
<tr>
<td><strong>batchSpawnT(Component, Object, Int32)</strong></td>
<td>Spawn a number of instances of the specified prefab and place the results in the specified array. This overload allows the user to manager the array where the objects will be spawned to avoid garbage generation. If no</td>
</tr>
<tr>
<td>should be less than or equal to the length of the array otherwise an out of bounds exception may occur.</td>
<td></td>
</tr>
</tbody>
</table>
pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to `spawn(Component)` because the pool is cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array otherwise an out of bounds exception may occur.

### batchSpawn(GameObject, GameObject, Int32)

Spawn a number of instances of the specified prefab and place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to `spawn(GameObject)` because the pool is cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array otherwise an out of bounds exception may occur.
See Also

Reference
UltimatePool Class
UltimatePooling Namespace
UltimatePool
batchSpawn Method (String, Int32)

Spawn a number of instance of a prefab with the specified name from the appropriate pool. This method will only succeed if the pool has been created before hand. Batch spawning is quicker than multiple calls to spawn(String) because the pool is cached on the first spawn.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#  JavaScript

```csharp
public static IEnumerable<Object> batchSpawn(
    string prefabName,
    int amount
)
```

Parameters

prefabName
  Type: System.String
  The name of the prefab to spawn from

amount
  Type: System.Int32
  The amount of instances to create from this prefab

Return Value

Type: IEnumerable<Object>
An enumeration of the spawned instances

See Also
Reference
UltimatePool Class
batchSpawn Overload
UltimatePooling Namespace
UltimatePoolbatchSpawn Method (Component, Int32)

Spawn a number of instances of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to spawn(Component) because the pool is cached on the first spawn.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```csharp
public static IEnumerable<Object> batchSpawn(
    Component prefab,
    int amount
)
```

Parameters

prefab
Type: Component
The component prefab to spawn from

amount
Type: System.Int32
The amount of instances to create from this prefab

Return Value
Type: IEnumerable<Object>
An enumeration of the spawned instances
See Also

Reference
UltimatePool Class
batchSpawn Overload
UltimatePooling Namespace
Ultimate PoolbatchSpawn\(T\) Method (Component, Int32)

Spawn a number of instances of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to `spawn\(T\)(Component)` because the pool is cached on the first spawn.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

```csharp
public static IEnumerable\<T\> batchSpawn\<T\>(
    Component prefab,
    int amount
)
where T : Object
```

### Parameters

- **prefab**
  - Type: **Component**  
    - The component prefab to spawn from

- **amount**
  - Type: **SystemInt32**  
    - The amount of instances to create from this prefab

### Type Parameters

- **T**
  - The type of object to return the instances as
Return Value
Type: `IEnumerable<T>`
An enumeration of the spawned instances

See Also

Reference
UltimatePool Class
batchSpawn Overload
UltimatePooling Namespace
Ultimate Pooling Scripting Reference

UltimatePoolbatchSpawn Method (GameObject, Int32)

Spawn a number of instances of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to spawn(GameObject) because the pool is cached on the first spawn.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#  JavaScript  Copy

```csharp
public static IEnumerable<GameObject> batchSpawn(
    GameObject prefab,
    int amount
)
```

Parameters

- **prefab**
  - Type: GameObject
  - The prefab to spawn from

- **amount**
  - Type: System.Int32
  - The amount of instances to create from this prefab

Return Value

- Type: IEnumerable<GameObject>
- An enumeration of the spawned instances

See Also
Reference
UltimatePool Class
batchSpawn Overload
UltimatePooling Namespace
UltimatePool batchSpawn Method (String, Object, Int32)

Spawn a number of instances of a prefab with the specified name from the appropriate pool. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. This method will only succeed if the pool has been created beforehand. Batch spawning is quicker than multiple calls to `spawn(String)` because the pool is cached on the first spawn. If `amount` is specified then the value should be less than or equal to the length of the array otherwise an out of bounds exception may occur.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

#### C#  
```csharp
public static void batchSpawn(
    string prefabName,
    Object[] objects,
    int amount = -1
)
```

#### JavaScript

```javascript
// Not applicable
```

### Parameters

- **prefabName**  
  - Type: `System.String`  
  - The name of the prefab to spawn from

- **objects**  
  - Type: `Object`  
  - The array to store the spawned objects in

- **amount** *(Optional)*
Type: SystemInt32
The amount of objects to spawn. If the value is set to -1 then the array is filled

See Also

Reference
UltimatePool Class
batchSpawn Overload
UltimatePooling Namespace
Ultimate Pool batchSpawn<T> Method (Component, T, Int32)

Spawn a number of instance of the specified prefab and place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker that multiple calls to spawnT(Component) because the pool is cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array otherwise an out of bounds exception may occur.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

**Syntax**

```csharp
public static void batchSpawn<T>(
    Component prefab,
    T[] objects,
    int amount = -1
)
where T : Object
```

**Parameters**

- **prefab**  
  Type: Component  
  The prefab to spawn from

- **objects**  
  Type: T  
  The array to store the spawned objects in
**amount (Optional)**
Type: `SystemInt32`
The amount of objects to spawn. If the value is set to -1 then the array is filled

**Type Parameters**

\( T \)
The type of prefab that will be spawned

⚠️ **See Also**

**Reference**

- UltimatePool Class
- batchSpawn Overload
- UltimatePooling Namespace
UltimatePoolbatchSpawn Method (Component, Object, Int32)

Spawn a number of instances of the specified prefab and place the results in the specified array. This overload allows the user to manager the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to spawn(Component) because the pool is cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array otherwise an out of bounds exception may occur.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>public static void batchSpawn(</td>
</tr>
<tr>
<td></td>
<td>Component prefab,</td>
</tr>
<tr>
<td></td>
<td>Object[] objects,</td>
</tr>
<tr>
<td></td>
<td>int amount = -1</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
</tbody>
</table>

### Parameters

**prefab**
- Type: Component
- The prefab to spawn from

**objects**
- Type: Object
- The array to store the spawned objects in

**amount** *(Optional)*
Type: **SystemInt32**  
The amount of objects to spawn. If the value is set to -1 then the array is filled

See Also

Reference

UltimatePool Class
batchSpawn Overload
UltimatePooling Namespace
Ultimate Pool\nbatchSpawn Method

GameObject, GameObject, Int32)

Spawn a number of instances of the specified prefab and place the results in the specified array. This overload allows the user to manage the array where the objects will be spawned to avoid garbage generation. If no pool exists, then one is automatically created for the prefab type. Batch spawning is quicker than multiple calls to spawn(GameObject) because the pool is cached on the first spawn. If amount is specified then the value should be less than or equal to the length of the array otherwise an out of bounds exception may occur.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```
public static void batchSpawn(
    GameObject prefab,
    GameObject[] objects,
    int amount = -1
)
```

Parameters

`prefab`
Type: GameObject
The prefab to spawn from

`objects`
Type: GameObject
The array to store the spawned objects in

`amount (Optional)"
Type: SystemInt32
The amount of objects to spawn. If the value is set to -1 then the array is filled

See Also

Reference
UltimatePool Class
batchSpawn Overload
UltimatePooling Namespace
# UltimatePool despawn Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>despawn(Object)</td>
<td>Direct replacement for 'Object.Destroy' for pooling. Allows the specified instance to be returned to the pool and re-used at a later time.</td>
</tr>
<tr>
<td>despawn(Object, Single)</td>
<td>Direct overload replacement for 'Object.Destroy' for pooling. Allows the specified instance to be returned to the pool and re-used at a later time.</td>
</tr>
</tbody>
</table>

## See Also

Reference

*UltimatePool Class*

*UltimatePooling Namespace*
UltimatePooling Scripting Reference

UltimatePooling Despawn Method (Object)

Direct replacement for 'Object.Destroy' for pooling. Allows the specified instance to be returned to the pool and re-used at a later time.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

```csharp
public static void despawn(
    Object instance
)
```

### Parameters

**instance**

- **Type:** Object
- A reference to a spawned instance

### See Also

**Reference**

- UltimatePool Class
- despawn Overload
- UltimatePooling Namespace
UltimatePooling Scripting Reference

UltimatePool despawn Method (Object, Single)

Direct overload replacement for 'Object.Destroy' for pooling. Allows the specified instance to be returned to the pool and re-used at a later time.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
Version: 0.0.0.0

## Syntax

```csharp
public static void despawn(
    Object instance,
    float time
)
```

### Parameters

**instance**
- **Type:** Object
- A reference to a spawned instance

**time**
- **Type:** System.Single
- The amount of time to wait before despawning the instance

## See Also

Reference  
UltimatePool Class  
despawn Overload  
UltimatePooling Namespace
## UltimatePool despawnAll Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>despawnAll(GameObject)</td>
<td>Calls all spawned instances of the specified prefab back to their pool. This method allows you to pass a prefab such as a 'Bullet' which will subsequently cause all 'Bullet' instances to be despawned.</td>
</tr>
<tr>
<td>despawnAll(GameObject, Single)</td>
<td>Calls all spawned instances of the specified prefab back to their pool. This method allows you to pass a prefab such as a 'Bullet' which will subsequently cause all 'Bullet' instances to be despawned.</td>
</tr>
</tbody>
</table>

---

### See Also

Reference

UltimatePool Class
UltimatePooling Namespace
UltimatePooling Scripting Reference

UltimatePool
despawnAll Method
(GameObject)

Calls all spawned instances of the specified prefab back to their pool. This method allows you to pass a prefab such as a 'Bullet' which will subsequently cause all 'Bullet' instances to be despawned.

Namespace: UltimatePooling
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```csharp
public static void despawnAll(
    GameObject prefab
)
```

Parameters

`prefab`

Type: GameObject
The prefab to despawn all instances of

See Also

Reference
UltimatePool Class
despawnAll Overload
UltimatePooling Namespace
Ultimate Pool\n\ndespawnAll Method (GameObject, Single)

Calls all spawned instances of the specified prefab back to their pool. This method allows you to pass a prefab such as a 'Bullet' which will subsequently cause all 'Bullet' instances to be despawned.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
Version: 0.0.0.0

### Syntax

**C#**  
```csharp
public static void despawnAll(
    GameObject prefab,
    float time
)
```

**JavaScript**

### Parameters

- **prefab**
  - Type: `GameObject`
  - The prefab to despawn all instances of

- **time**
  - Type: `System.Single`
  - The amount of time to wait before despawning

### See Also

Reference
- UltimatePool Class
- despawnAll Overload
- UltimatePooling Namespace
**Ultimate Poolspawn Method**

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>spawn(String)</code></td>
<td>Spawn an instance of a prefab with the specified name. The prefab name is the same name used when creating the pool, or if no name is used then the name of the prefab supplied is substituted. This method will only succeed if the pool has been created before hand.</td>
</tr>
<tr>
<td><code>spawn(Component)</code></td>
<td>Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</td>
</tr>
<tr>
<td><code>spawnT(Component)</code></td>
<td>Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</td>
</tr>
<tr>
<td><code>spawn(GameObject)</code></td>
<td>Spawn an instance of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</td>
</tr>
<tr>
<td><code>spawn(String, Vector3, Quaternion)</code></td>
<td>Spawn an instance of a prefab with the specified name. This</td>
</tr>
</tbody>
</table>
method will only succeed if the pool has been created beforehand.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spawn (Component, Vector3, Quaternion)</td>
<td>Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the component prefab type.</td>
</tr>
<tr>
<td>spawnT (Component, Vector3, Quaternion)</td>
<td>Spawn an instance of the specified component prefab from the appropriate pool. If no pools exists, then one is automatically created for the component prefab type.</td>
</tr>
<tr>
<td>spawn (GameObject, Vector3, Quaternion)</td>
<td>Spawn an instance of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.</td>
</tr>
</tbody>
</table>

See Also

Reference
UltimatePool Class
UltimatePooling Namespace
UltimatePoolspawn Method (String)

Spawn an instance of a prefab with the specified name. The prefab name is the same name used when creating the pool, or if no name is used then the name of the prefab supplied is substituted. This method will only succeed if the pool has been created before hand.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| public static GameObject spawn(  
|     string prefabName   
| )                     |

### Parameters

**prefabName**  
Type: System.String  
The name of the prefab to spawn from

### Return Value

Type: GameObject  
An instance of the prefab with the specified name

### See Also

Reference  
UltimatePool Class  
spawn Overload  
UltimatePooling Namespace
UltimatePooling Scripting Reference

UltimatePoolspawn Method (Component)

Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

**C#**

```csharp
public static Object spawn(
    Component prefab
)
```

**JavaScript**

```
```

### Parameters

`prefab`

Type: **Component**  
The component prefab to spawn from

### Return Value

Type: **Object**  
An instance of the prefab supplied

### See Also

**Reference**  
UltimatePool Class  
spawn Overload  
UltimatePooling Namespace
UltimatePoolspawn\text{T} Method (Component)

Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

## Syntax

```csharp
public static T spawn\text<T>(
    Component prefab
)
where T : Object
```

### Parameters

- **prefab**
  - Type: **Component**  
    - The component prefab to spawn from

### Type Parameters

- **T**  
  - The type of object to return the instance as

### Return Value

- Type: **T**  
  - An instance of the prefab supplied

## See Also
Reference
UltimatePool Class
spawn Overload
UltimatePooling Namespace
Ultimate Pool

spawn Method (GameObject)

Spawn an instance of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.

**Namespace**: UltimatePooling

**Assembly**: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

**Syntax**

```csharp
public static GameObject spawn(
    GameObject prefab
)
```

**Parameters**

*prefab*

Type: `GameObject`

The prefab to spawn from

**Return Value**

Type: `GameObject`

An instance of the prefab supplied

**See Also**

Reference

UltimatePool Class

spawn Overload

UltimatePooling Namespace
UltimatePoolspawn Method (String, Vector3, Quaternion)

Spawn an instance of a prefab with the specified name. This method will only succeed if the pool has been created before hand.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

⚠️ Syntax

```csharp
public static GameObject spawn(
    string prefabName,
    Vector3 position,
    Quaternion identity
)
```

**Parameters**

*prefabName*
  - Type: `SystemString`
  - The name of the prefab to spawn from

*position*
  - Type: `Vector3`
  - The position to spawn the prefab at

*identity*
  - Type: `Quaternion`
  - The initial rotation to spawn the prefab with

**Return Value**

*Type: `GameObject`*
  - An instance of the prefab with the specified name
See Also

Reference
UltimatePool Class
spawn Overload
UltimatePooling Namespace
Ultimate Pool\_spawn Method 
(Component, Vector3, Quaternion)

Spawn an instance of the specified component prefab from the appropriate pool. If no pool exists, then one is automatically created for the component prefab type.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>public static</strong> Object spawn(</td>
<td></td>
</tr>
</tbody>
</table>
  Component **prefab**,  |
  Vector3 **position**,  |
  Quaternion **rotation**  |
| ) |             |

**Parameters**

**prefab**
- Type: **Component**  
  The component prefab to spawn from

**position**
- Type: **Vector3**  
  The position to spawn the prefab at

**rotation**
- Type: **Quaternion**  
  The initial rotation to spawn the prefab with

**Return Value**
- Type: **Object**
An instance of the component prefab supplied

See Also

Reference
UltimatePool Class
spawn Overload
UltimatePooling Namespace
UltimatePoolspawn\text{T} Method
(\text{Component, Vector3, Quaternion})

Spawn an instance of the specified component prefab from the
appropriate pool. If no pools exists, then one is automatically created
for the component prefab type.

\textbf{Namespace}: UltimatePooling
\textbf{Assembly}: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

\section*{Syntax}

\begin{verbatim}
public static \text{T} spawn<\text{T}>(
    Component \text{prefab},
    Vector3 \text{position},
    Quaternion \text{rotation}
)
\end{verbatim}

\textit{where} \text{T} : Object

\section*{Parameters}

\textit{prefab}
\begin{itemize}
    \item Type: \textbf{Component}
    \item The component prefab to spawn from
\end{itemize}

\textit{position}
\begin{itemize}
    \item Type: \textbf{Vector3}
    \item The position to spawn the prefab at
\end{itemize}

\textit{rotation}
\begin{itemize}
    \item Type: \textbf{Quaternion}
    \item The initial rotation to spawn the prefab with
\end{itemize}

\section*{Type Parameters}
The type of object to return the instance as

Return Value
Type: \( T \)
An instance of the component prefab supplied

See Also

Reference
UltimatePool Class
spawn Overload
UltimatePooling Namespace
UltimatePoolspawn Method (GameObject, Vector3, Quaternion)

Spawn an instance of the specified prefab from the appropriate pool. If no pool exists, then one is automatically created for the prefab type.

**Namespace:** UltimatePooling  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

## Syntax

```csharp
public static GameObject spawn(
    GameObject prefab,
    Vector3 position,
    Quaternion rotation
)
```

### Parameters

- **prefab**
  - Type: GameObject  
  - The prefab to spawn from

- **position**
  - Type: Vector3  
  - The position to spawn the prefab at

- **rotation**
  - Type: Quaternion  
  - The initial rotation to spawn the prefab with

### Return Value
Type: **GameObject**
An instance of the prefab supplied

See Also

Reference
UltimatePool Class
spawn Overload
UltimatePooling Namespace
Ultimate Pooling Scripting Reference

UltimatePool Properties

The UltimatePool type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pools</td>
<td>Access the pool manager which maintains all existing pools. Allows pools to be created and destroyed.</td>
</tr>
</tbody>
</table>

Top

See Also

Reference
UltimatePool Class
UltimatePooling Namespace
UltimatePool\n
Pools Property

Access the pool manager which maintains all existing pools. Allows pools to be created and destroyed.

**Namespace:** UltimatePooling

**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

## Syntax

```csharp
public static PoolManager Pools { get; }
```

**Property Value**

Type: `PoolManager`

## See Also

Reference

- UltimatePool Class
- UltimatePooling Namespace
# UltimatePooling.Demo Namespace

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Star] Benchmark</td>
<td></td>
</tr>
<tr>
<td>![Star] Ex0_SpawnExample</td>
<td>This class shows how to spawn and despawn prefab objects.</td>
</tr>
<tr>
<td>![Star] Ex1_SpawnAtExample</td>
<td>This class shows how to spawn prefab objects using the overloaded methods.</td>
</tr>
<tr>
<td>![Star] Ex3_CreatePoolExample</td>
<td>This class shows how to create a new pool for a prefab type and initialize its spawning values.</td>
</tr>
<tr>
<td>![Star] Ex4_CreateResourcesPoolExample</td>
<td>This example shows how to create a new resources pool for a prefab in the</td>
</tr>
<tr>
<td><strong>Ex5_DestroyPoolExample</strong></td>
<td>This example shows how to destroy a pool at runtime. Note that all pooled items are destroyed but all items spawned by this pool will remain in the scene.</td>
</tr>
</tbody>
</table>
Benchmark Class


Inheritance Hierarchy


Namespace: UltimatePooling.Demo
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class Benchmark : MonoBehaviour</td>
<td></td>
</tr>
</tbody>
</table>

The Benchmark type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark</td>
<td>Initializes a new instance of the Benchmark class</td>
</tr>
</tbody>
</table>

Fields
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefab</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference
UltimatePooling.Demo Namespace
Benchmark Constructor

Initializes a new instance of the Benchmark class

**Namespace:** UltimatePooling.Demo  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

⚠️ **Syntax**

```csharp
public Benchmark()
```

⚠️ **See Also**

Reference

Benchmark Class  
UltimatePooling.Demo Namespace
Benchmark Fields

The Benchmark type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefab</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference

Benchmark Class
UltimatePooling.Demo Namespace
Benchmark prefab Field


**Namespace:** UltimatePooling.Demo  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>GameObject</td>
</tr>
</tbody>
</table>

**Field Value**  
**Type:** GameObject

### See Also

**Reference**  
Benchmark Class  
UltimatePooling.Demo Namespace
Ex0_SpawnExample Class

This class shows how to spawn and despawn prefab objects.

▲ Inheritance Hierarchy

System
  Object
    Component
      Behaviour
        MonoBehaviour
          UltimatePooling.DemoEx0_SpawnExample

Namespace: UltimatePooling.Demo
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

▲ Syntax

```
public class Ex0_SpawnExample : MonoBehaviour
```

The Ex0_SpawnExample type exposes the following members.

▲ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex0_SpawnExample</td>
<td>Initializes a new instance of the Ex0_SpawnExample class</td>
</tr>
</tbody>
</table>

▲ Fields
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefab</td>
<td>The prefab we want to spawn - Assigned in the editor inspector.</td>
</tr>
</tbody>
</table>

See Also

Reference
UltimatePooling.Demo Namespace
Ex0_SpawnExample Constructor

Initializes a new instance of the Ex0_SpawnExample class

Namespace: UltimatePooling.Demo
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#                   JavaScript
public Ex0_SpawnExample()   

See Also

Reference
Ex0_SpawnExample Class
UltimatePooling.Demo Namespace
Ex0_SpawnExample Fields

The Ex0_SpawnExample type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefab</td>
<td>The prefab we want to spawn - Assigned in the editor inspector.</td>
</tr>
</tbody>
</table>

See Also

Reference
- Ex0_SpawnExample Class
- UltimatePooling.Demo Namespace
Ex0_SpawnExample prefab Field

The prefab we want to spawn - Assigned in the editor inspector.

**Namespace:** UltimatePooling.Demo  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) **Version:** 0.0.0.0

### Syntax

```csharp
public GameObject prefab
```

### Field Value

**Type:** GameObject

### See Also

**Reference**

- Ex0_SpawnExample Class
- UltimatePooling.Demo Namespace
Ex1_SpawnAtExample Class

This class shows how to spawn prefab objects using the overloaded methods.

Inheritance Hierarchy

```
System
  Object
    Component
      Behaviour
        MonoBehaviour
            UltimatePooling.DemoEx1_SpawnAtExample
```

Namespace: UltimatePooling.Demo
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

```
C#        JavaScript
public class Ex1_SpawnAtExample : MonoBehaviour
```

The `Ex1_SpawnAtExample` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex1_SpawnAtExample</td>
<td>Initializes a new instance of the <code>Ex1_SpawnAtExample</code> class</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>prefab</td>
<td>The prefab we want to spawn - Assigned in the editor inspector.</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

UltimatePooling.Demo Namespace
Ex1_SpawnAtExample Constructor

Initializes a new instance of the Ex1_SpawnAtExample class

**Namespace:** UltimatePooling.Demo  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public Ex1_SpawnAtExample()</code></td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- Ex1_SpawnAtExample Class
- UltimatePooling.Demo Namespace
Ex1_SpawnAtExample Fields

The Ex1_SpawnAtExample type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefab</td>
<td>The prefab we want to spawn - Assigned in the editor inspector.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

Ex1_SpawnAtExample Class

UltimatePooling.Demo Namespace
Ex1_SpawnAtExample prefab Field

The prefab we want to spawn - Assigned in the editor inspector.

**Namespace:** UltimatePooling.Demo  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>public GameObject prefab</td>
<td></td>
</tr>
</tbody>
</table>

### Field Value

**Type:** `GameObject`

### See Also

**Reference**  
- Ex1_SpawnAtExample Class  
- UltimatePooling.Demo Namespace
Ex3_CreatePoolExample Class

This class shows how to create a new pool for a prefab type and initialize its spawning values.

Inheritance Hierarchy

System
  Object
  Component
  Behaviour
    MonoBehaviour
      UltimatePooling.Demo
          Ex3_CreatePoolExample

Namespace: UltimatePooling.Demo
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class Ex3_CreatePoolExample : MonoBehaviour</td>
<td></td>
</tr>
</tbody>
</table>

The Ex3_CreatePoolExample type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex3_CreatePoolExample</td>
<td>Initializes a new instance of the Ex3_CreatePoolExample class</td>
</tr>
</tbody>
</table>

Top
### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefab</td>
<td>The prefab we want to spawn - Assigned in the editor inspector.</td>
</tr>
</tbody>
</table>

### See Also

Reference

[UltimatePooling.Demo Namespace](#)
Ex3_CreatePoolExample Constructor

Initializes a new instance of the Ex3_CreatePoolExample class

**Namespace:** UltimatePooling.Demo

**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>public Ex3_CreatePoolExample()</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Reference**

Ex3_CreatePoolExample Class
UltimatePooling.Demo Namespace
Ex3_CreatePoolExample Fields

The Ex3_CreatePoolExample type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefab</td>
<td>The prefab we want to spawn - Assigned in the editor inspector.</td>
</tr>
</tbody>
</table>

See Also

Reference
Ex3_CreatePoolExample Class
UltimatePooling.Demo Namespace
Ex3_CreatePoolExample prefab Field

The prefab we want to spawn - Assigned in the editor inspector.

**Namespace:** UltimatePooling.Demo  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll)  
**Version:** 0.0.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>public GameObject prefab</td>
<td></td>
</tr>
</tbody>
</table>

Field Value  
Type: **GameObject**

### See Also

**Reference**  
Ex3_CreatePoolExample Class  
UltimatePooling.Demo Namespace
Ex4_CreateResourcesPoolExample Class

This example shows how to create a new resources pool for a prefab in the resources folder.

Inheritance Hierarchy

```
System
  Object
    Component
      Behaviour
        MonoBehaviour
          UltimatePooling.DemoEx4_CreateResourcesPoolExample
```

Namespace: UltimatePooling.Demo
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#  

```
public class Ex4_CreateResourcesPoolExample : MonoBehaviour
```

JavaScript

```
```

The Ex4_CreateResourcesPoolExample type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex4_CreateResourcesPoolExample</td>
<td>Initializes a new instance of the class</td>
</tr>
</tbody>
</table>
### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefabName</td>
<td>The name of the prefab we want to spawn - Located in the resources folder.</td>
</tr>
</tbody>
</table>

### See Also

Reference

_UltimatePooling.Demo Namespace_
Ex4_CreateResourcesPoolExample Constructor

Initializes a new instance of the Ex4_CreateResourcesPoolExample class

**Namespace:** UltimatePooling.Demo  
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

## Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>public Ex4_CreateResourcesPoolExample()</td>
<td></td>
</tr>
</tbody>
</table>

## See Also

Reference
- Ex4_CreateResourcesPoolExample Class
- UltimatePooling.Demo Namespace
Ex4_CreateResourcesPoolExample Fields

The Ex4_CreateResourcesPoolExample type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefabName</td>
<td>The name of the prefab we want to spawn - Located in the resources folder.</td>
</tr>
</tbody>
</table>

See Also

Reference
Ex4_CreateResourcesPoolExample Class
UltimatePooling.Demo Namespace
Ex4_CreateResourcesPoolExample Field

The name of the prefab we want to spawn - Located in the resources folder.

**Namespace:** UltimatePooling.Demo  
**Assembly:** Assembly-CSHarp (in Assembly-CSHarp.dll)  
**Version:** 0.0.0.0

### Syntax

```csharp
public string prefabName
```

### Field Value

**Type:** String

### See Also

**Reference**  
Ex4_CreateResourcesPoolExample Class  
UltimatePooling.Demo Namespace
Ex5_DestroyPoolExample Class

This example shows how to destory a pool at runtime. Note that all pooled iters are destroyed but all items spawned by this pool will remain in the scene.

Inheritance Hierarchy

```
System
  Object
  Component
    Behaviour
      MonoBehaviour
        UltimatePooling.DemoEx5_DestroyPoolExample
```

Namespace: UltimatePooling.Demo
Assembly: Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

Syntax

C#   | JavaScript
---|---

```
public class Ex5_DestroyPoolExample : MonoBehaviour
```

The Ex5_DestroyPoolExample type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌼 Ex5_DestroyPoolExample</td>
<td>Initializes a new instance of the Ex5_DestroyPoolExample class</td>
</tr>
</tbody>
</table>
### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefab</td>
<td>The prefab we want to spawn - Assigned in the editor inspector.</td>
</tr>
</tbody>
</table>

### See Also

Reference

UltimatePooling.Demo Namespace
Ex5_DestroyPoolExample Constructor

Initializes a new instance of the Ex5_DestroyPoolExample class

**Namespace:** UltimatePooling.Demo
**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

### Syntax

**C#**

```csharp
public Ex5_DestroyPoolExample()
```

**JavaScript**

```javascript
public Ex5_DestroyPoolExample()
```

### See Also

**Reference**

Ex5_DestroyPoolExample Class
UltimatePooling.Demo Namespace
Ex5_DestroyPoolExample Fields

The Ex5_DestroyPoolExample type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefab</td>
<td>The prefab we want to spawn - Assigned in the editor inspector.</td>
</tr>
</tbody>
</table>

See Also

Reference
- Ex5_DestroyPoolExample Class
- UltimatePooling.Demo Namespace
Ex5_DestroyPoolExample prefab

Field

The prefab we want to spawn - Assigned in the editor inspector.

**Namespace:** UltimatePooling.Demo

**Assembly:** Assembly-CSharp (in Assembly-CSharp.dll) Version: 0.0.0.0

⚠️ **Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>public GameObject prefab</td>
<td></td>
</tr>
</tbody>
</table>

Field Value

Type: GameObject

⚠️ **See Also**

Reference

Ex5_DestroyPoolExample Class
UltimatePooling.Demo Namespace