Shared References

Smart pointer type that cannot be uninitialized or assigned null.

A **Shared Reference** is a strong, non-nullable **Smart Pointer** for data objects outside of the Engine's `UObject` system. This means you cannot reset a Shared Reference, assign a null object to it, or create an empty one. Because of this, Shared References always contain a valid object, and do not even have an `IsValid` method. When choosing between Shared References and **Shared Pointers**, Shared References are the preferred option unless you need an empty or nullable object. If you need potentially-empty or nullable references, you should use Shared Pointers instead.

> Unlike a standard C++ reference, a Shared Reference can be reassigned to another object after creation.

Declaration and Initialization

Shared References are non-nullable, so initialization requires a data object. Attempting to create a Shared Reference without a valid object will not compile, and attempting to initialize a Shared Reference to a null pointer variable.

```
1  // Create a shared reference to a new node
2  TSharedRef<FMyObjectType> NewReference = MakeShared<FMyObjectType>();
3
```

Copy full snippet

Attempting to create a Shared Reference without a valid object will not compile:

```
1  // Neither of these will compile:
```

```
TSharedRef<FMyObjectType> UnassignedReference;
TSharedRef<FMyObjectType> NullAssignedReference = nullptr;
// This will compile, but will assert if NullObject is actually null.
TSharedRef<FMyObjectType> NullAssignedReference = NullObject;

```

Copy full snippet

Converting Between Shared Pointers and Shared References

Converting between Shared Pointers and Shared References is a common practice. Shared References implicitly convert to Shared Pointers, and provide the additional guarantee that the new Shared Pointer will reference a valid object. Conversion is handled by the normal syntax:

```
TSharedPtr<FMyObjectType> MySharedPointer = MySharedReference;

```

Copy full snippet

You can create a Shared Reference from a Shared Pointer with the `Shared Pointer` function, `ToSharedRef`, as long as the Shared Pointer references a non-null object. Attempting to create a Shared Reference from a null Shared Pointer will cause the program to assert.

```
// Ensure your shared pointer is valid before dereferencing to avoid a
potential assertion.
If (MySharedPointer.IsValid())
{
MySharedReference = MySharedPointer.ToSharedRef();
}

```

Copy full snippet

Comparison

You can test Shared References against each other for equality. In this context, equality means referencing the same object.

```
TSharedRef<FMyObjectType> ReferenceA, ReferenceB;
if (ReferenceA == ReferenceB)
{
// ...
}
```

Copy full snippet