```
<Foo>
    <BarSubstitute />
</Foo>
```

@

C

O

d

e

{

[

F

a

C

t

p

u

b

1

i

C

V

O

i

d

S

h

O

u

1

d

H

a

V

e

B

a

r

S

u

b

S

t

i

t

U

t

e

I

n

S

t

e

a

d

O

f

B

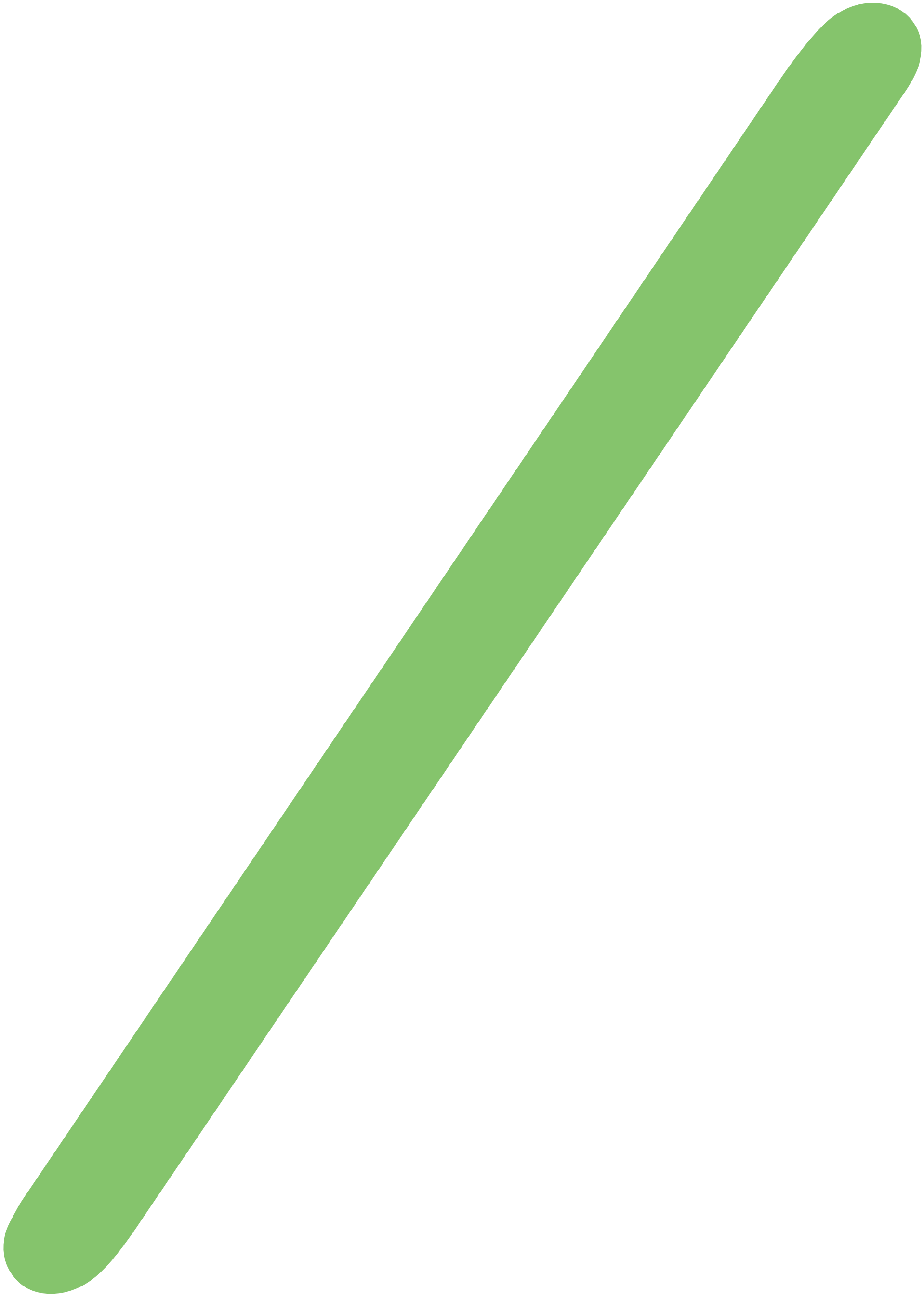
a

r

(

)

{



T

e

1

b

U

n

i

t

t

O

r

e

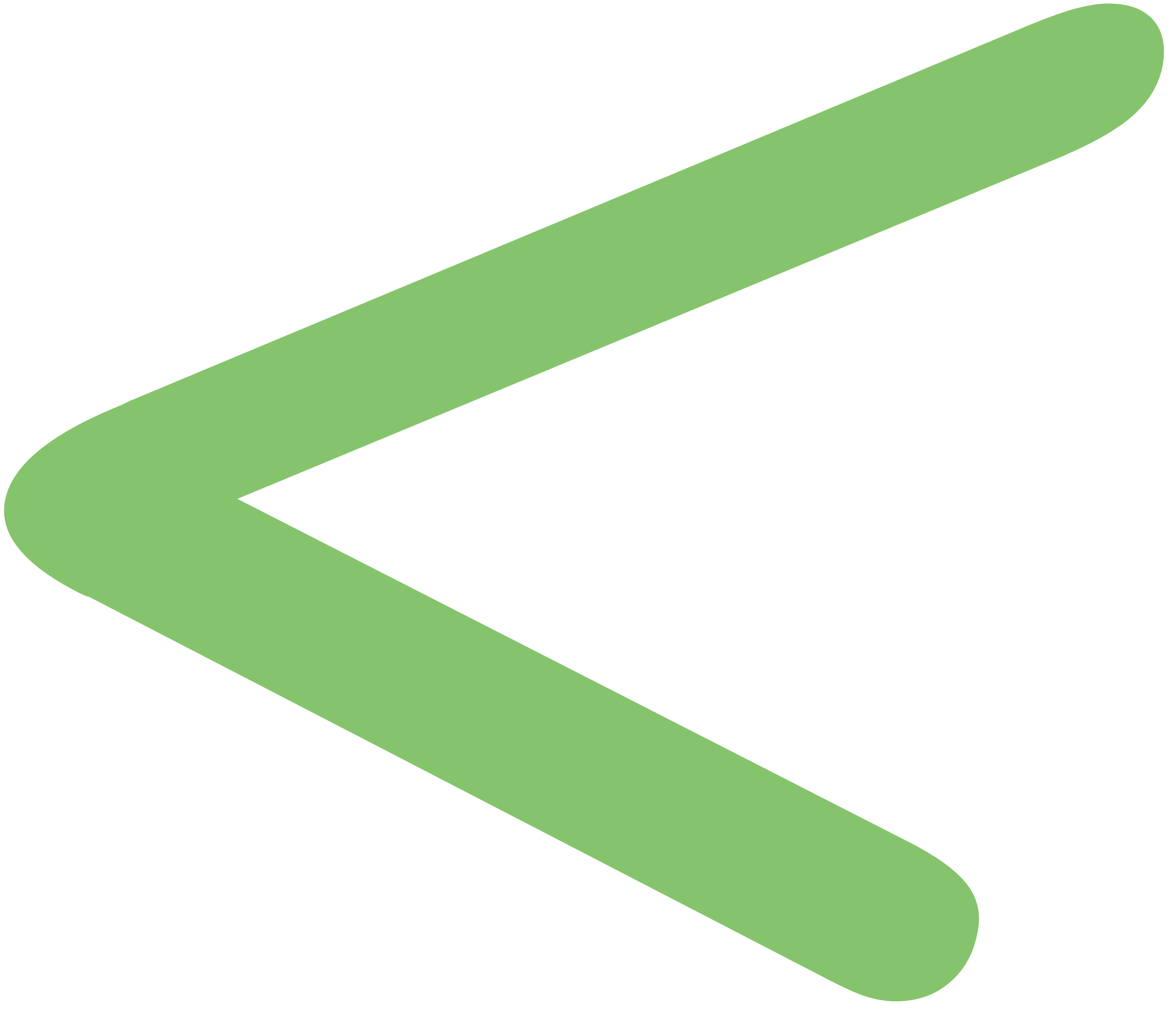
p

1

a

C

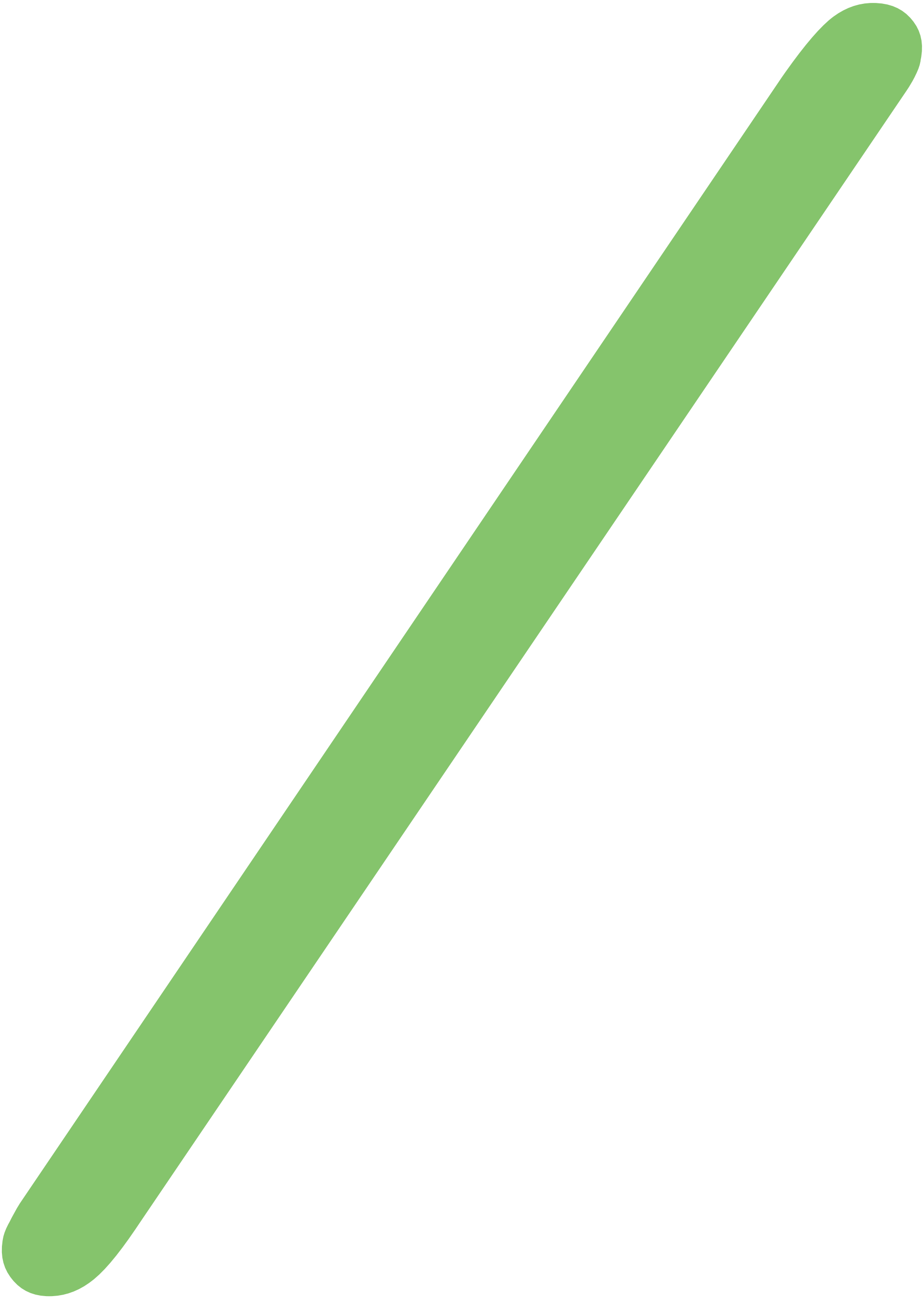
e

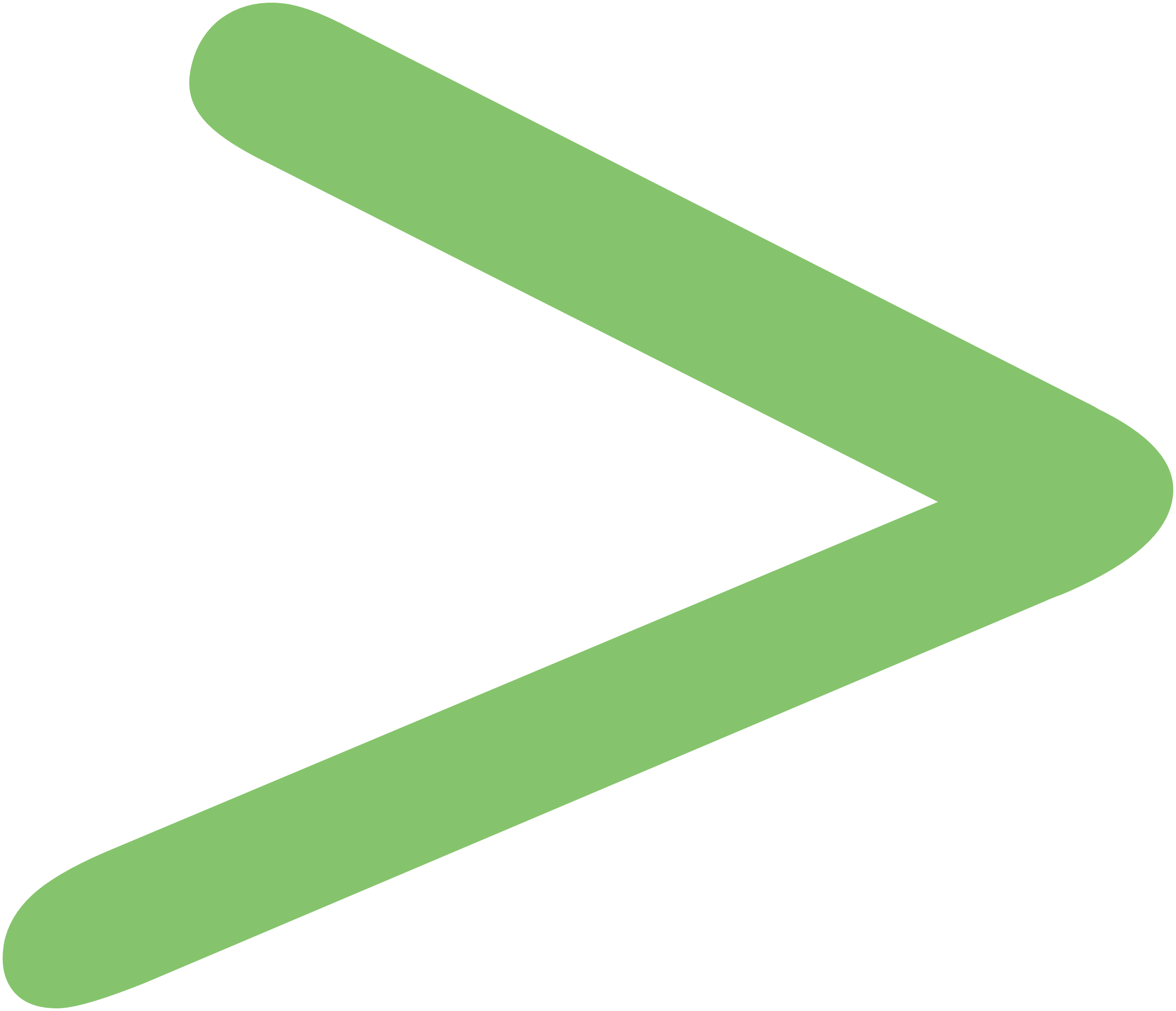


B

a

r





W

i

t

h

B

a

r

S

u

b

S

t

i

u

t

e

C

O

m

p

O

n

e

t

F

a

C

t

O

r

e

S

A

d

d

<

B

a

r

B

a

r

S

u

b

S

t

i

t

u

e

¡

R

n

d

e

r

O

u

r

C

m

p

n

e

n

t

I

R

e

n

d

e

r

d

F

r

a

g

m

e

n

t

C

U

t

$=$

R

e

n

d

e

r

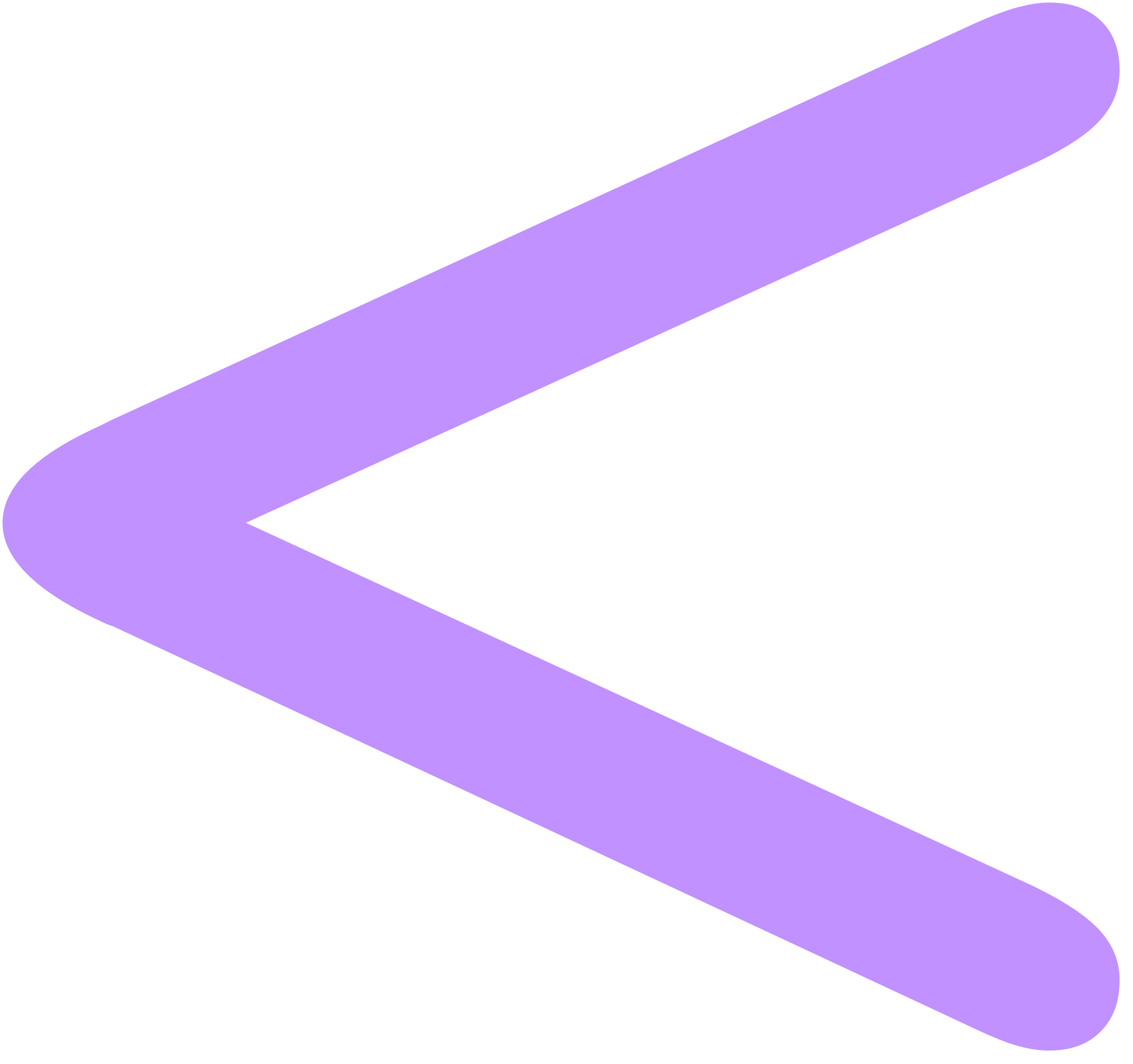
(

@

F

O

O



F

O

A

S

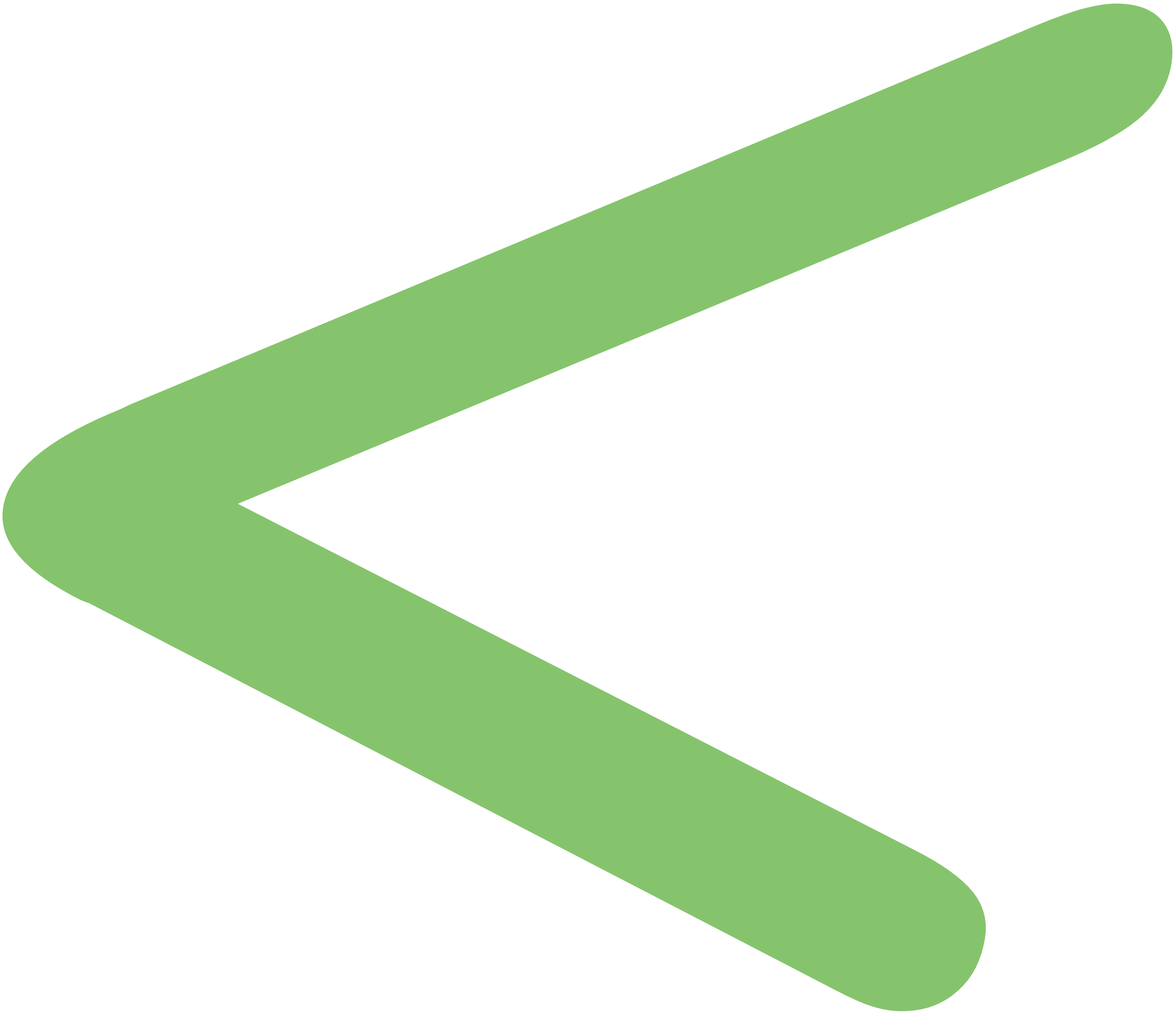
S

r

t

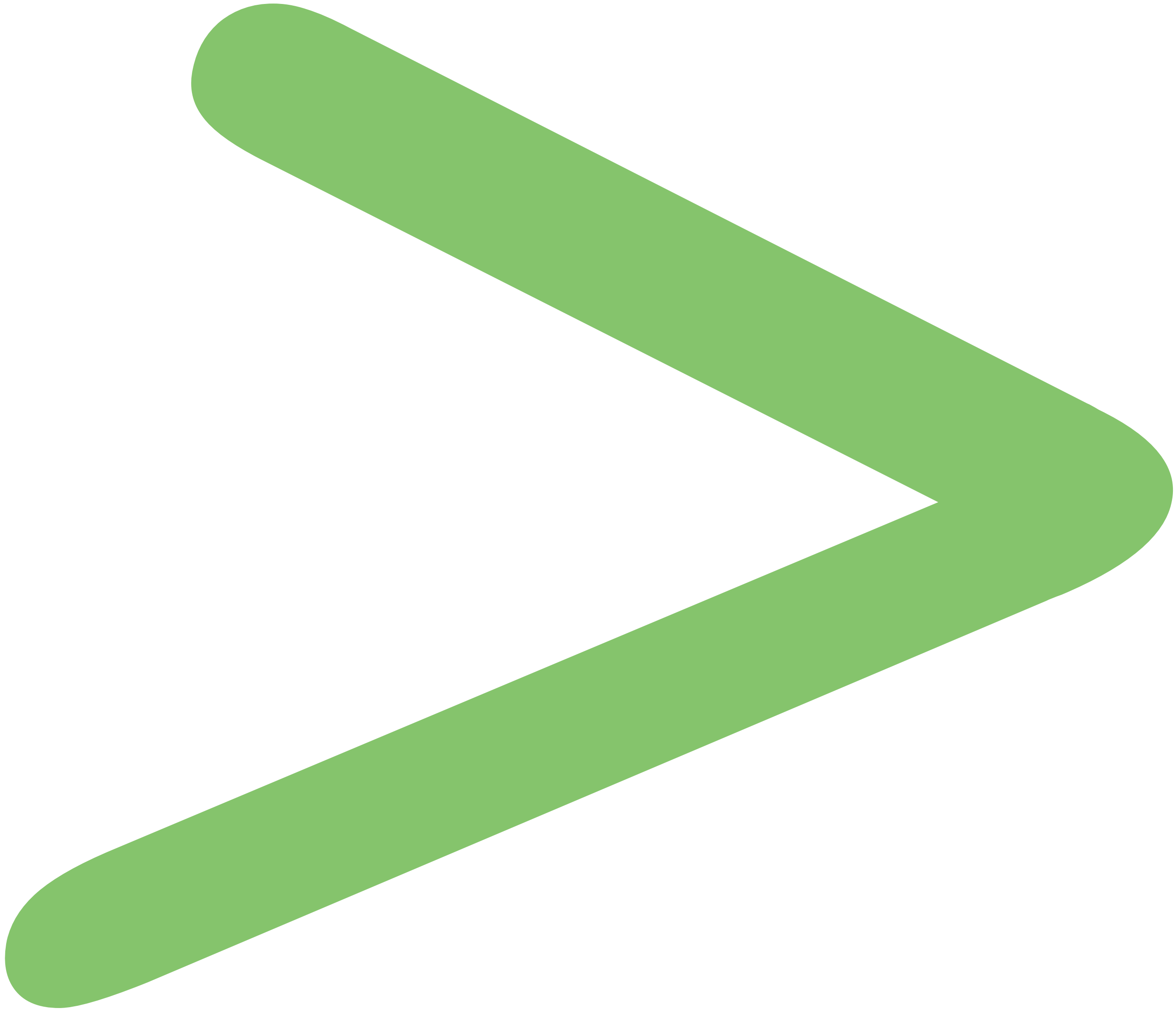
h

a



B

a



S

n

O

t

p

a

r

t

f

t

h

e

e

n

d

e

r

r

e

C

u

t

H

a

S

C

O

m

p

O

n

e

n

t

B

a

r

(

)

S

h

O

u

1

d

)

B

e

F

a

1

S

e

i

S

u

b

S

t

u

e

S

p

r

O

f

t

h

n

d

r

t

r

e

S

u

b

S

t

i

u

t

e

>

)

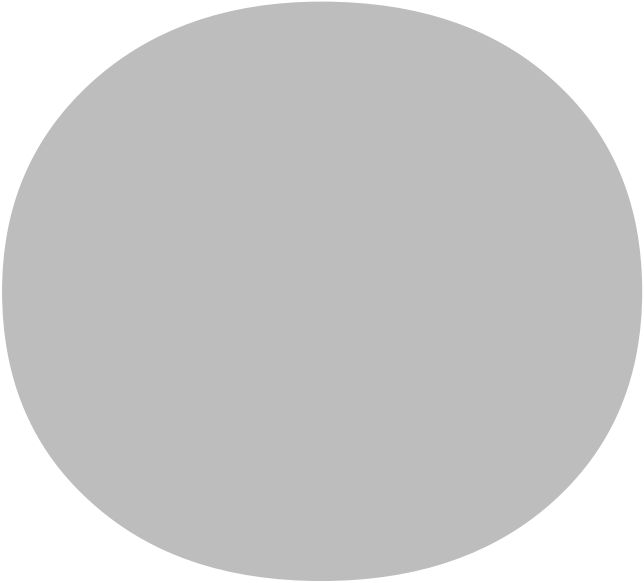
S

h

O

u

d



B

e

T

r

u

e

i

}

}

Create test doubles

Create a replacement

```
@code {
    [Fact]
    public void ShouldHaveBarSubstituteInsteadOfBar()
    {
        // Tell bUnit to replace <Bar /> with BarSubstitute
        ComponentFactories.Add<Bar, BarSubstitute>();

        // Render our component
        IRenderedFragment cut = Render(@<Foo></Foo>);

        // Assert that <Bar /> is not part of the render tree
        cut.HasComponent<Bar>().Should().BeFalse();
        // Assert that <BarSubstitute /> is part of the render tree
        cut.HasComponent<BarSubstitute>().Should().BeTrue();
    }
}
```

```
<Foo>
    <BarSubstitute />
</Foo>
```

- With ComponentFactories we can tell bUnit to replace <Bar> with a component of our choice
- Making setup easier or certain behaviour easier to assert

Conclusion