

#WWDC18

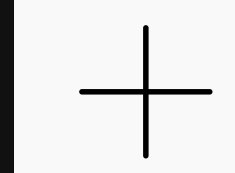
# Embracing Algorithms

Dave Abrahams

# Meet Crusty

Mind your own business


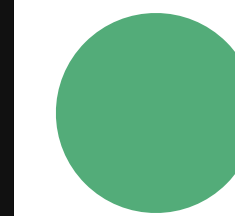
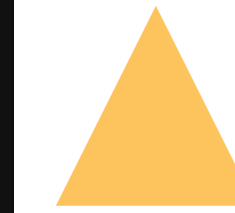
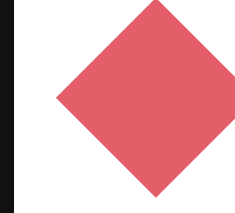
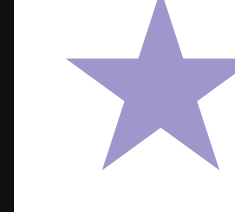
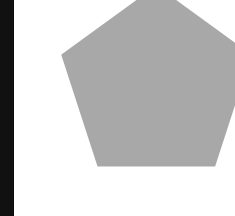






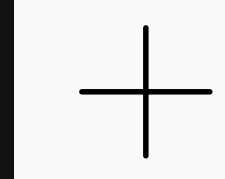
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Triangle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout





Bring to Front

Send to Back

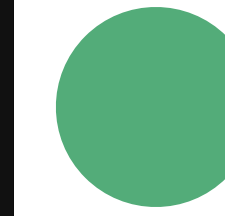
Shapes

Bring Forward

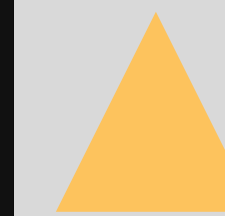
Send Backward



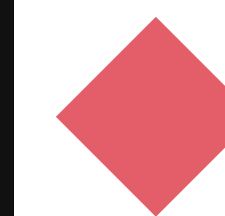
Square



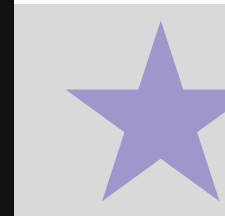
Circle



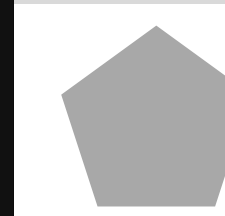
Triangle



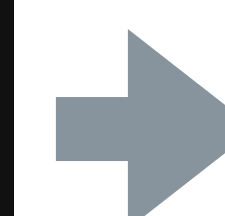
Diamond



Star



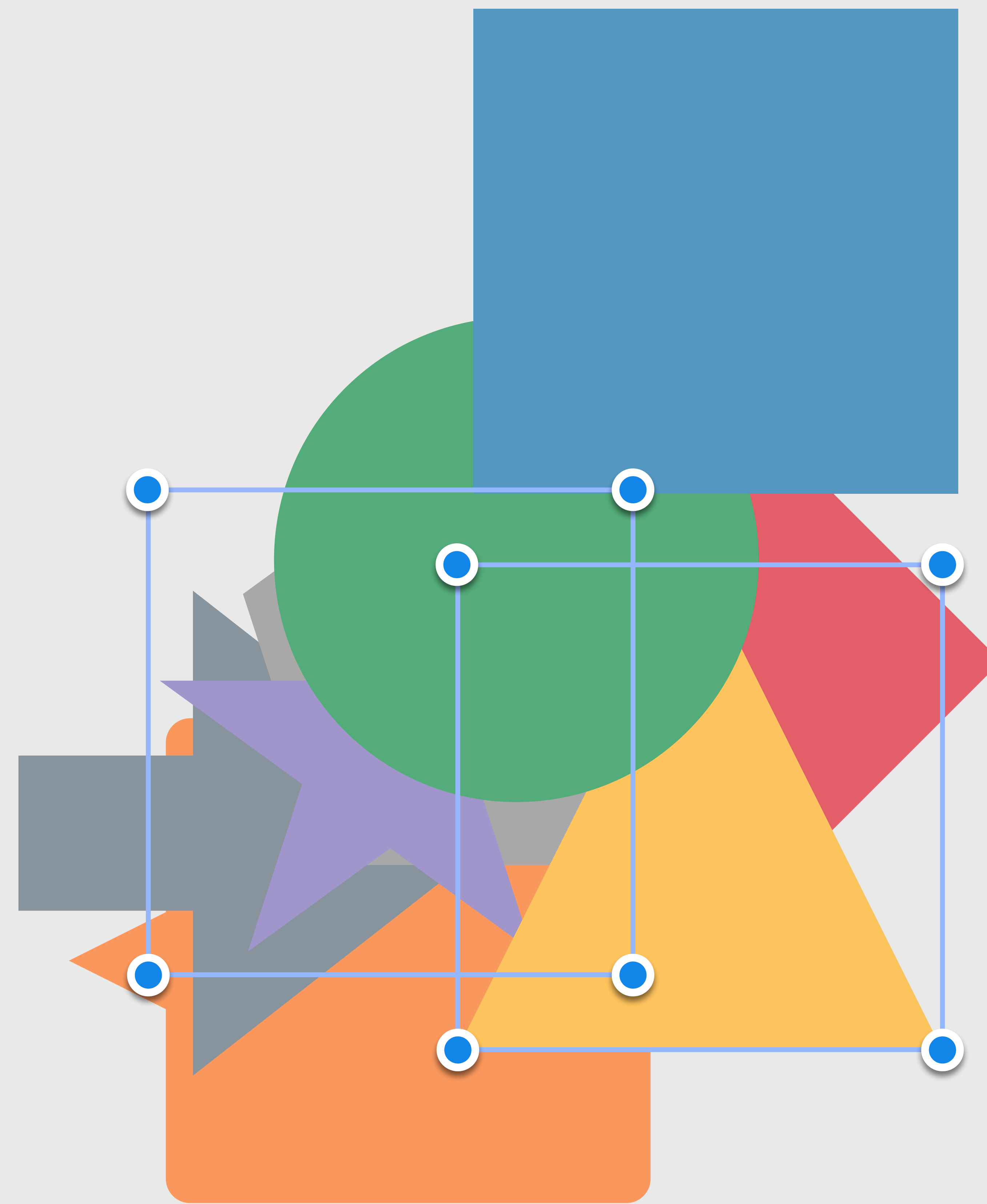
Polygon

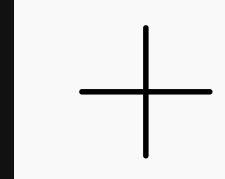


Arrow



Callout






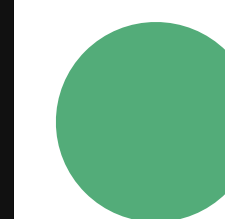
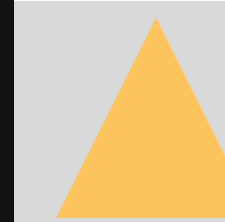
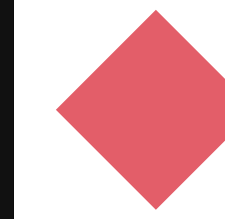
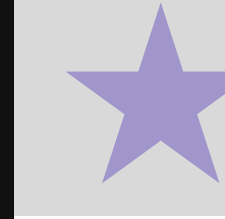
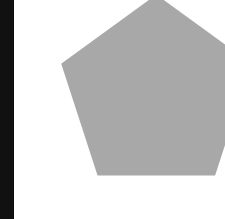
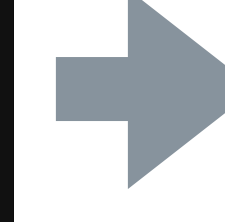

Bring to Front

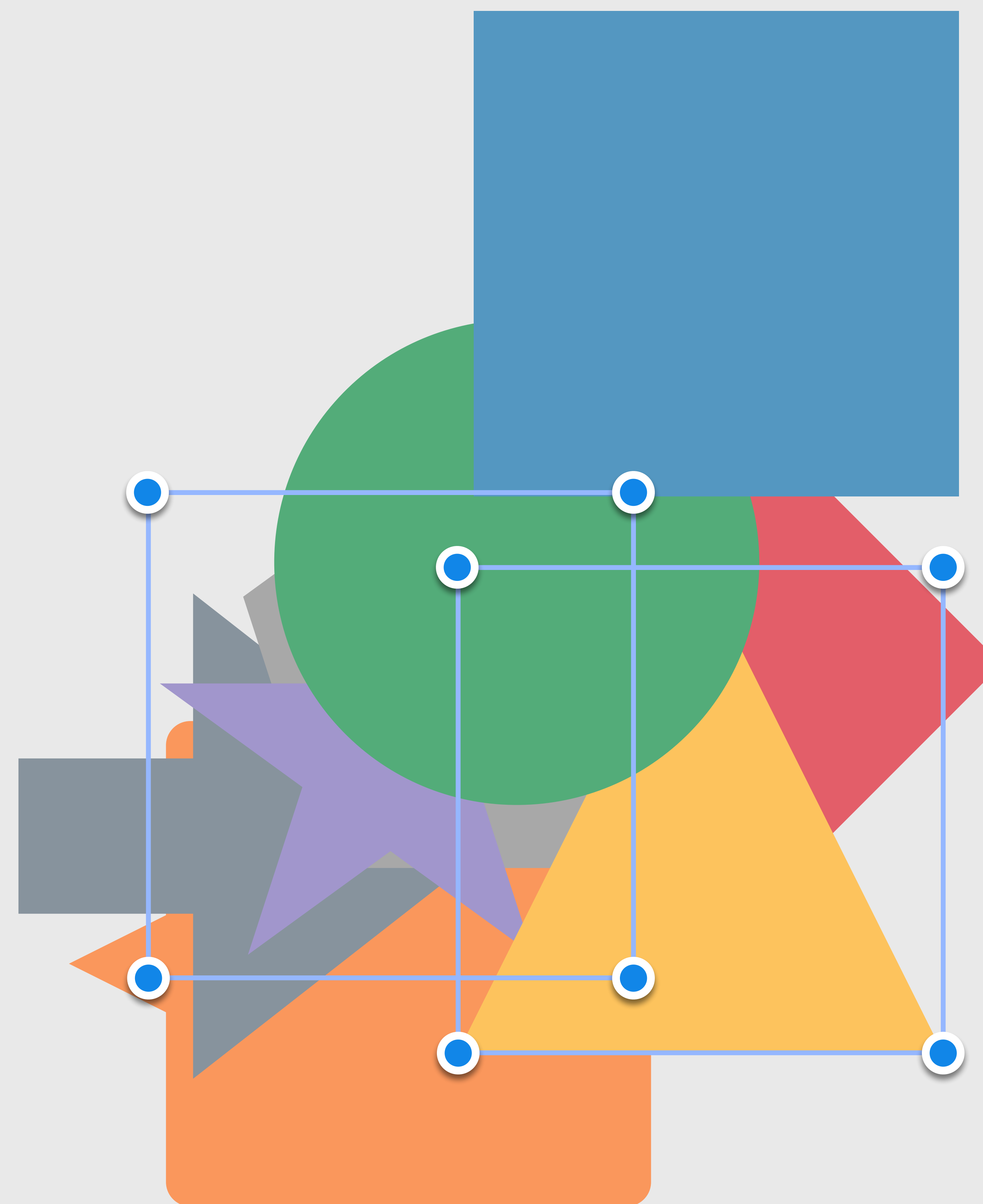
Send to Back

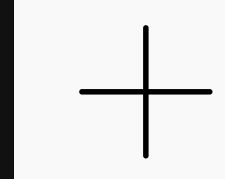
Shapes

Bring Forward

Send Backward

-  Square
-  Circle
-  Triangle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout


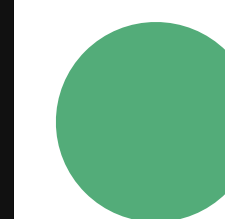
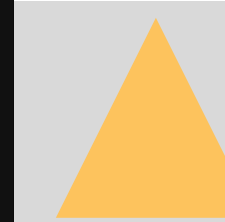
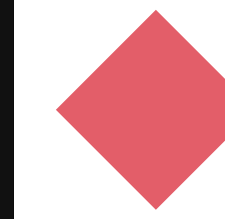
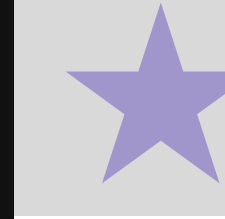
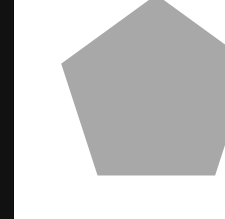
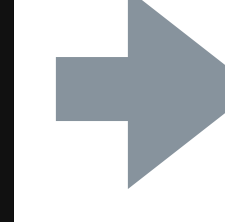



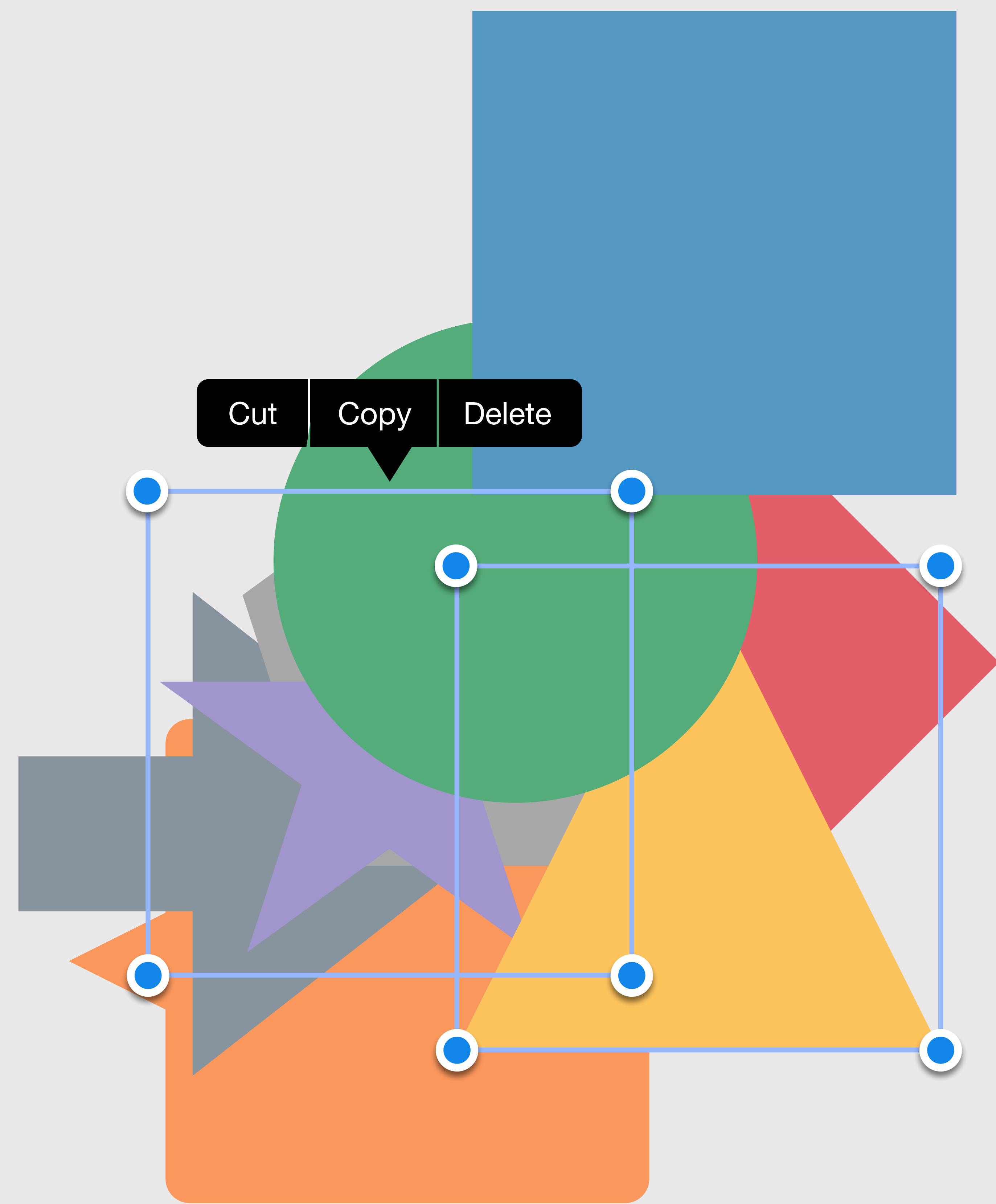


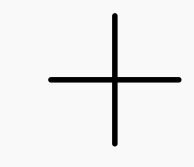
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Triangle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout





Bring to Front

Send to Back

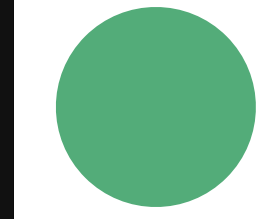
Shapes

Bring Forward

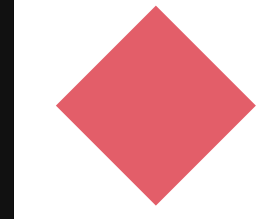
Send Backward



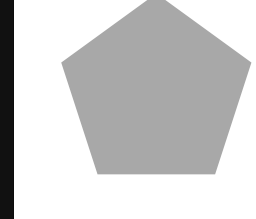
Square



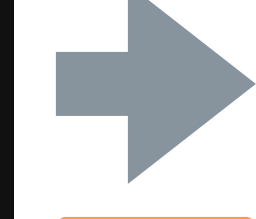
Circle



Diamond



Polygon



Arrow



Callout



```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in 0..  
      shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```





```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in 0..  
      shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in 0..  
      shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in 0..  
      shapes.count {  
        if shapes[i].isSelected {  
          shapes.remove(at: i)  
        }  
      }  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in 0..  
      shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func deleteSelection() {  
    var i = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
      i += 1  
    }  
  }  
}
```



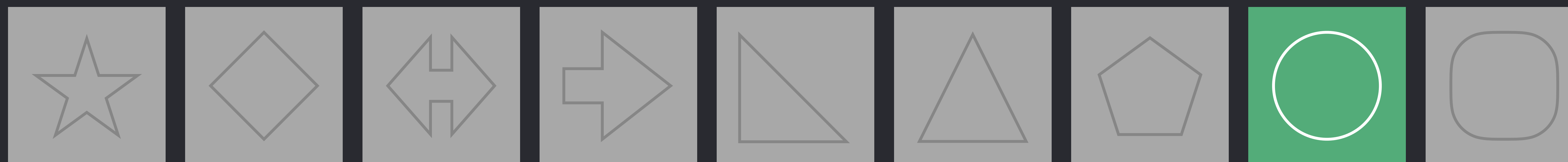
```
extension Canvas {  
  mutating func deleteSelection() {  
    var i = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
      i += 1  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func deleteSelection() {  
    var i = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
      i += 1  
    }  
  }  
}
```

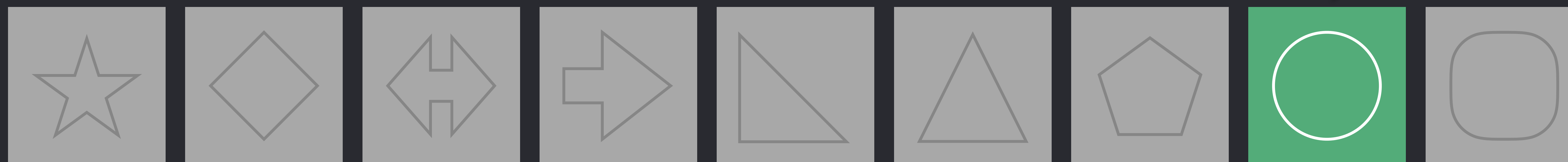


```
extension Canvas {  
  mutating func deleteSelection() {  
    var i = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
      i += 1  
    }  
  }  
}
```

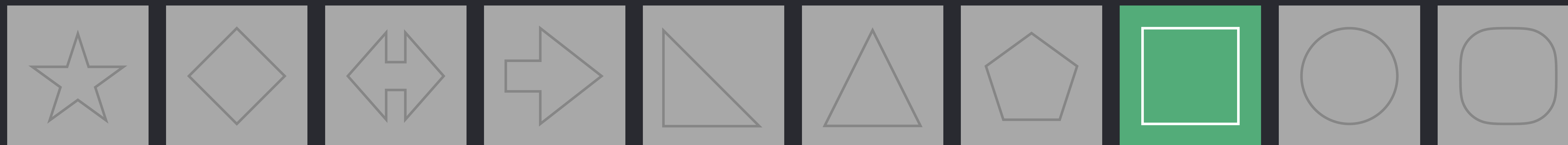




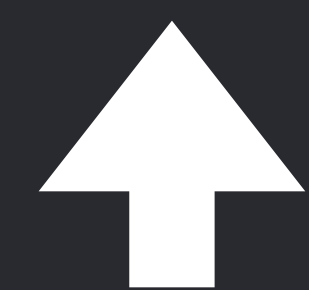
```
extension Canvas {  
  mutating func deleteSelection() {  
    var i = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
      i += 1  
    }  
  }  
}
```



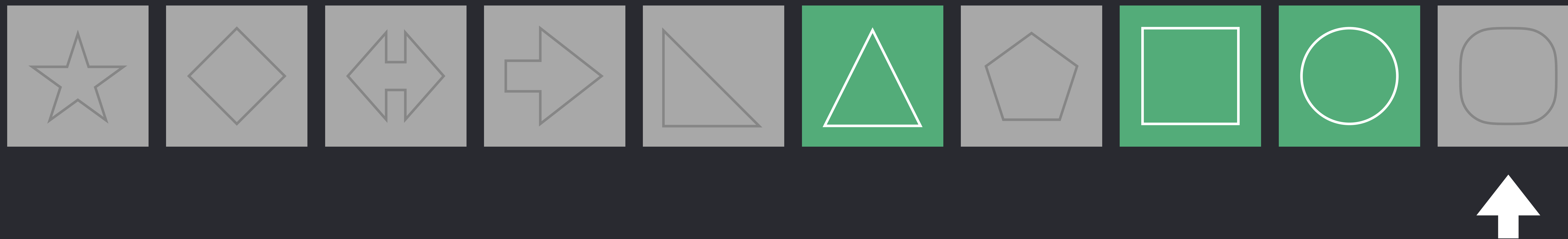
```
extension Canvas {  
  mutating func deleteSelection() {  
    var i = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
      else {  
        i += 1  
      }  
    }  
  }  
}
```



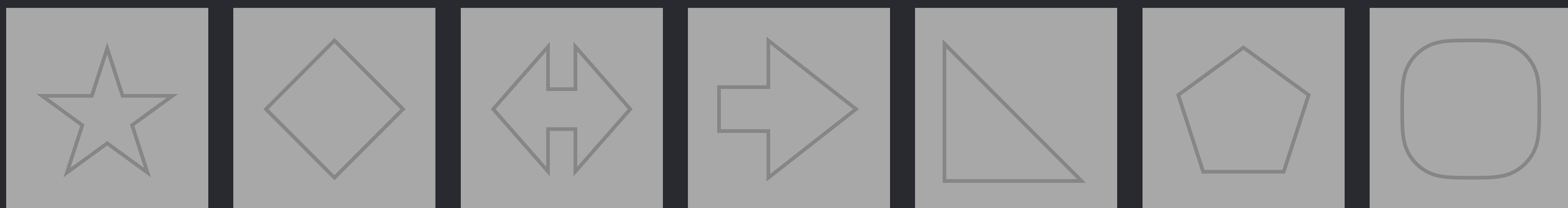
```
extension Canvas {  
  mutating func deleteSelection() {  
    var i = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
      else {  
        i += 1  
      }  
    }  
  }  
}
```



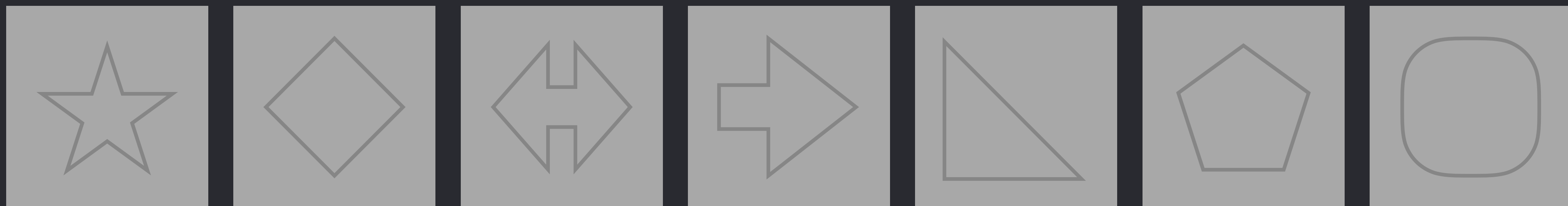
```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in (0..  
      shapes.count).reversed() {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```



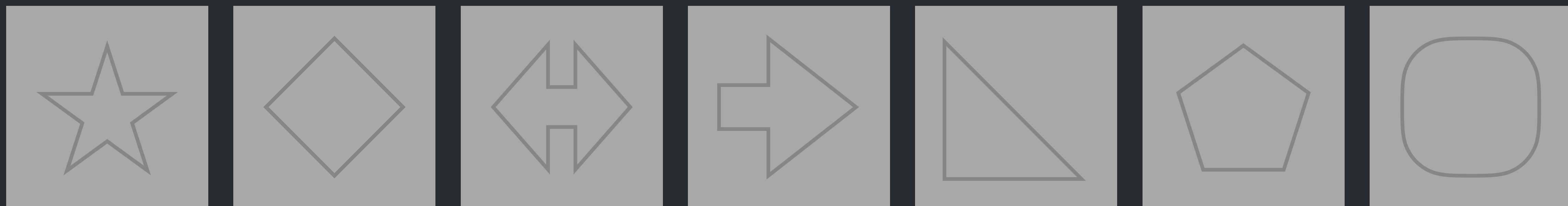
```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in (0..  
shapes.count).reversed() {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in (0..  
shapes.count).reversed() {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in (0..  
      shapes.count).reversed() {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```



```
extension Canvas {
  mutating func deleteSelection() {
    for i in (0..
```

#### Summary

Removes and returns the element at the specified position.

#### Declaration

```
mutating func remove(at index: Int) -> Shape
```

#### Discussion

All the elements following the specified position are moved up to close the gap.

```
var measurements: [Double] = [1.1, 1.5, 2.9, 1.2, 1.5, 1.3, 1.2]
let removed = measurements.remove(at: 2)
print(measurements)
// Prints "[1.1, 1.5, 1.2, 1.5, 1.3, 1.2]"
```

#### Complexity

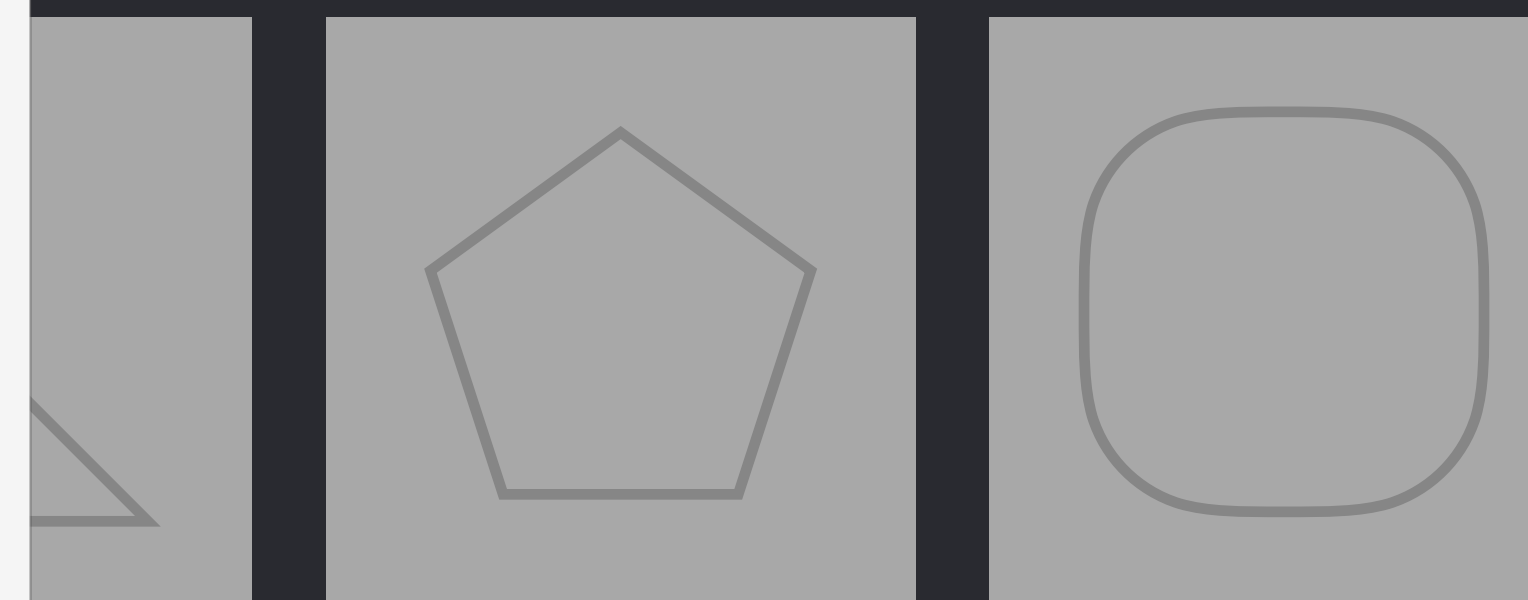
$O(n)$ , where  $n$  is the length of the array.

#### Parameters

**index** The position of the element to remove. `index` must be a valid index of the array.

#### Returns

The element at the specified index.





```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in (0..  
      shapes.count).reversed() {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```

#### Summary

Removes and returns the element at the specified position.

#### Declaration

```
mutating func remove(at index: Int) -> Shape
```

#### Discussion

All the elements following the specified position are moved up to close the gap.

```
var measurements: [Double] = [1.1, 1.5, 2.9, 1.2, 1.5, 1.3, 1.2]  
let removed = measurements.remove(at: 2)  
print(measurements)  
// Prints "[1.1, 1.5, 1.2, 1.5, 1.3, 1.2]"
```

#### Complexity

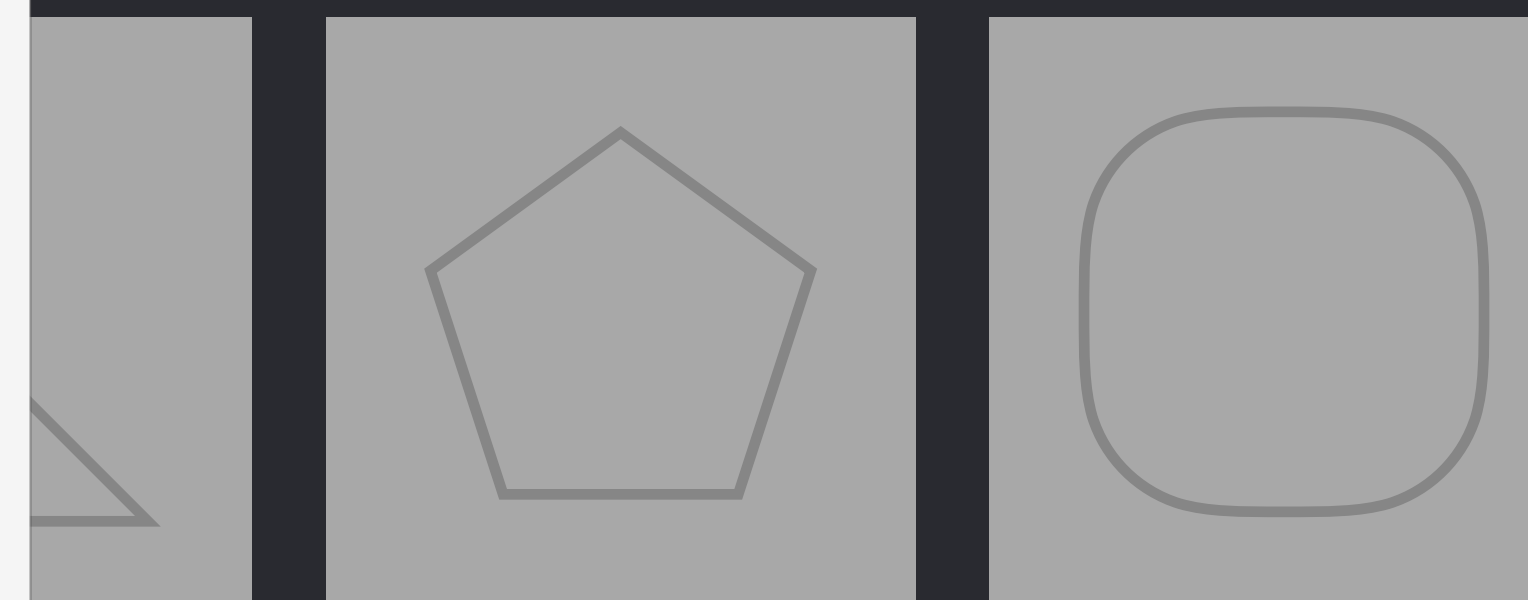
$O(n)$ , where  $n$  is the length of the array.

#### Parameters

**index** The position of the element to remove. `index` must be a valid index of the array.

#### Returns

The element at the specified index.



```
extension Canvas {
  mutating func deleteSelection() {
    for i in (0..
```

#### Summary

Removes and returns the element at the specified position.

#### Declaration

```
mutating func remove(at index: Int) -> Shape
```

#### Discussion

All the elements following the specified position are moved up to close the gap.

```
var measurements: [Double] = [1.1, 1.5, 2.9, 1.2, 1.5, 1.3, 1.2]
let removed = measurements.remove(at: 2)
print(measurements)
// Prints "[1.1, 1.5, 1.2, 1.5, 1.3, 1.2]"
```

#### Complexity

$O(n)$ , where  $n$  is the length of the array.

#### Parameters

`index` The position of the element to remove. `index` must be a valid index of the array.

#### Returns

The element at the specified index.

## Summary

Removes and returns the element at the specified position.

## Declaration

```
mutating func remove(at index: Int) -> Shape
```

## Discussion

All the elements following the specified position are moved up to close the gap.

```
var measurements: [Double] = [1.1, 1.5, 2.9, 1.2, 1.5, 1.3, 1.2]
let removed = measurements.remove(at: 2)
print(measurements)
// Prints "[1.1, 1.5, 1.2, 1.5, 1.3, 1.2]"
```

## Complexity

$O(n)$ , where  $n$  is the length of the array.

## Parameters

`index` The position of the element to remove. `index` must be a valid index of the array.

## Returns

The element at the specified index.

## Summary

Removes and returns the element at the specified position.

## Declaration

```
mutating func remove(at index: Int) -> Shape
```

## Discussion

All the elements following the specified position are moved up to close the gap.

```
var measurements: [Double] = [1.1, 1.5, 2.9, 1.2, 1.5, 1.3, 1.2]
let removed = measurements.remove(at: 2)
print(measurements)
// Prints "[1.1, 1.5, 1.2, 1.5, 1.3, 1.2]"
```

## Complexity

$O(n)$ , where  $n$  is the length of the array.

## Parameters

`index` The position of the element to remove. `index` must be a valid index of the array.

## Returns

The element at the specified index.

## Summary

Removes and returns the element at the specified position.

## Declaration

```
mutating func remove(at index: Int) -> Shape
```

## Discussion

All the elements following the specified position are moved up to close the gap.

```
var measurements: [Double] = [1.1, 1.5, 2.9, 1.2, 1.5, 1.3, 1.2]
let removed = measurements.remove(at: 2)
print(measurements)
// Prints "[1.1, 1.5, 1.2, 1.5, 1.3, 1.2]"
```

## Complexity

$O(n)$ , where  $n$  is the length of the array.

## Parameters

`index` The position of the element to remove. `index` must be a valid index of the array.

## Returns

The element at the specified index.

## Summary

Removes and returns the element at the specified position.

## Declaration

```
mutating func remove(at index: Int) -> Shape
```

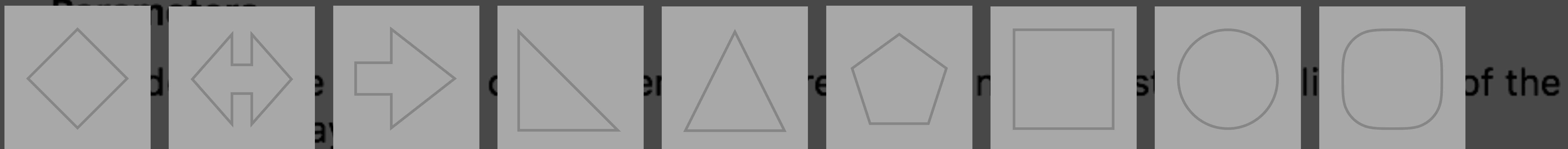
## Discussion

All the elements following the specified position are moved up to close the gap.

```
var measurements: [Double] = [1.1, 1.5, 2.9, 1.2, 1.5, 1.3, 1.2]
let removed = measurements.remove(at: 2)
print(measurements)
// Prints "[1.1, 1.5, 1.2, 1.5, 1.3, 1.2]"
```

## Complexity

$O(n)$ , where  $n$  is the length of the array.



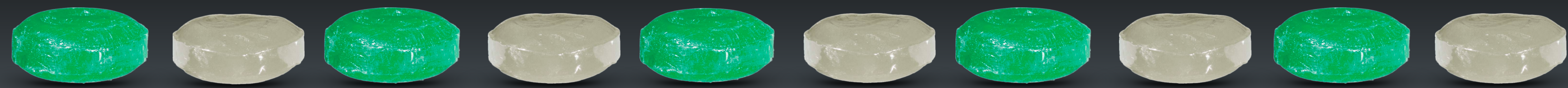
 Returns

The element at the specified index.

**Delete the Green Ones**



# Delete the Green Ones

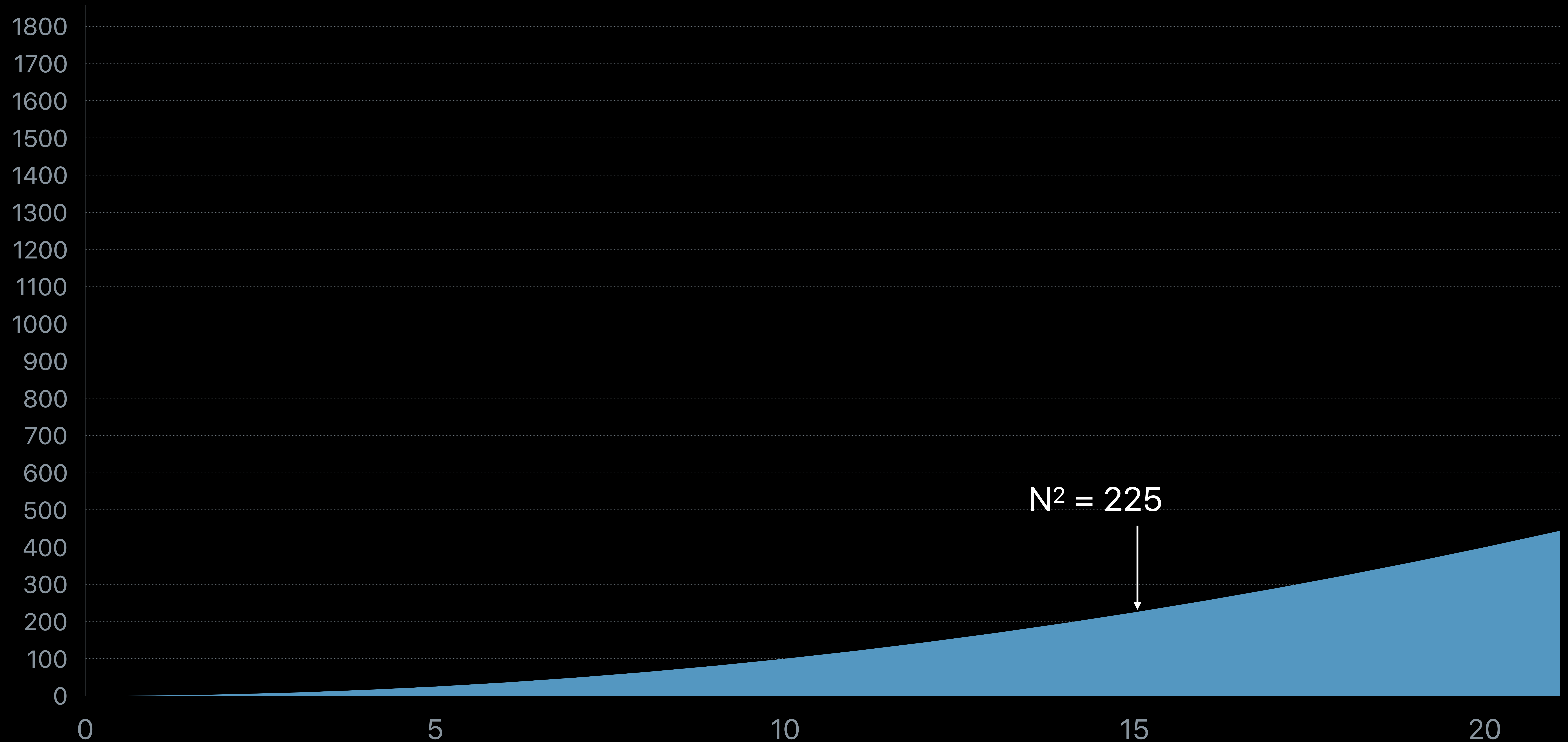




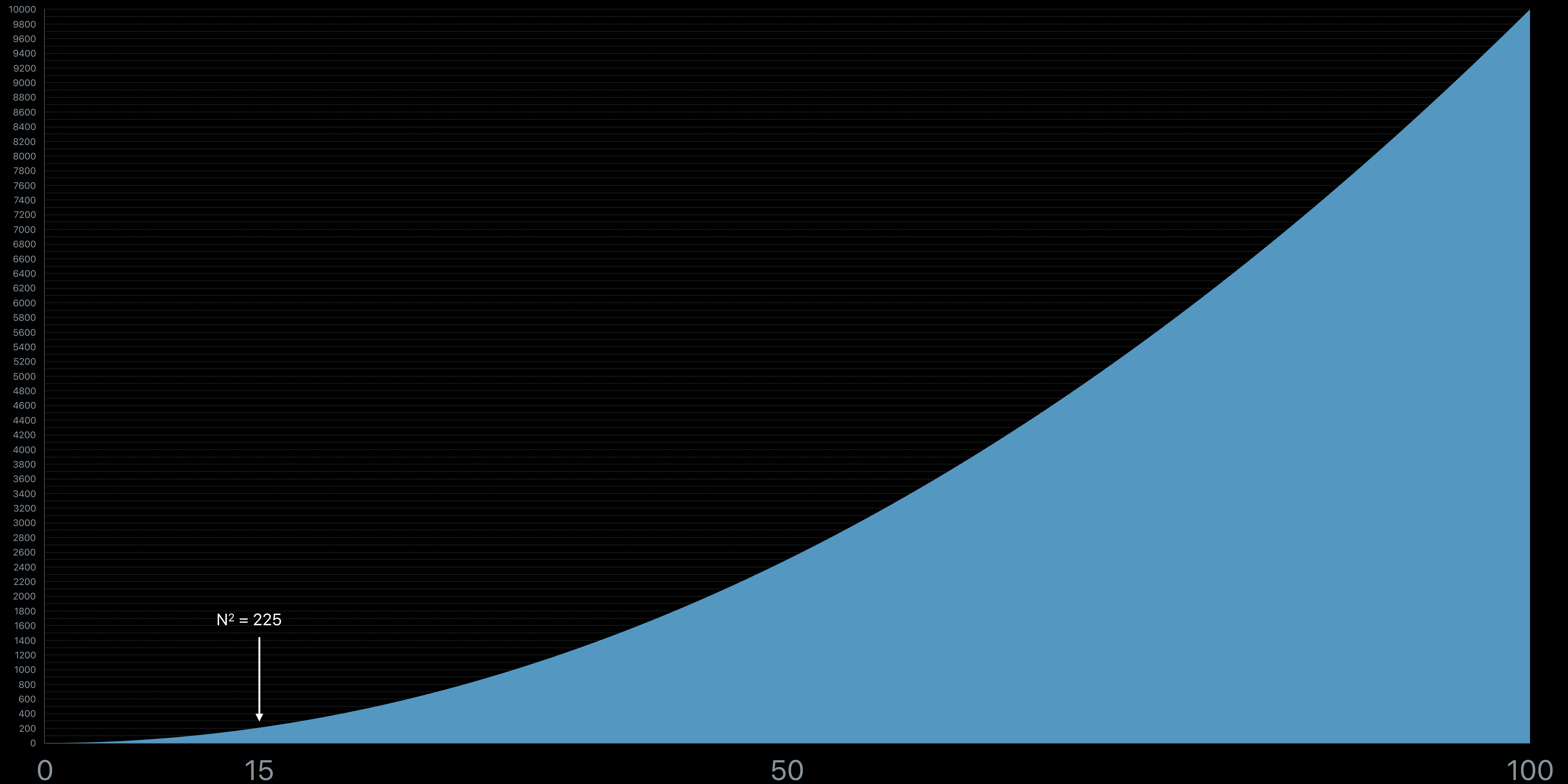
# Delete the Green Ones



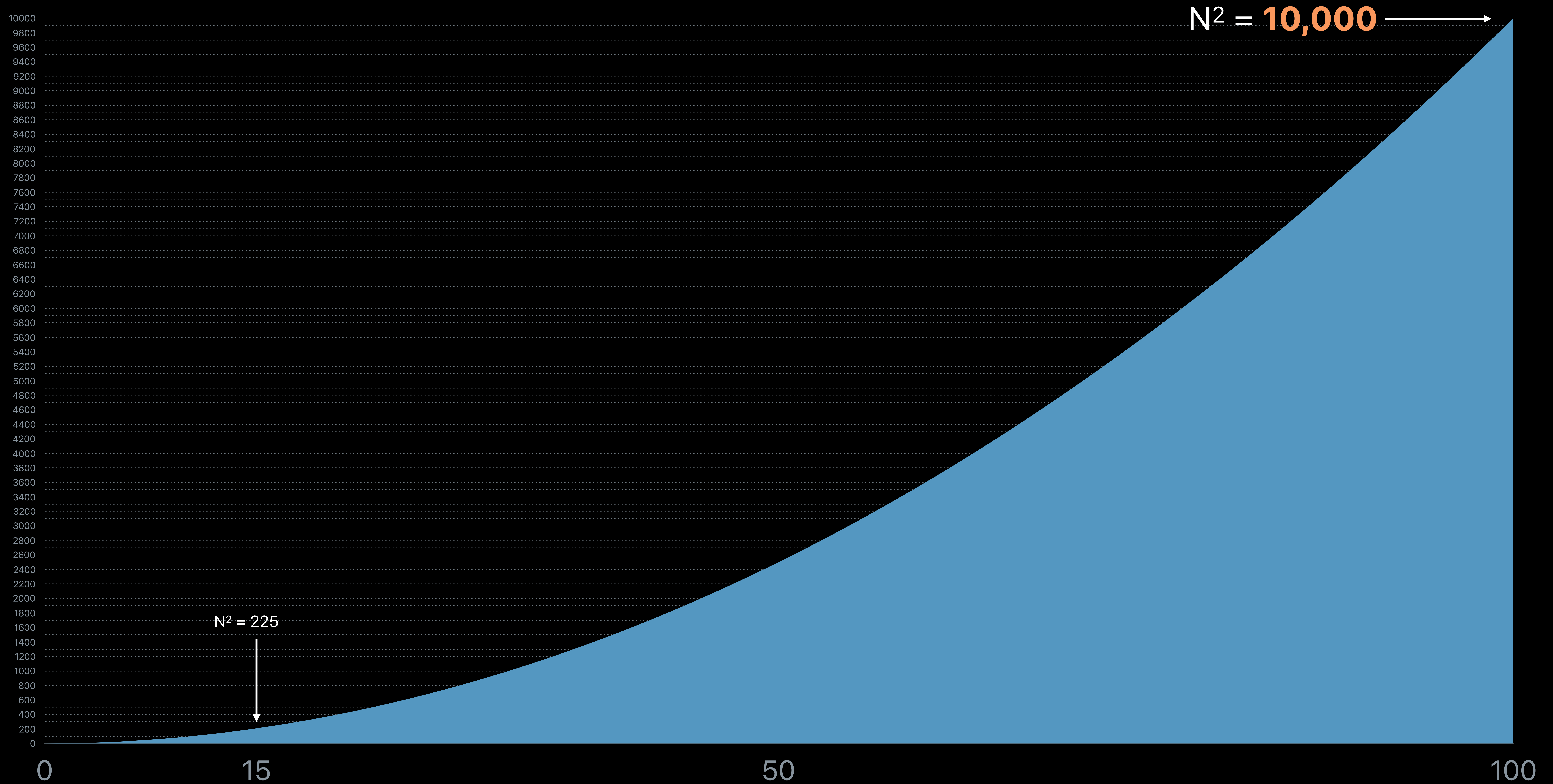
# Scalability Matters



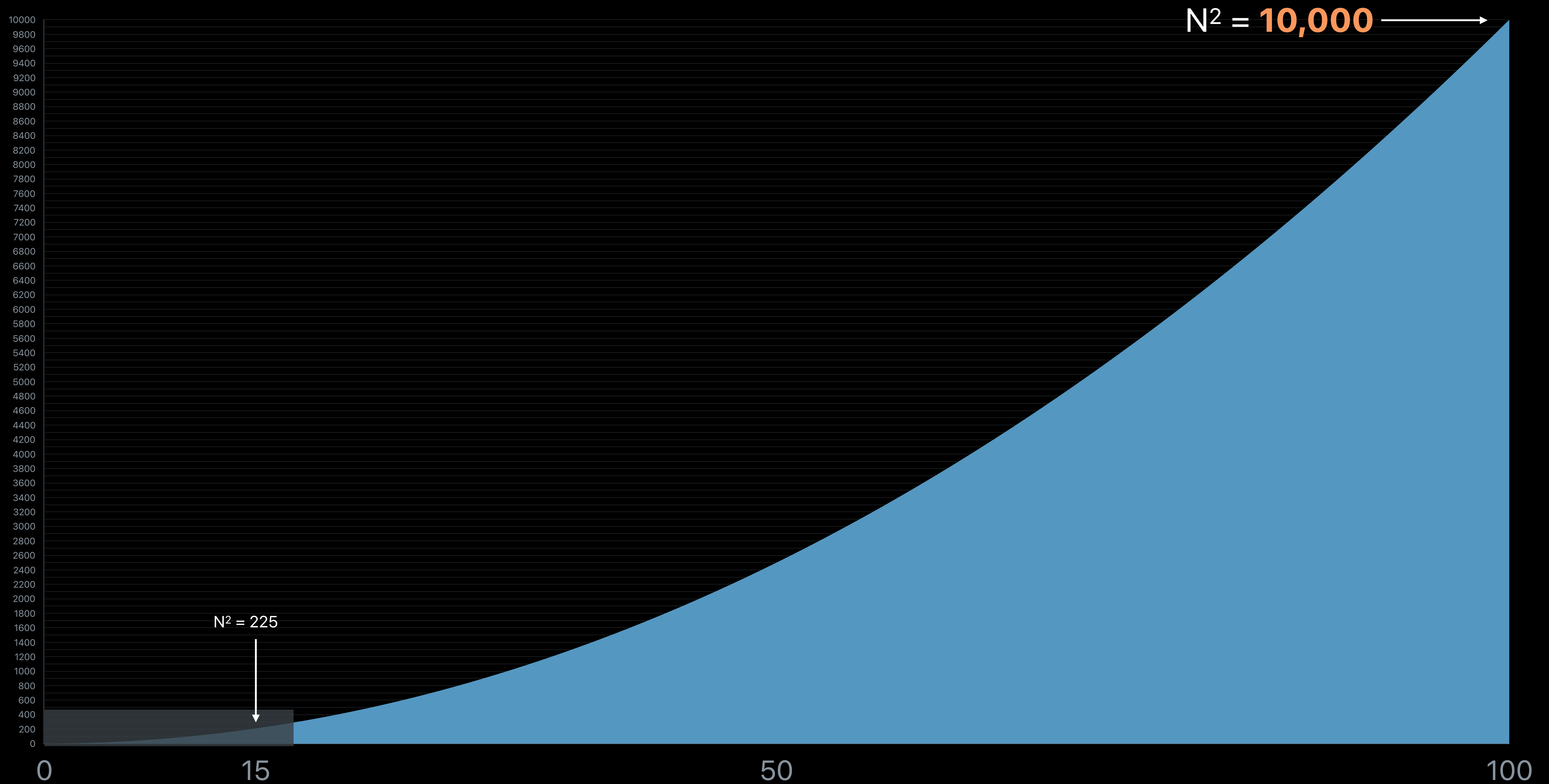
# Scalability Matters



# Scalability Matters

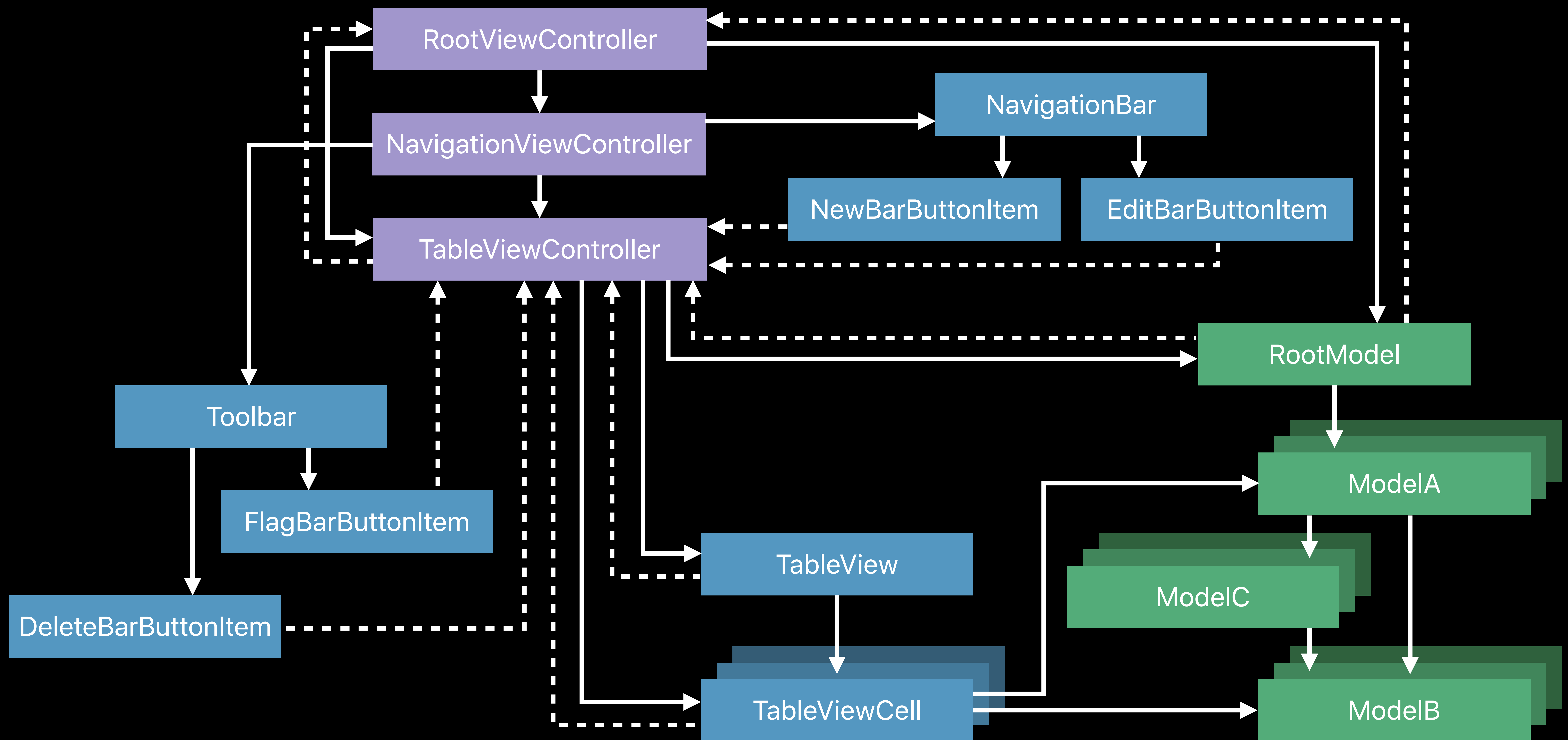


# Scalability Matters



# The Application Architect Speaks

# The Application Architect Speaks



# Algorithm: Look it Up



# Algorithm: Look it Up



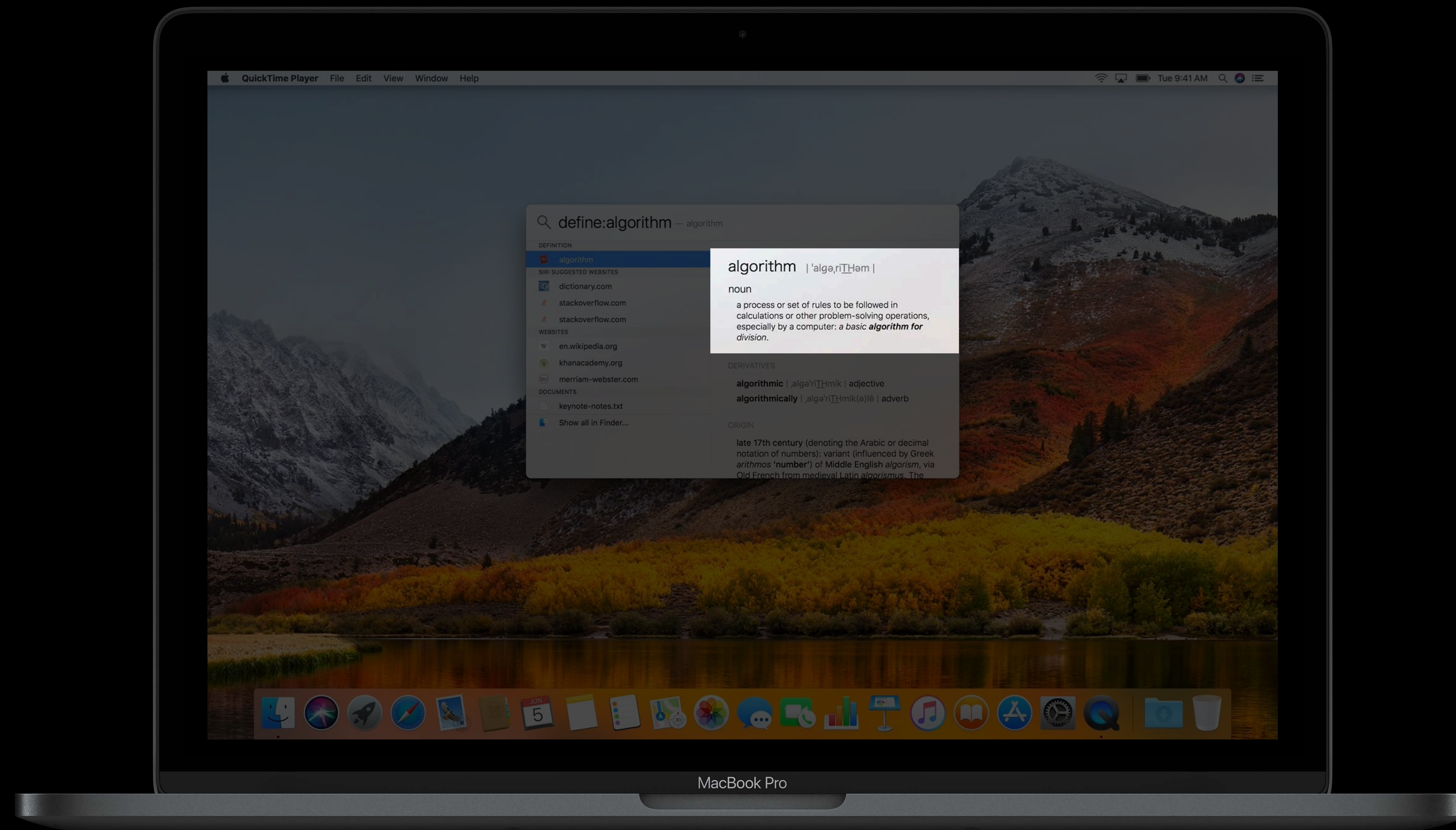
# Algorithm: Look it Up



# Algorithm: Look it Up



# Algorithm: Look it Up



define:algorithm — algorithm

DEFINITION

algorithm |'algə,rɪˈθɒm|

noun

a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer: a *basic algorithm for division*.

DERIVATIVES

algorithmic |algə'rɪθmɪk| adjective

algorithmically |algə'rɪθmɪk(ə)lɪ| adverb

ORIGIN

late 17th century (denoting the Arabic or decimal notation of numbers): variant (influenced by Greek *arithmos* 'number') of Middle English *algorism*, via Old French from medieval Latin *algorismus*. The

# Algorithm: Look it Up



algorithm | 'algə,rɪTHəm |

noun

a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer: a *basic algorithm for division*.

# Algorithm: Look it Up



# Algorithm: Look it Up



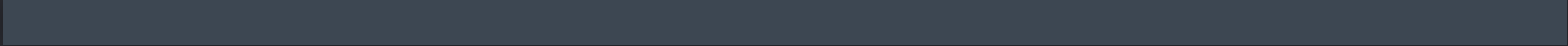
# Algorithm: Look it Up





```
extension Canvas {
  mutating func deleteSelection() {
    for i in (0..
```

```
extension Canvas {  
  mutating func deleteSelection() {  
    for i in (0..  
      shapes.count).reversed() {  
      if shapes[i].isSelected {  
        shapes.remove(at: i)  
      }  
    }  
  }  
}
```

```
extension Canvas {  
  mutating func deleteSelection() {  
      
  }  
}
```

```
extension Canvas {  
  mutating func deleteSelection() {  
    shapes.removeAll(where: { $0.isSelected })  
  }  
}
```

```
extension Canvas {  
  mutating func deleteSelection() {  
    shapes.removeAll(where: { $0.isSelected })  
  }  
}
```

```
extension Canvas {  
  mutating func deleteSelection() {  
    shapes.removeAll(where: { $0.isSelected })  
  }  
}
```

#### Summary

Removes from the collection all elements that satisfy the given predicate.

#### Declaration

```
mutating func removeAll(where predicate: (Self.Element) throws ->  
Bool) rethrows
```

#### Discussion

#### Complexity

$O(n)$ , where  $n$  is the length of the collection.

#### Parameters

**predicate** A closure that takes an element of the sequence as its argument and returns a Boolean value indicating whether the element should be removed from the collection.

```
extension Canvas {  
    mutating func deleteSelection() {  
        shapes.  
    }  
}
```

#### Summary

Removes from the collection all elements that satisfy the given predicate.

#### Declaration

```
mutating func removeAll(where predicate: (Self.Element) throws ->  
Bool) rethrows
```

#### Discussion

#### Complexity

$O(n)$ , where  $n$  is the length of the collection.

#### Parameters

`predicate` A closure that takes an element of the sequence as its argument and returns a Boolean value indicating whether the element should be removed from the collection.

## Summary

Removes from the collection all elements that satisfy the given predicate.

## Declaration

```
mutating func removeAll(where predicate: (Self.Element) throws -> Bool) rethrows
```

## Discussion

### Complexity

$O(n)$ , where  $n$  is the length of the collection.

## Parameters

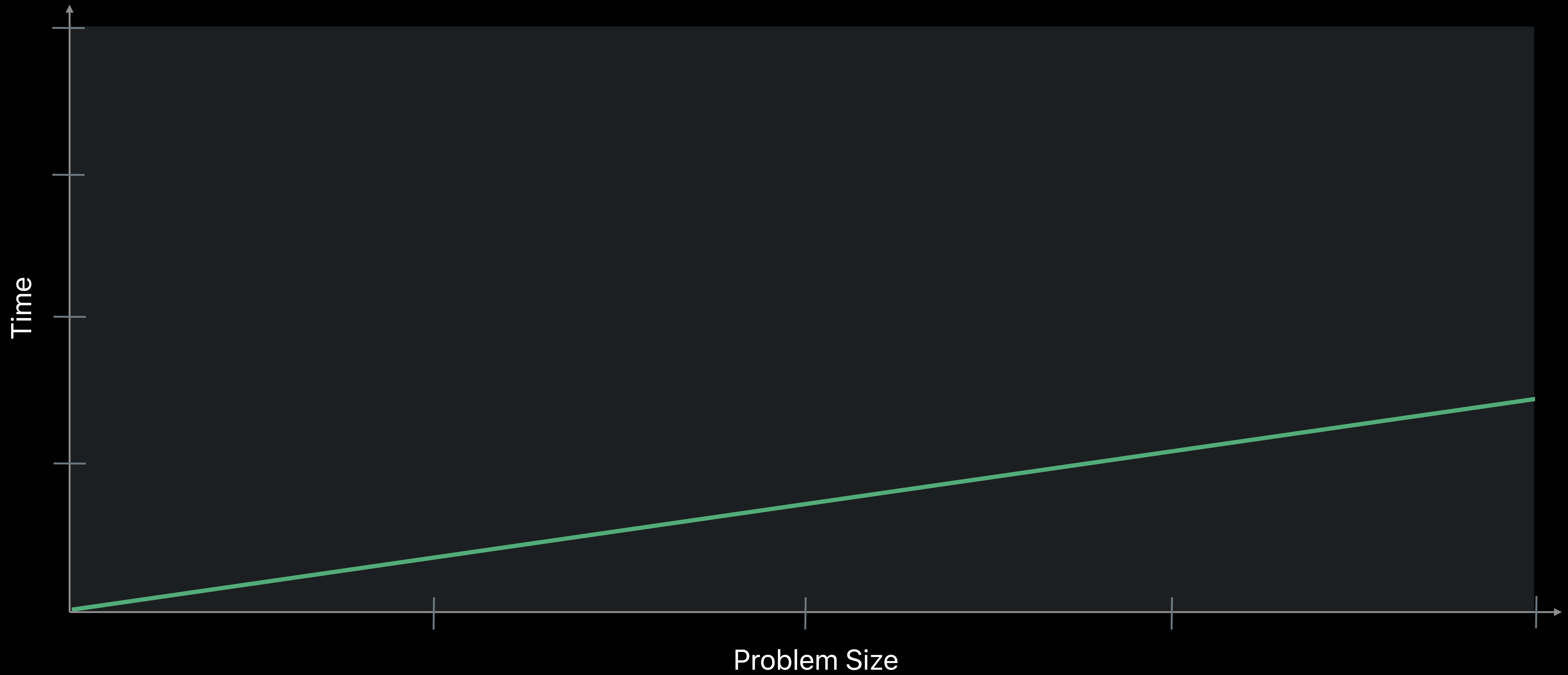
`predicate` A closure that takes an element of the sequence as its argument and returns a Boolean value indicating whether the element should be removed from the collection.



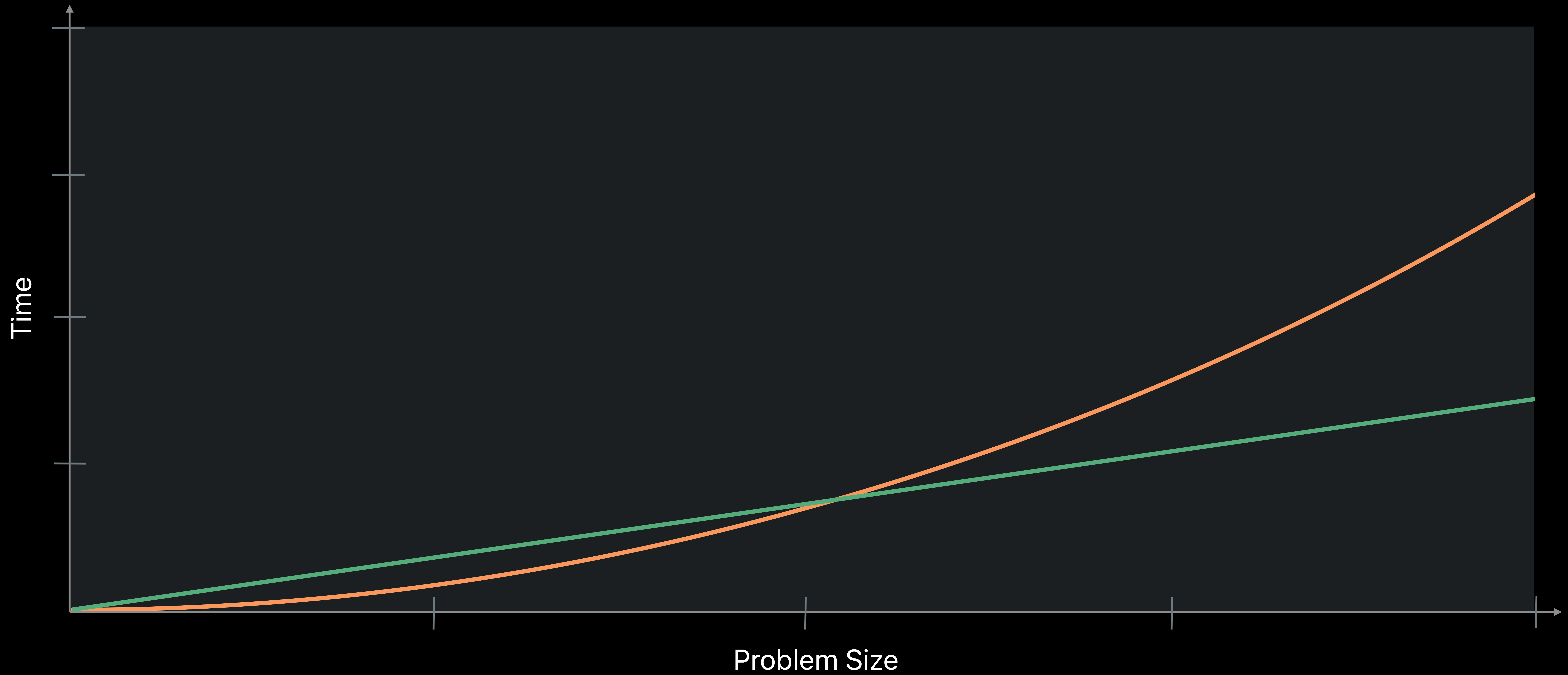
# Linear vs Quadratic



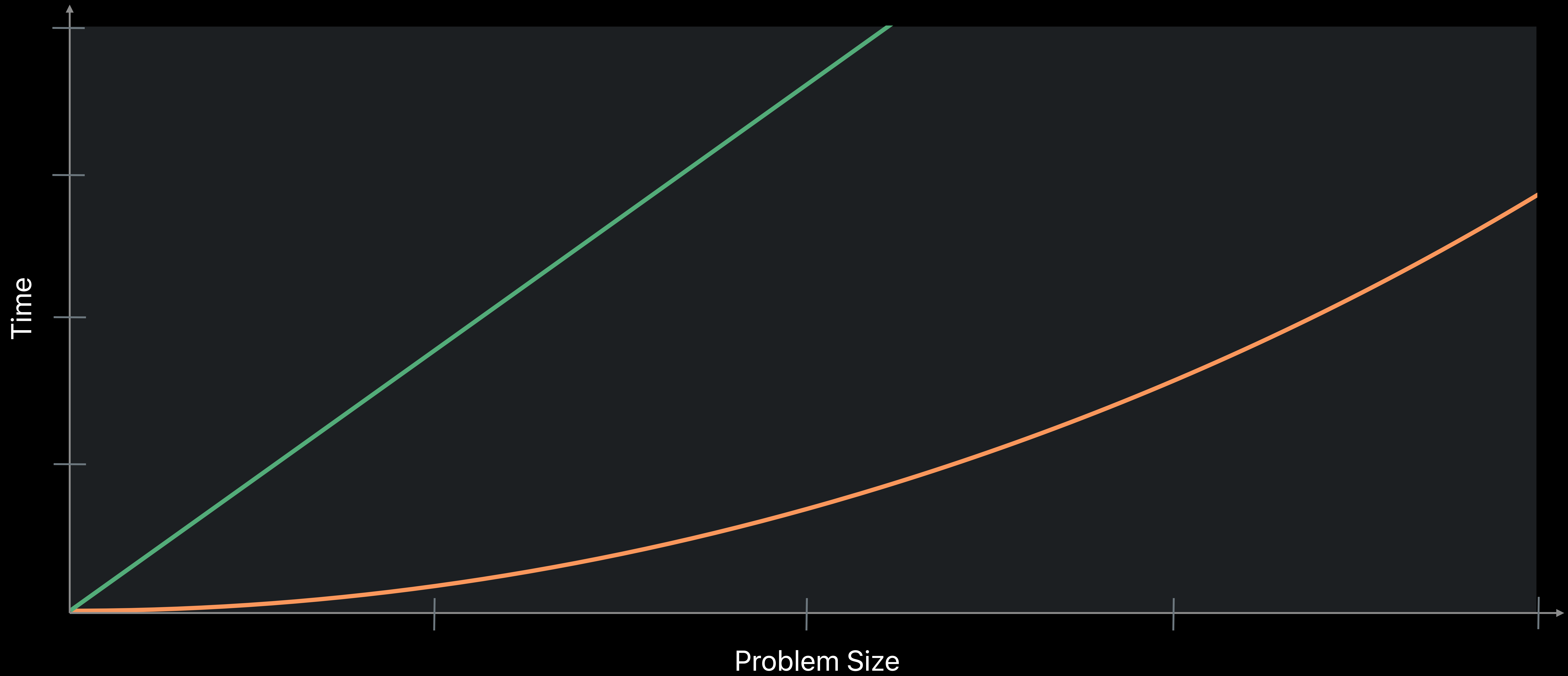
# Linear vs Quadratic



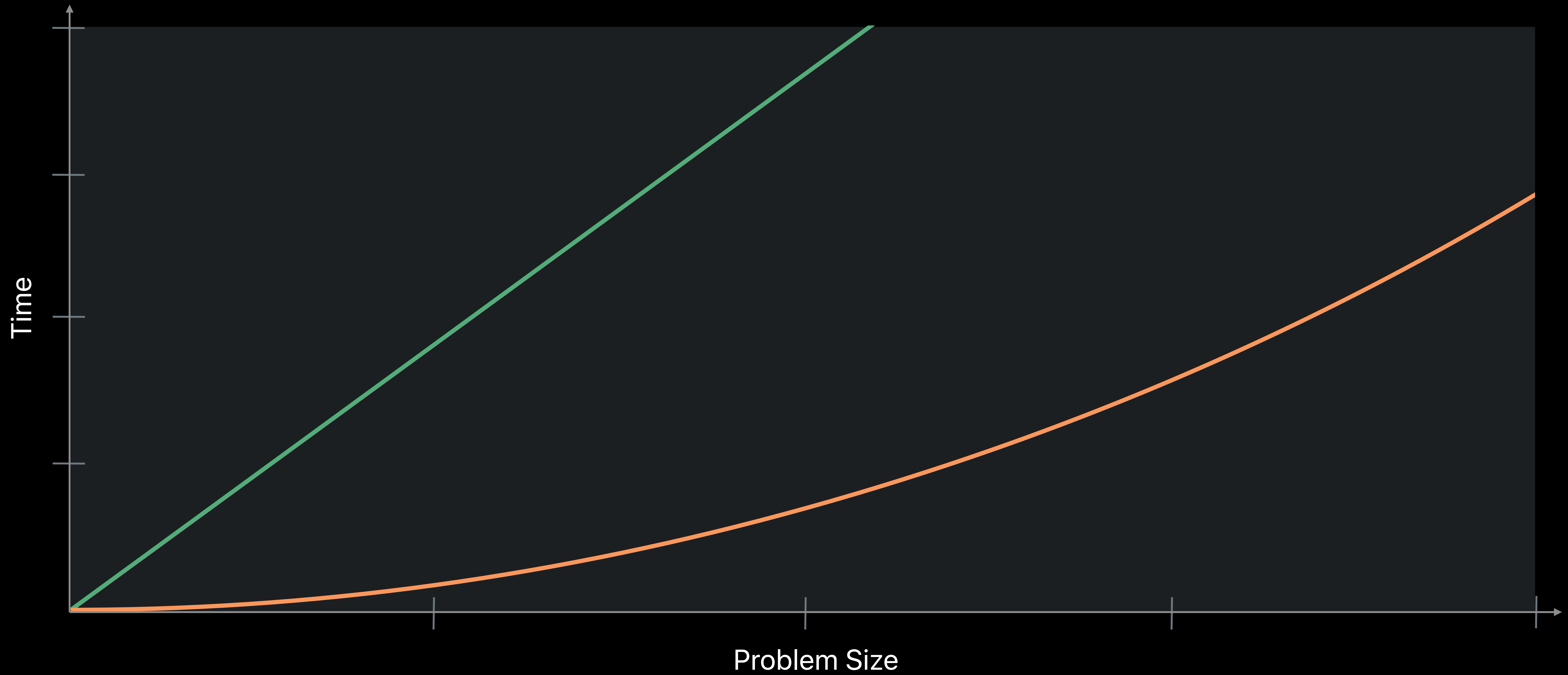
# Linear vs Quadratic



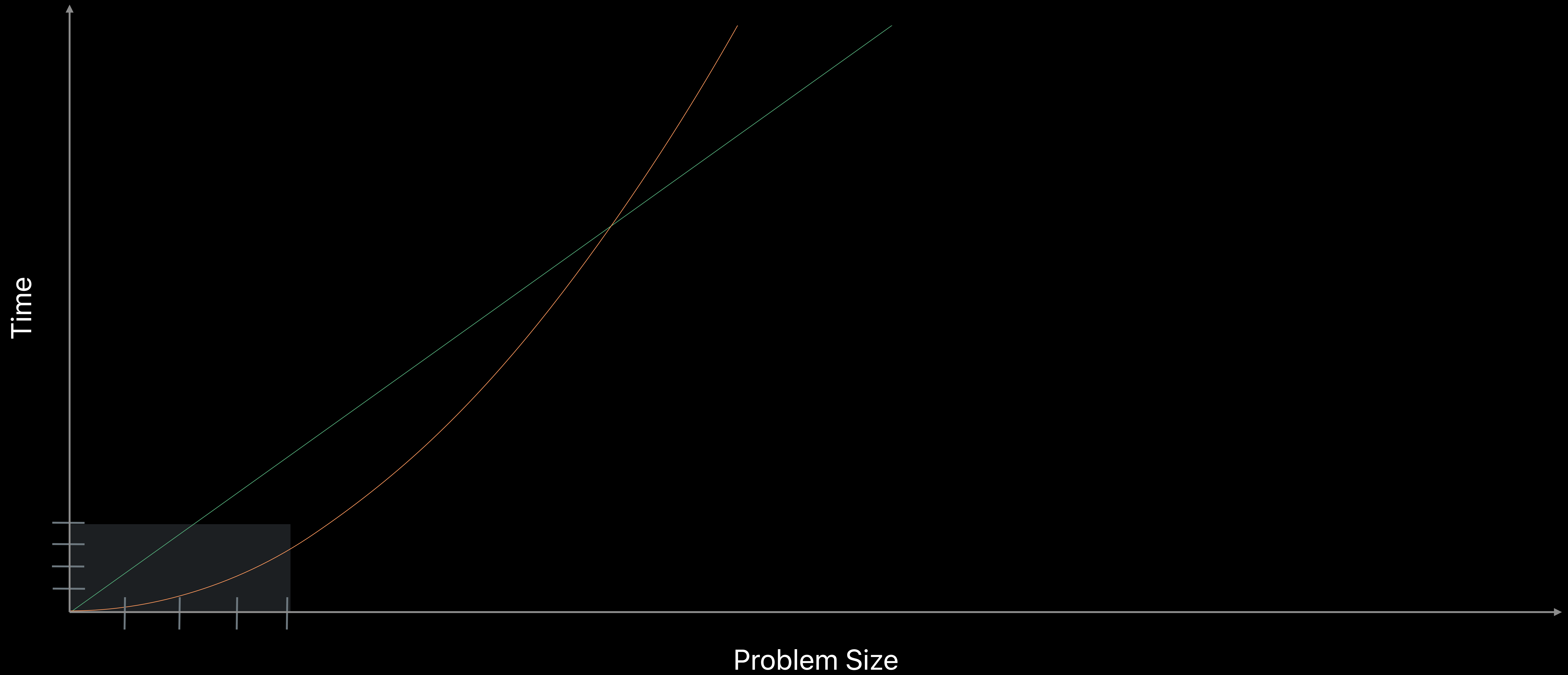
# Linear vs Quadratic



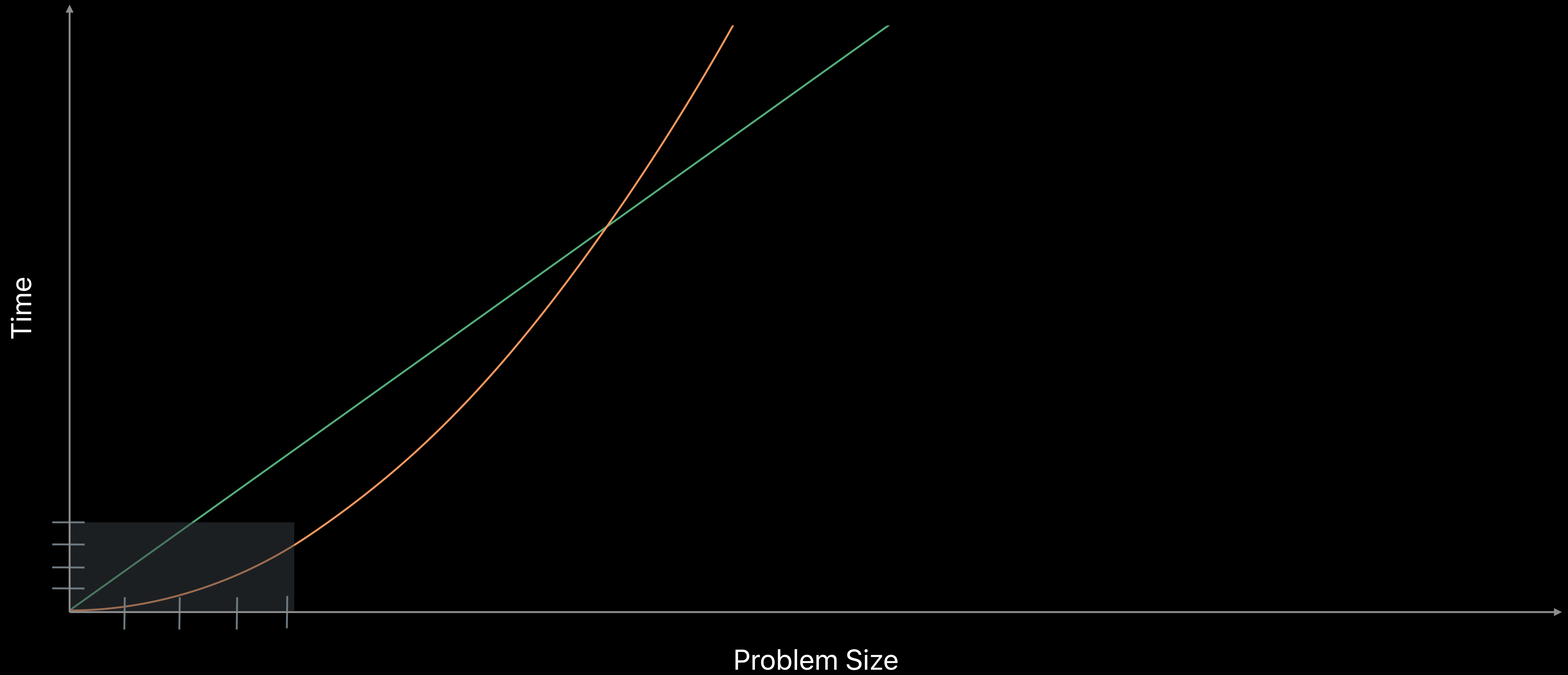
# Linear vs Quadratic



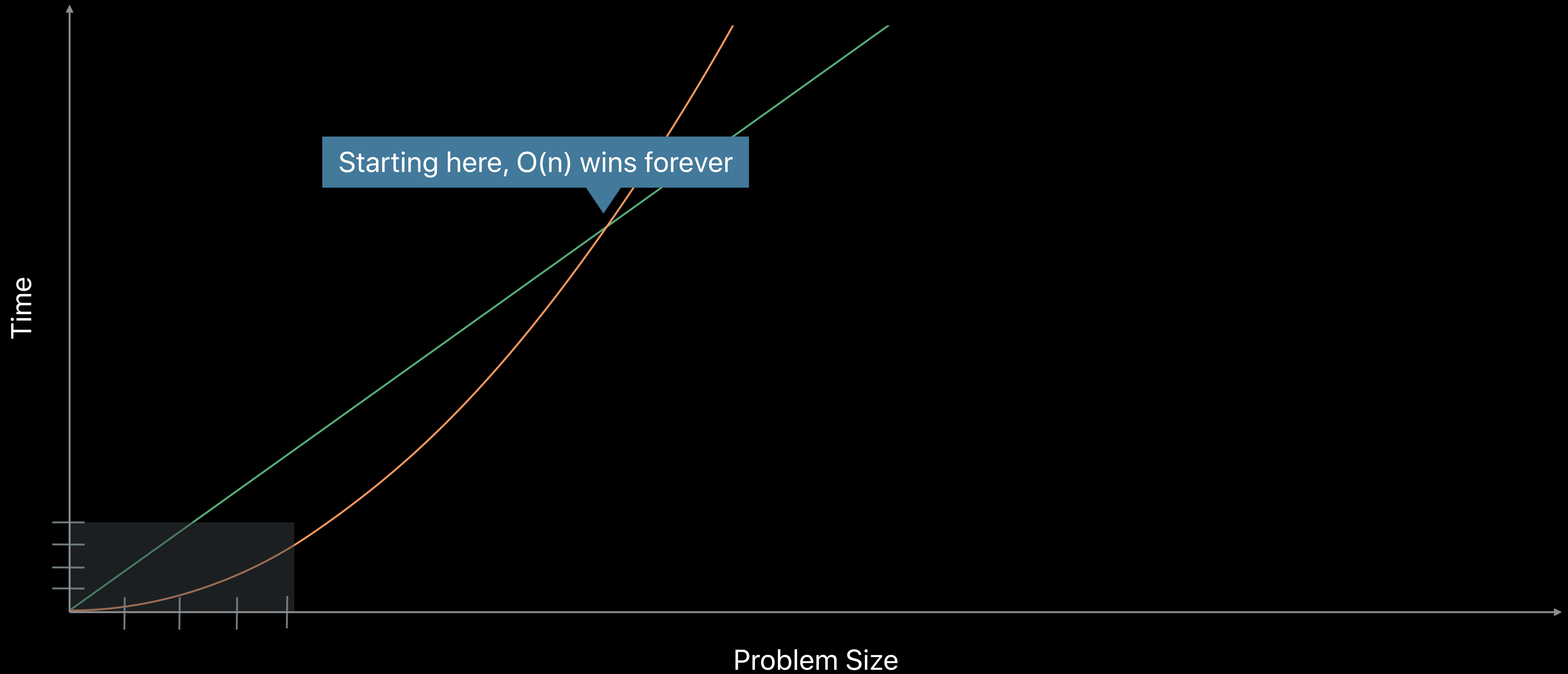
# Linear vs Quadratic



# Linear vs Quadratic



# Linear vs Quadratic





```
extension Canvas {  
  mutating func deleteSelection() {  
    shapes.removeAll(where: { $0.isSelected })  
  }  
}
```

```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// ...  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element) -> Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```



```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// ...  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element) -> Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```



```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// ...  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element)->Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```



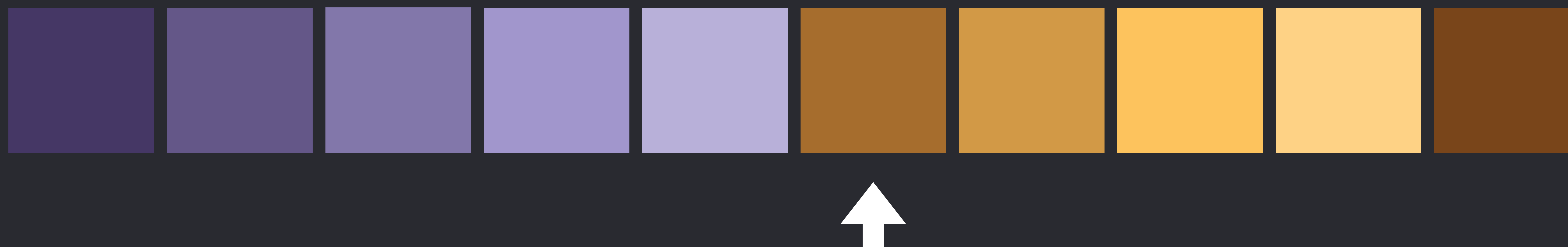
```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// ...  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element) -> Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```



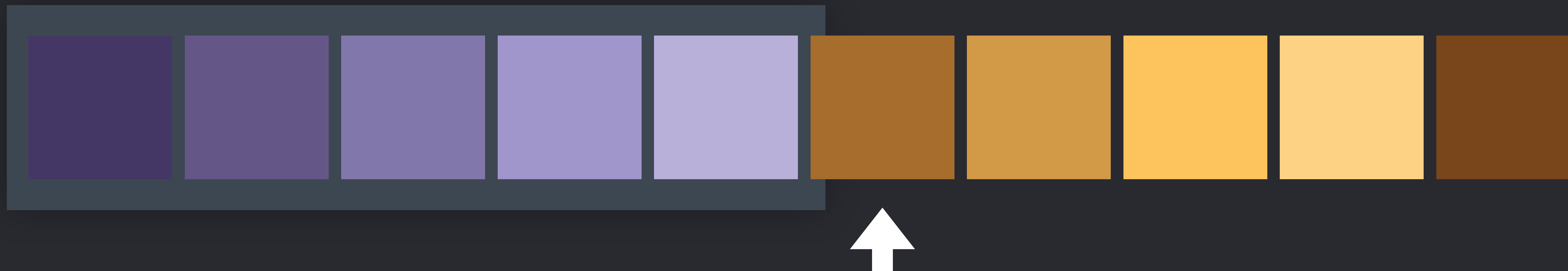
```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// ...  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element)->Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```



```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// ...  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element)->Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```

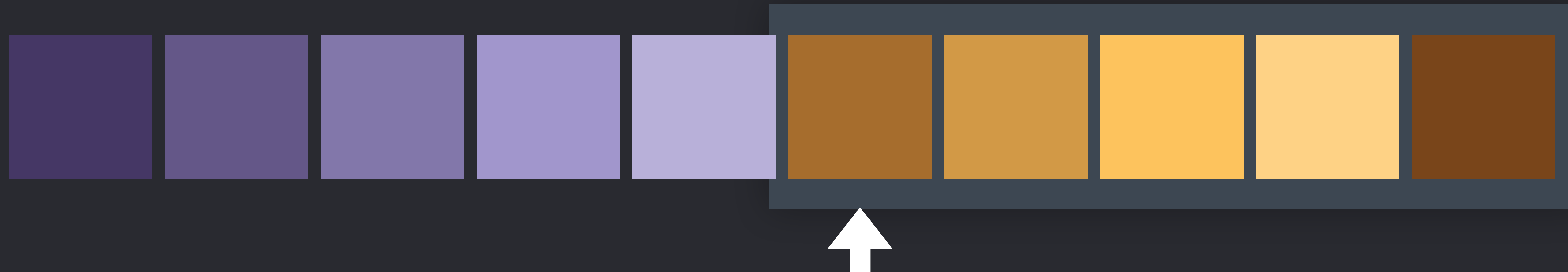


```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element) -> Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```





```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element)->Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```



```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element)->Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```



```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// - Complexity: O(n) where n is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element)->Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```

Means the same as  
suffixStart..  
endIndex



```
extension MutableCollection where Self : RangeReplaceableCollection {  
  /// Removes all elements satisfying `shouldRemove`.  
  ///  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func removeAll(where shouldRemove: (Element)->Bool) {  
    let suffixStart = halfStablePartition(isSuffixElement: shouldRemove)  
    removeSubrange(suffixStart...)  
  }  
}
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// returning the start position of the resulting suffix.
  ///
  /// - Complexity: O(n) where n is the number of elements.
  mutating func halfStablePartition(isSuffixElement: (Element) -> Bool) -> Index {
    guard var i = firstIndex(where: isSuffixElement) else { return endIndex }
    var j = index(after: i)
    while j != endIndex {
      if !isSuffixElement(self[j]) { swapAt(i, j); formIndex(after: &i) }
      formIndex(after: &j)
    }
    return i
  }
}

```



```
extension MutableCollection {  
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,  
  /// returning the start position of the resulting suffix.  
  ///  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func halfStablePartition(isSuffixElement: (Element) -> Bool) -> Index {  
    guard var i = firstIndex(where: isSuffixElement) else { return endIndex }  
    var j = index(after: i)  
    while j != endIndex {  
      if !isSuffixElement(self[j]) { swapAt(i, j); formIndex(after: &i) }  
      formIndex(after: &j)  
    }  
    return i  
  }  
}
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// returning the start position of the resulting suffix.
  ///
  /// - Complexity: O(n) where n is the number of elements.
  mutating func halfStablePartition(isSuffixElement: (Element) -> Bool) -> Index {
    guard var i = firstIndex(where: isSuffixElement) else { return endIndex }
    var j = index(after: i)
    while j != endIndex {
      if !isSuffixElement(self[j]) { swapAt(i, j); formIndex(after: &i) }
      formIndex(after: &j)
    }
    return i
  }
}

```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// returning the start position of the resulting suffix.
  ///
  /// - Complexity: O(n) where n is the number of elements.
  mutating func halfStablePartition(isSuffixElement: (Element) -> Bool) -> Index {
    guard var i = firstIndex(where: isSuffixElement) else { return endIndex }
    var j = index(after: i)
    while j != endIndex {
      if !isSuffixElement(self[j]) { swapAt(i, j); formIndex(after: &i) }
      formIndex(after: &j)
    }
    return i
  }
}

```





```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// returning the start position of the resulting suffix.
  ///
  /// - Complexity: O(n) where n is the number of elements.
  mutating func halfStablePartition(isSuffixElement: (Element) -> Bool) -> Index {
    guard var i = firstIndex(where: isSuffixElement) else { return endIndex }
    var j = index(after: i)
    while j != endIndex {
      if !isSuffixElement(self[j]) { swapAt(i, j); formIndex(after: &i) }
      formIndex(after: &j)
    }
    return i
  }
}

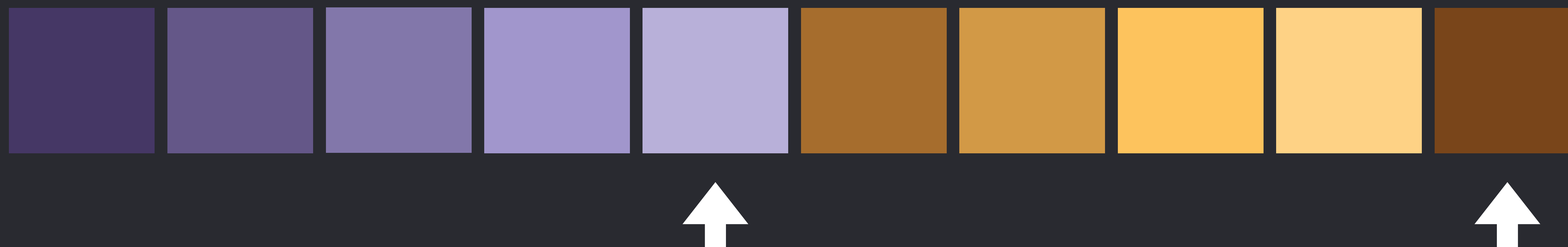
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// returning the start position of the resulting suffix.
  ///
  /// - Complexity: O(n) where n is the number of elements.
  mutating func halfStablePartition(isSuffixElement: (Element) -> Bool) -> Index {
    guard var i = firstIndex(where: isSuffixElement) else { return endIndex }
    var j = index(after: i)
    while j != endIndex {
      if !isSuffixElement(self[j]) { swapAt(i, j); formIndex(after: &i) }
      formIndex(after: &j)
    }
    return i
  }
}

```



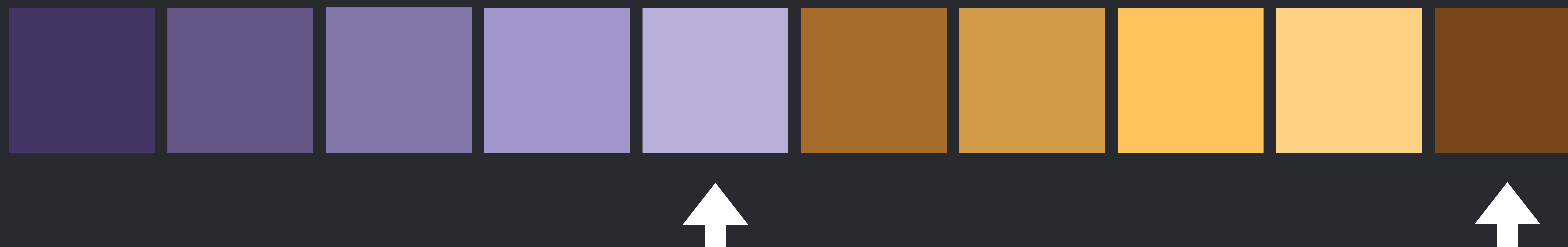
```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// returning the start position of the resulting suffix.
  ///
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.
  mutating func halfStablePartition(isSuffixElement: (Element) -> Bool) -> Index {
    guard var i = firstIndex(where: isSuffixElement) else { return endIndex }
    var j = index(after: i)
    while j != endIndex {
      if !isSuffixElement(self[j]) { swapAt(i, j); formIndex(after: &i) }
      formIndex(after: &j)
    }
    return i
  }
}

```



```
extension MutableCollection {  
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,  
  /// returning the start position of the resulting suffix.  
  ///  
  /// - Complexity:  $O(n)$  where  $n$  is the number of elements.  
  mutating func halfStablePartition(isSuffixElement: (Element) -> Bool) -> Index {  
    guard var i = firstIndex(where: isSuffixElement) else { return endIndex }  
    var j = index(after: i)  
    while j != endIndex {  
      if !isSuffixElement(self[j]) { swapAt(i, j); formIndex(after: &i) }  
      formIndex(after: &j)  
    }  
    return i  
  }  
}
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// returning the start position of the resulting suffix.
  ///
  /// - Complexity: O(n) where n is the number of elements.
  mutating func halfStablePartition(isSuffixElement: (Element) -> Bool) -> Index {
    guard var i = firstIndex(where: isSuffixElement) else { return endIndex }
    var j = index(after: i)
    while j != endIndex {
      if !isSuffixElement(self[j]) { swapAt(i, j); formIndex(after: &i) }
      formIndex(after: &j)
    }
    return i
  }
}

```



**Get to know your standard library**



# Swift Standard Library

Solve complex problems and write high-performance, readable code.

On This Page

[Overview](#) ⌵

[Topics](#) ⌵

[See Also](#) ⌵

## Overview

The Swift standard library defines a base layer of functionality for writing Swift programs, including:

- Fundamental data types such as [Int](#), [Double](#), and [String](#)
- Common data structures such as [Array](#), [Dictionary](#), and [Set](#)
- Global functions such as [print\(\\_:separator:terminator:\)](#) and [abs\(\\_:\)](#)
- Protocols, such as [Collection](#) and [Equatable](#), that describe common abstractions.
- Protocols, such as [CustomDebugStringConvertible](#) and [CustomReflectable](#), that you use to customize operations that are available to all types.
- Protocols, such as [OptionSet](#), that you use to provide implementations that would otherwise require boilerplate code.

[https://developer.apple.com/documentation/swift/swift\\_standard\\_library](https://developer.apple.com/documentation/swift/swift_standard_library)

# Readability and Maintainability

Before

```
for i in (0..<shapes.count).reversed() {  
  if shapes[i].isSelected {  
    shapes.remove(at: i)  
  }  
}
```

After

```
shapes.removeAll(where: { $0.isSelected })
```



# Readability and Maintainability

Before

```
// Remove each selected item.  
for i in (0..<shapes.count).reversed() {  
    if shapes[i].isSelected {  
        shapes.remove(at: i)  
    }  
}
```

After

```
shapes.removeAll(where: { $0.isSelected })
```

# Readability and Maintainability

Before

```
// Remove each selected item.  
// Go backward to avoid shifting  
// as-yet-unprocessed elements.  
for i in (0..<shapes.count).reversed() {  
    if shapes[i].isSelected {  
        shapes.remove(at: i)  
    }  
}
```

After

```
shapes.removeAll(where: { $0.isSelected })
```

# Readability and Maintainability

Before

```
// Remove each selected item.  
// Go backward to avoid shifting  
// as-yet-unprocessed elements.  
for i in (0..  
    shapes.count).reversed() {  
    if shapes[i].isSelected {  
        shapes.remove(at: i)  
    }  
}
```

After

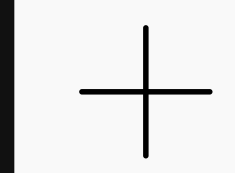
```
shapes.removeAll { $0.isSelected }
```

“No Raw Loops”

Sean Parent (a.k.a. “Crusty's cousin”), *C++ Seasoning*

```
// Done, and done!
```


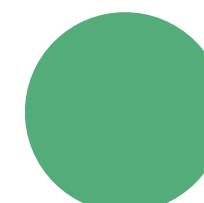

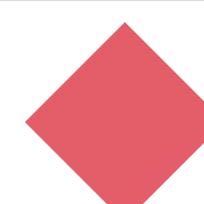

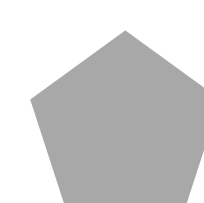
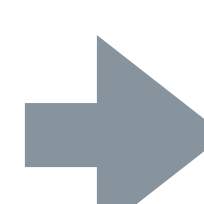

```
extension Canvas {  
    mutating func deleteSelection() {  
        shapes.removeAll { $0.isSelected }  
    }  
}
```

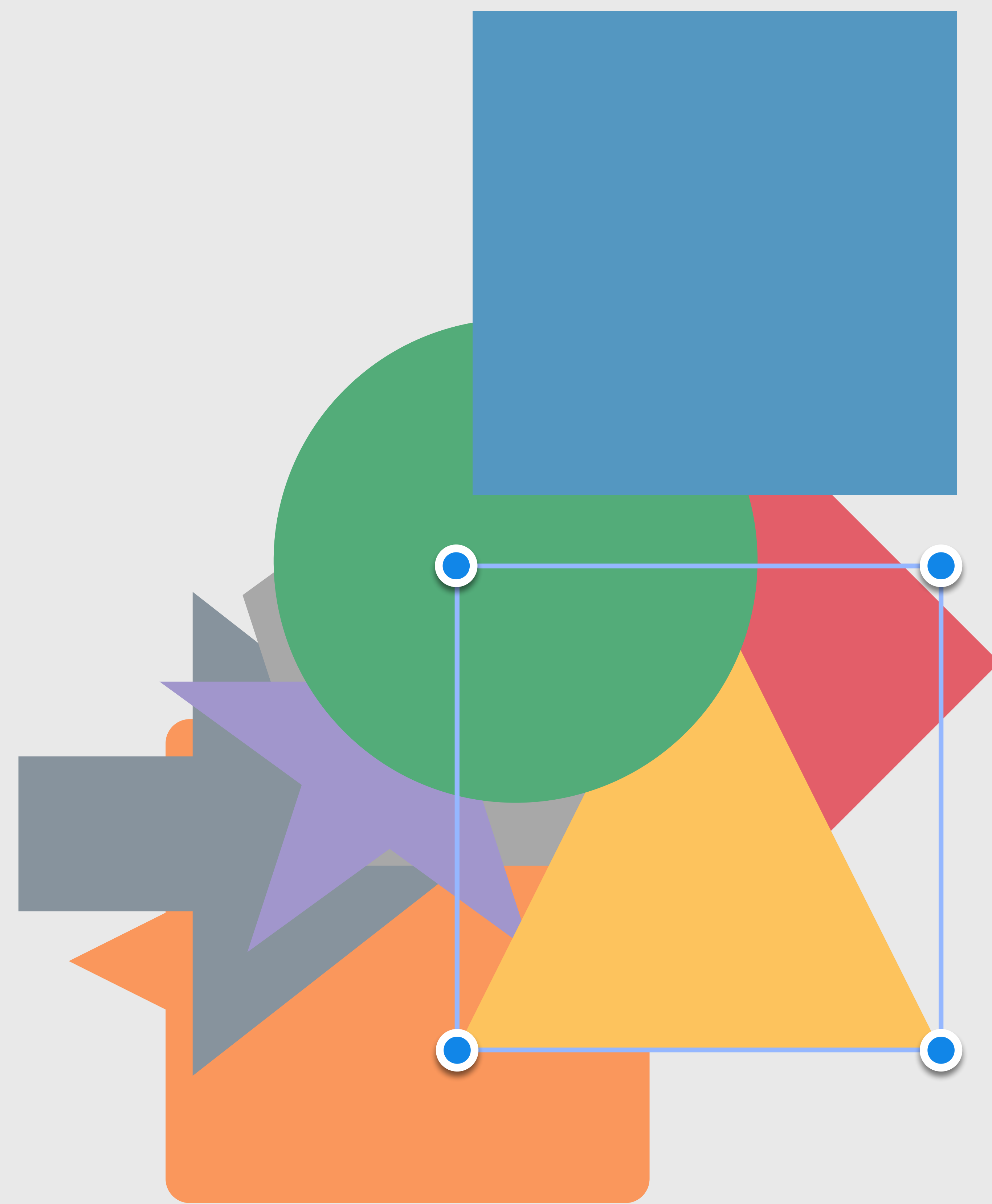


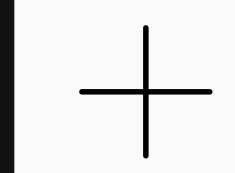
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Triangle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout



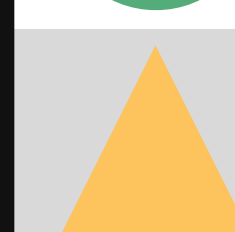
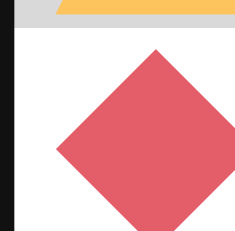
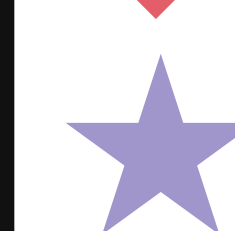
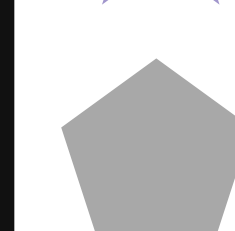




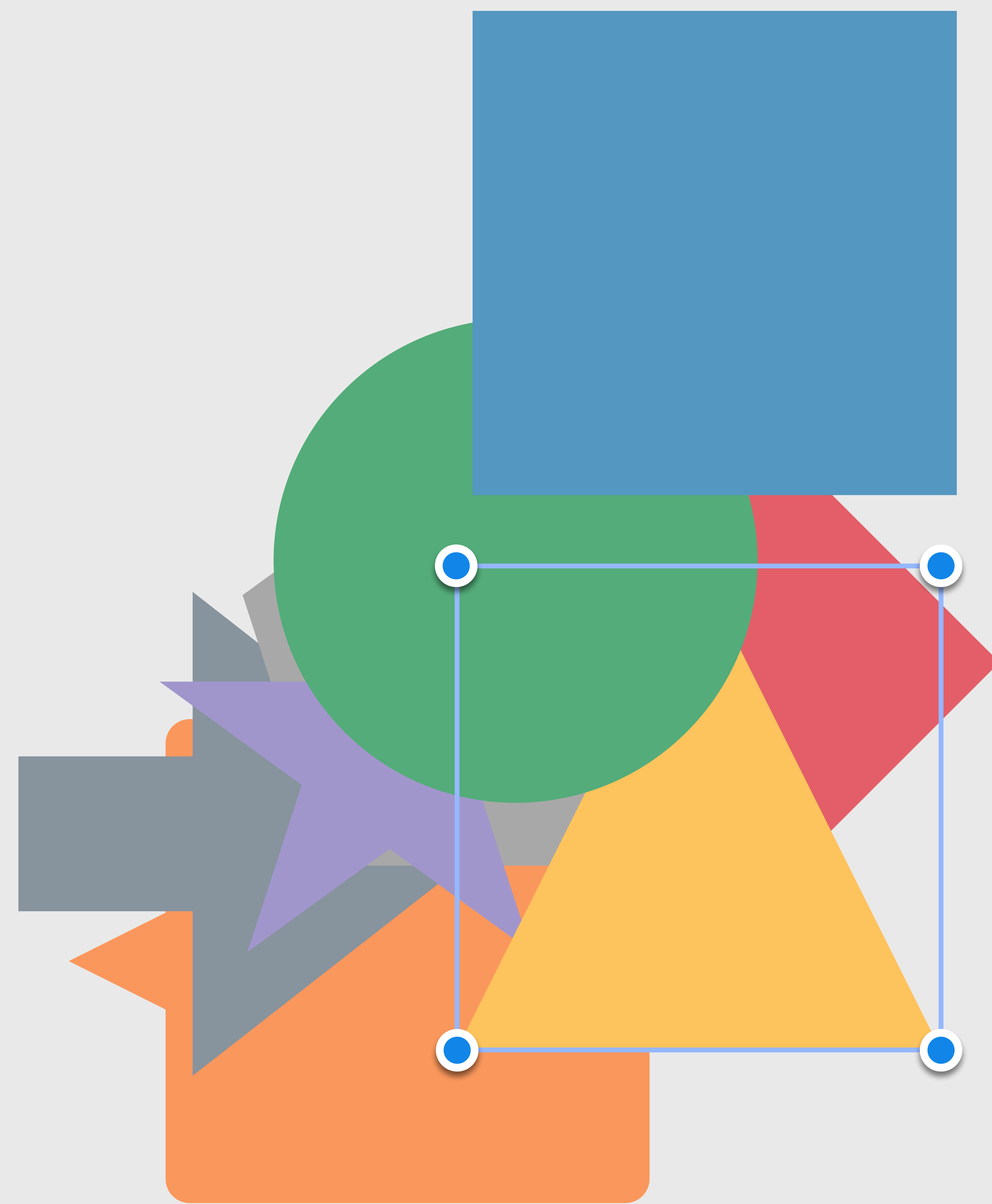


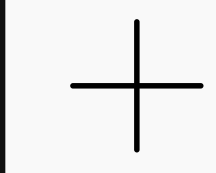
Bring Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Triangle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout





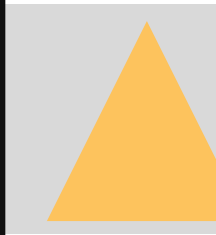

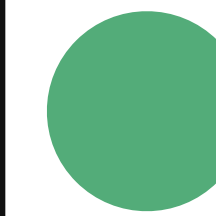
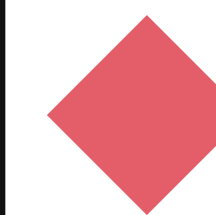
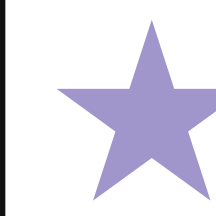
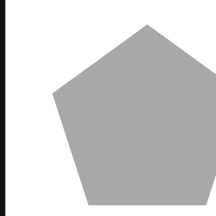
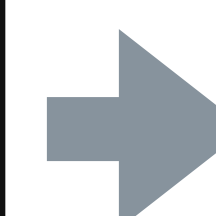

Bring Front

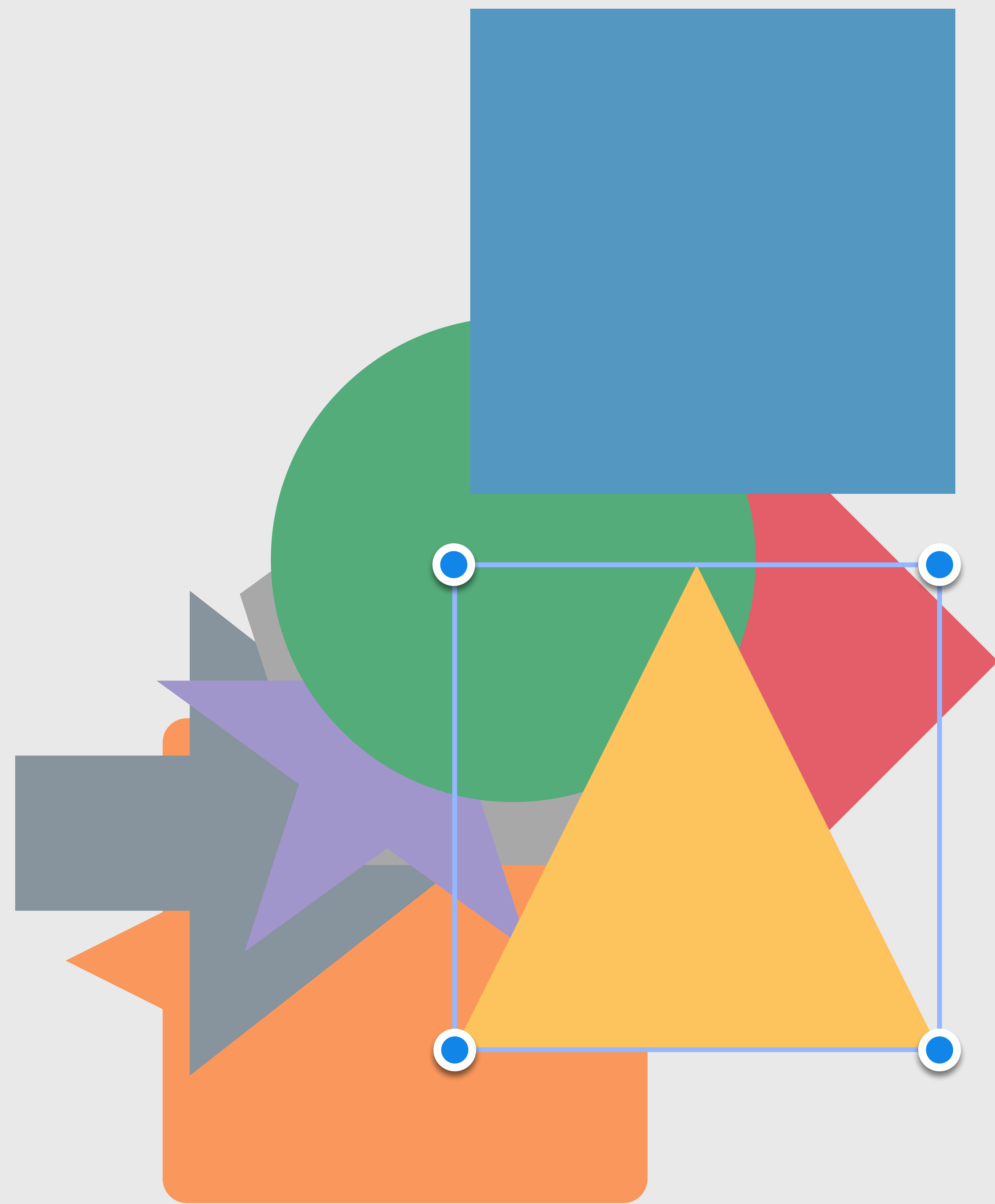
Send to Back

Shapes

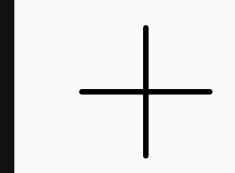
Bring Forward

Send Backward

-  Triangle
-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout





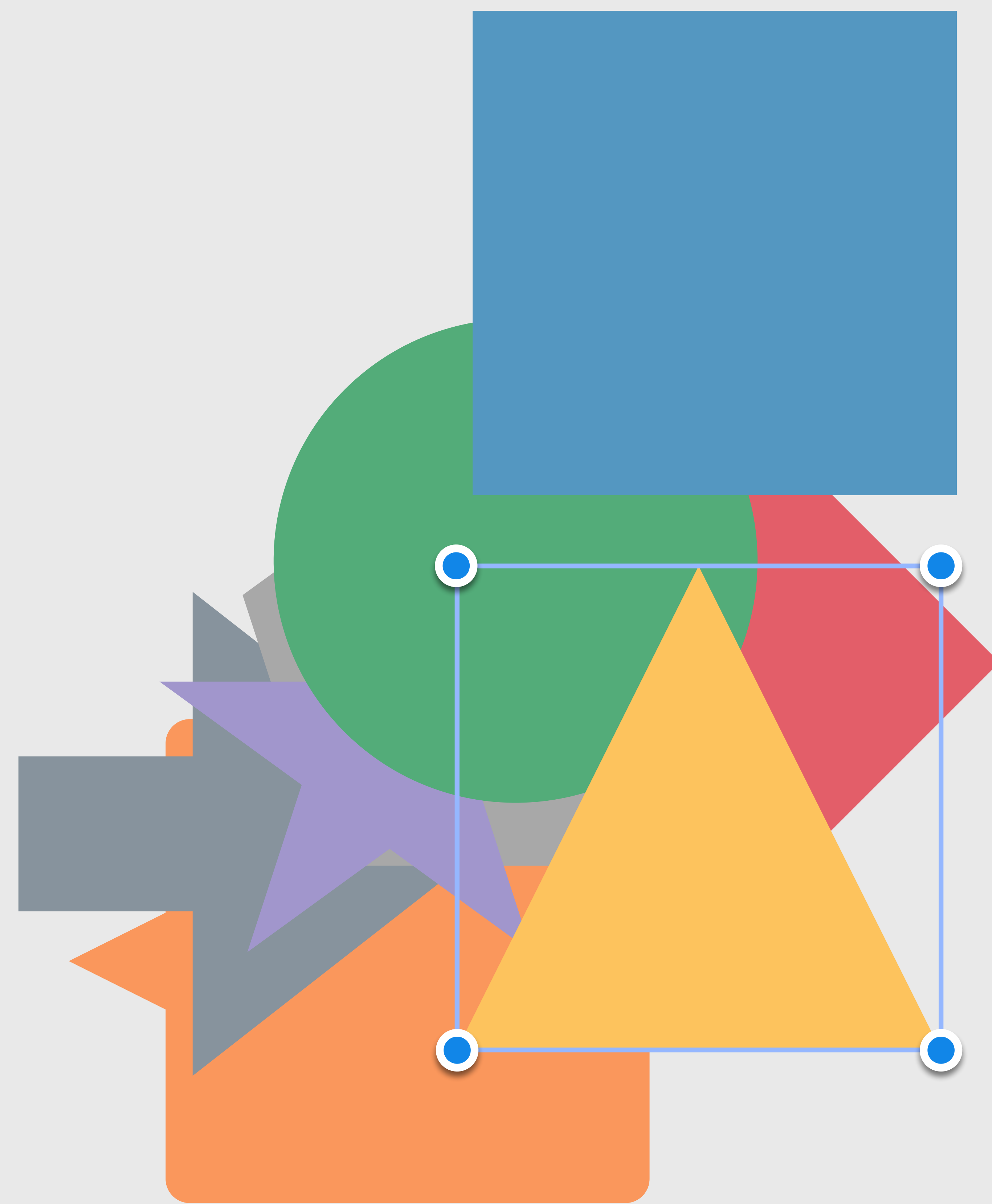


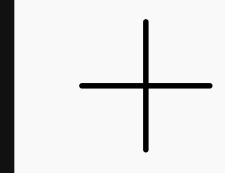
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Triangle
- Square
- Circle
- Diamond
- Star
- Polygon
- Arrow
- Callout





Bring to Front

Send Back

Shapes

Bring Forward

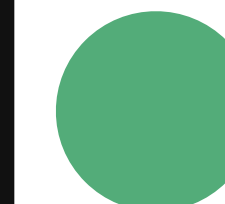
Send Backward



Triangle



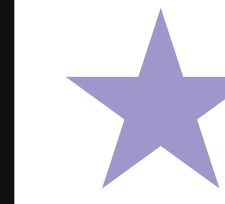
Square



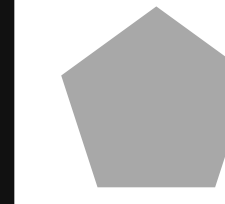
Circle



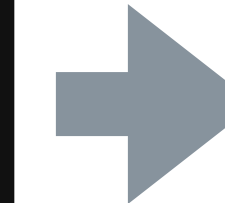
Diamond



Star



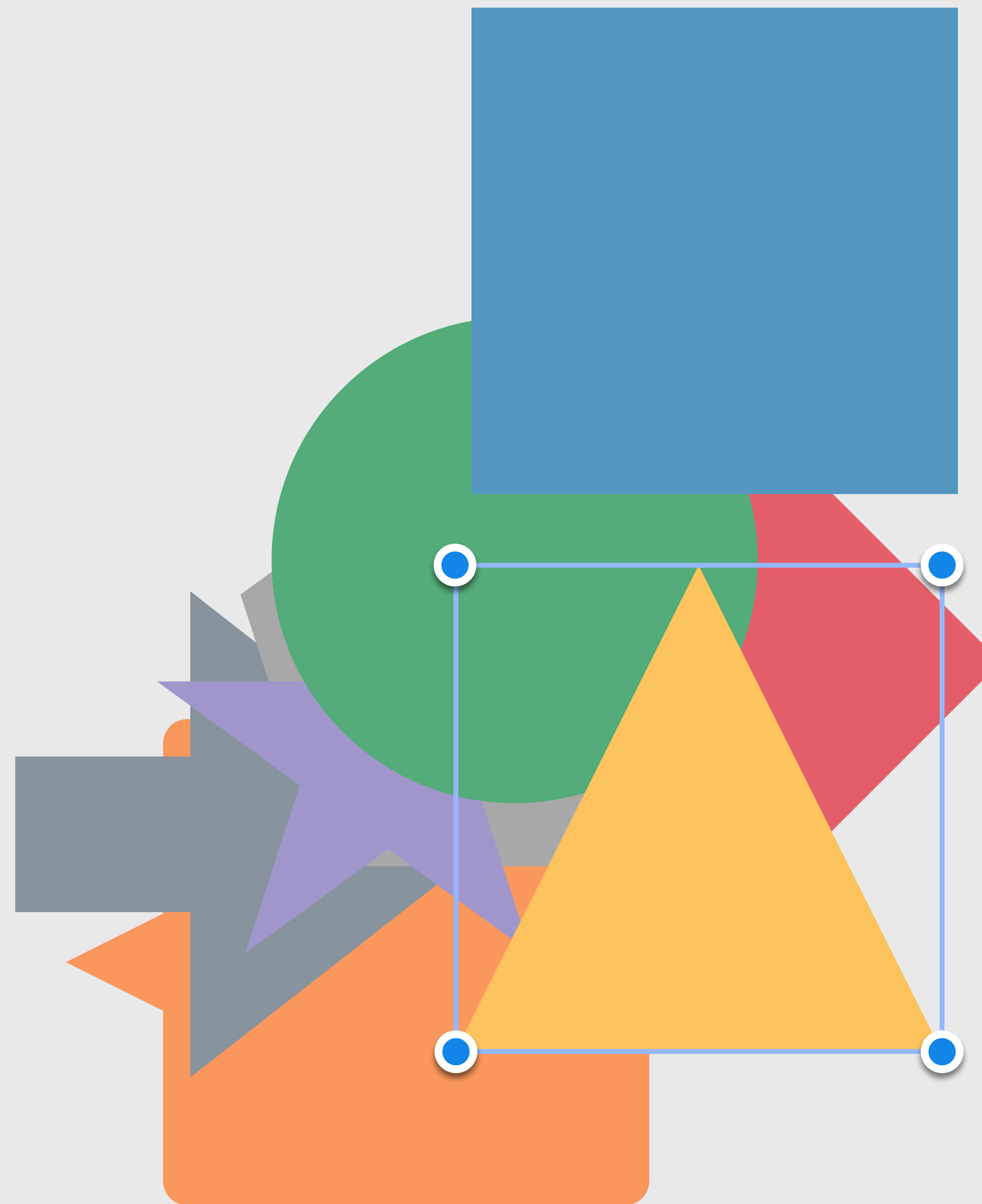
Polygon

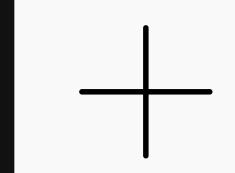


Arrow



Callout






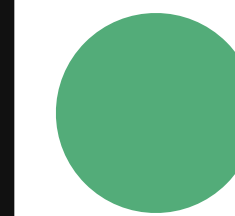
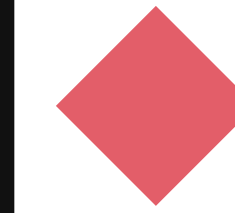
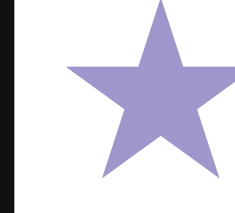
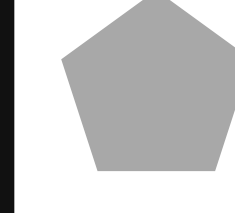
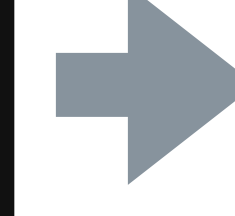


Bring to Front

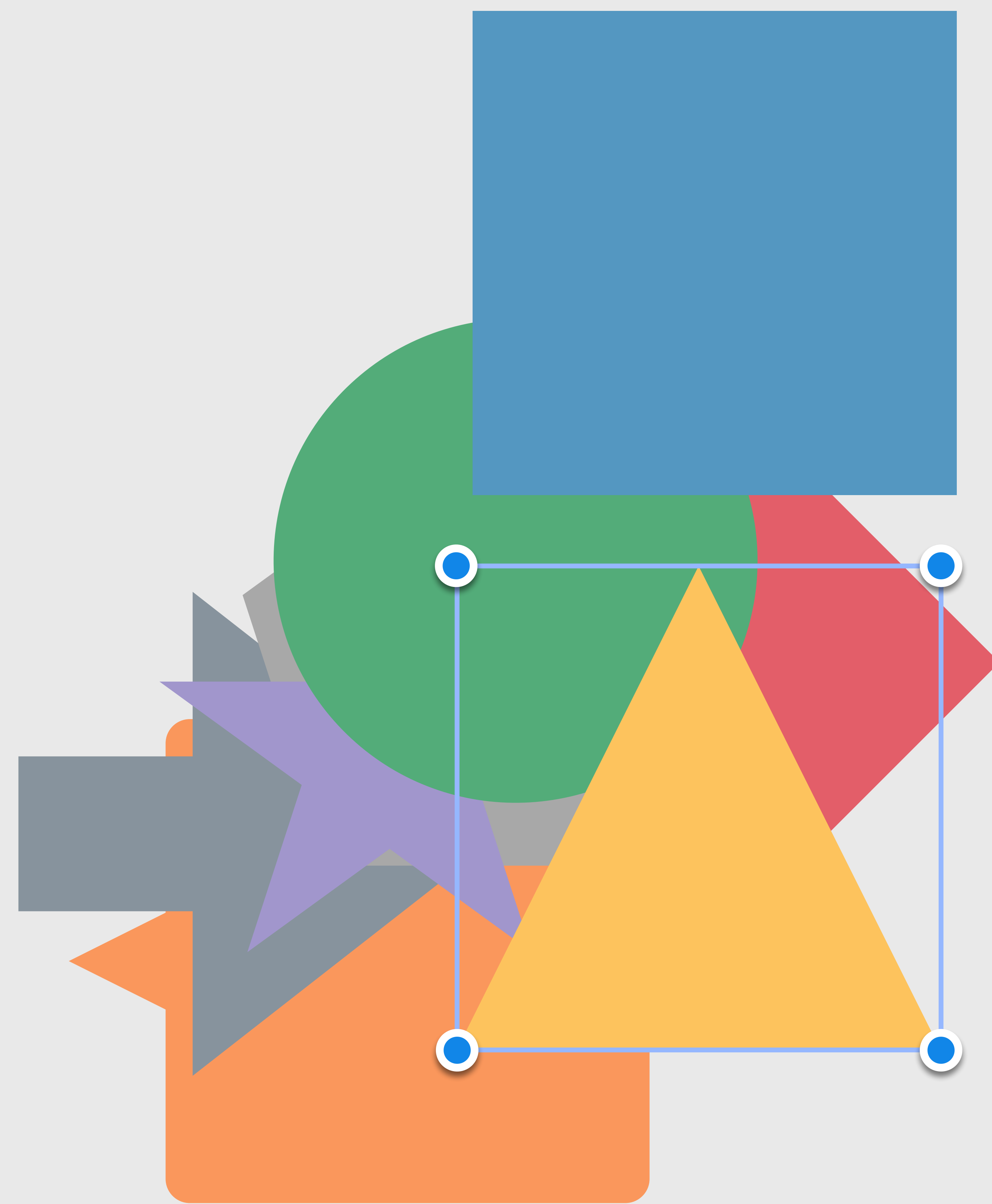
Send Back

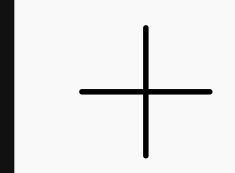
Shapes

Bring Forward

Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout
-  Triangle


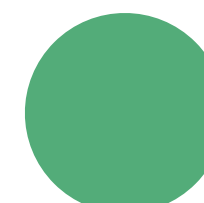
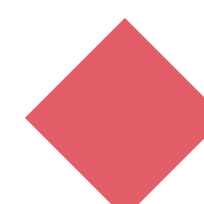
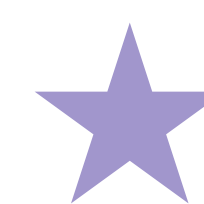
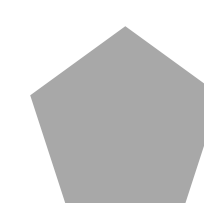
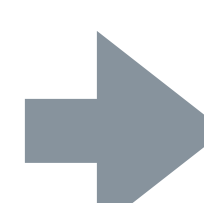




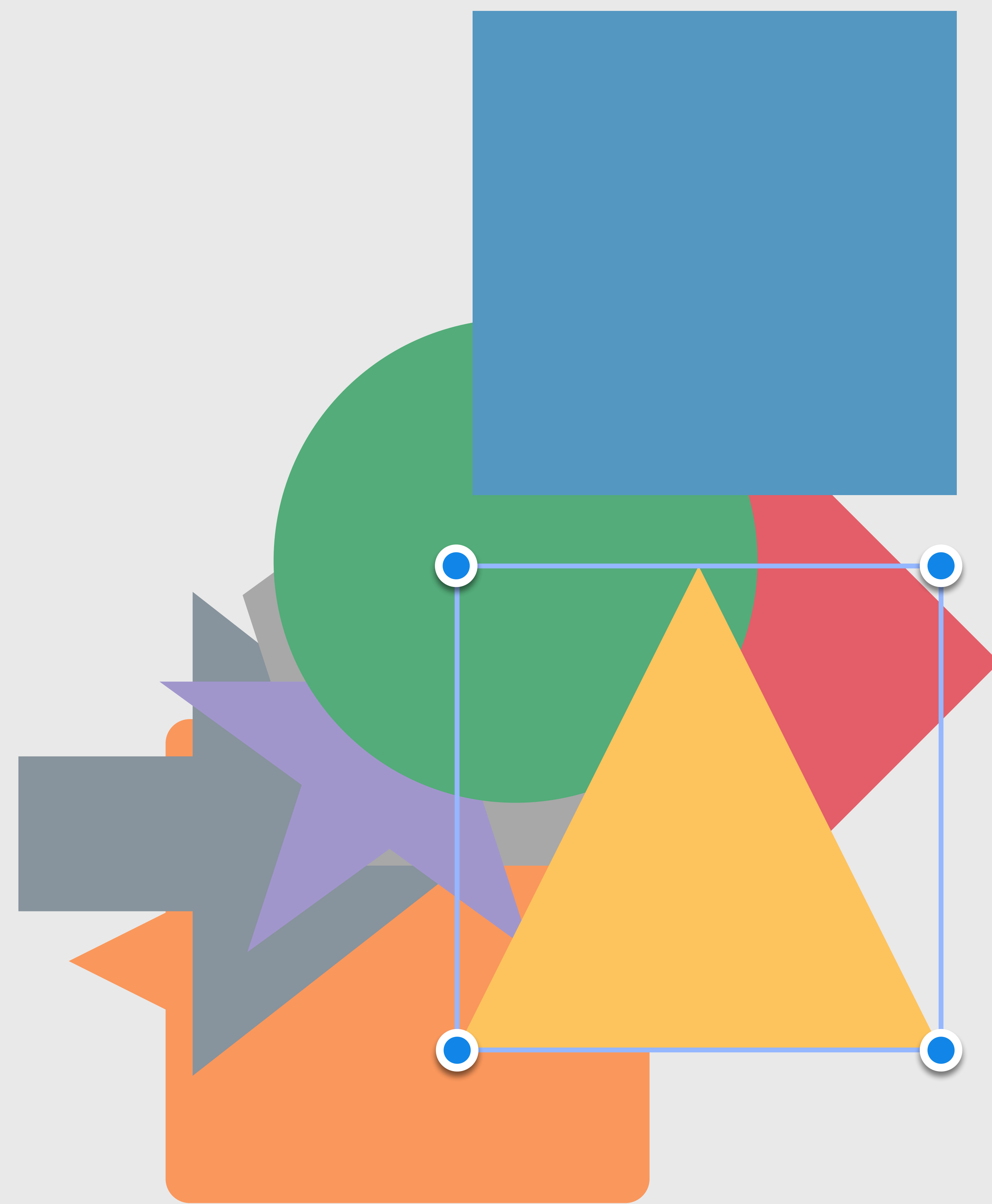


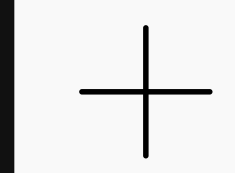
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout
-  Triangle


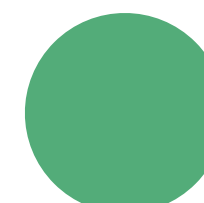
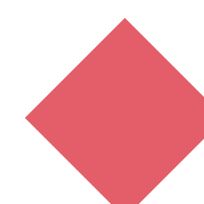
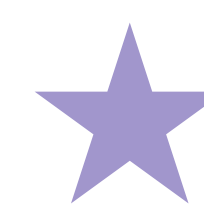
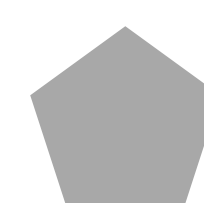
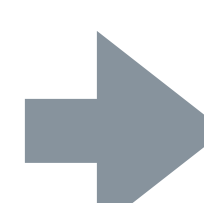




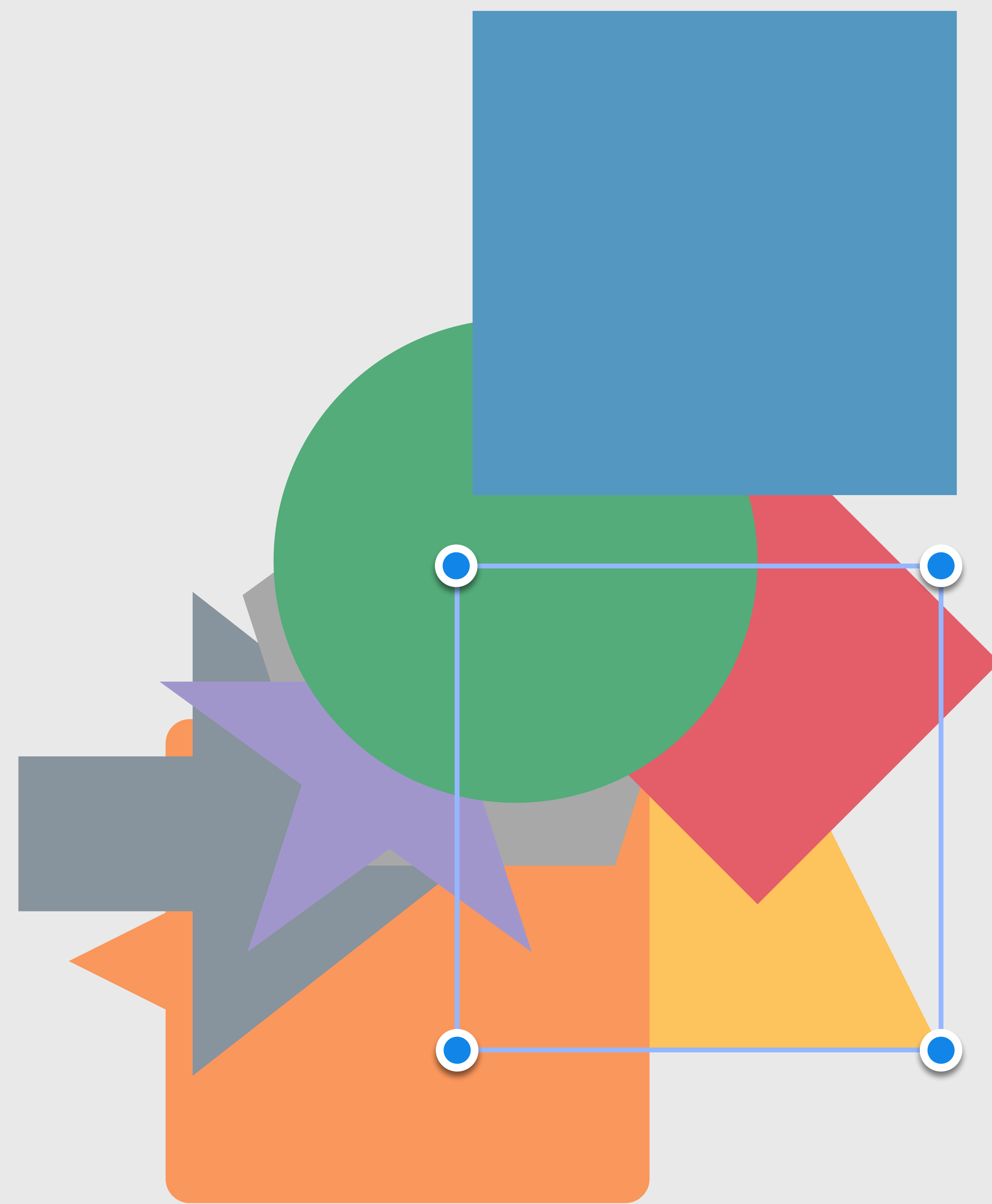


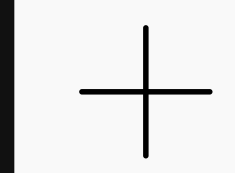
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout
-  Triangle



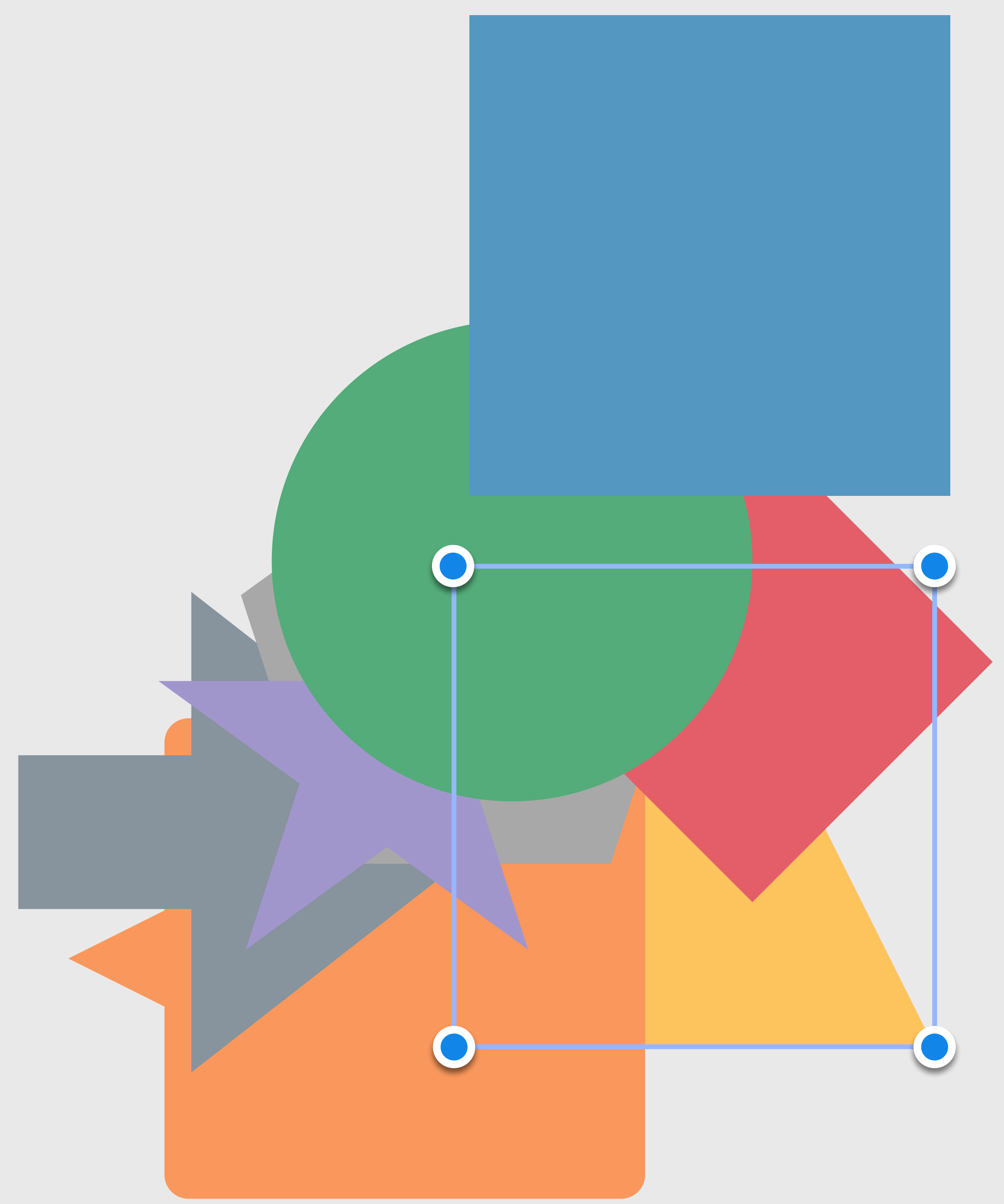


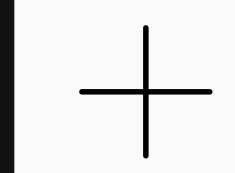
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Square
- Circle
- Diamond
- Star
- Polygon
- Arrow
- Callout
- Triangle



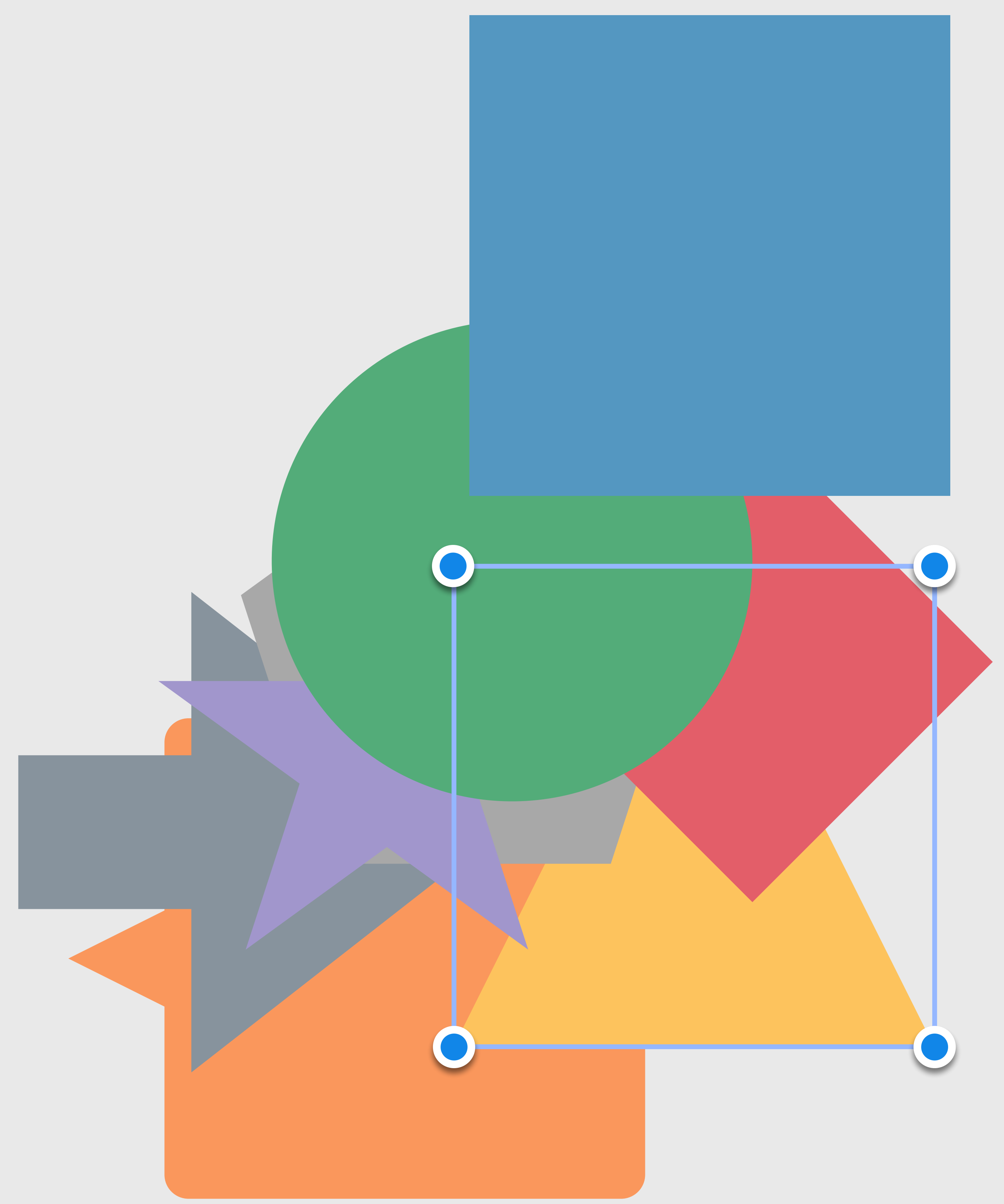


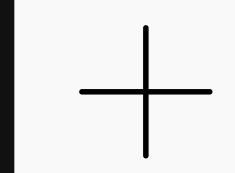
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Square
- Circle
- Diamond
- Star
- Polygon
- Arrow
- Triangle
- Callout


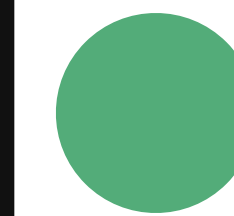
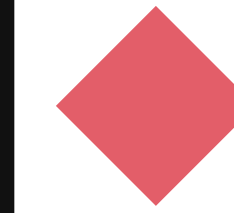
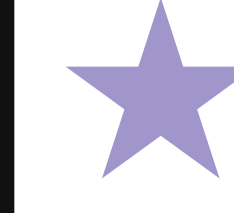
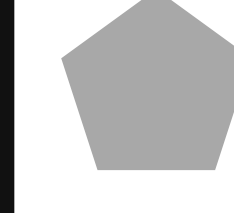
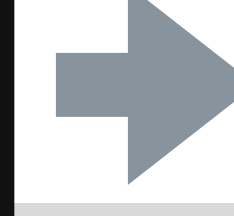




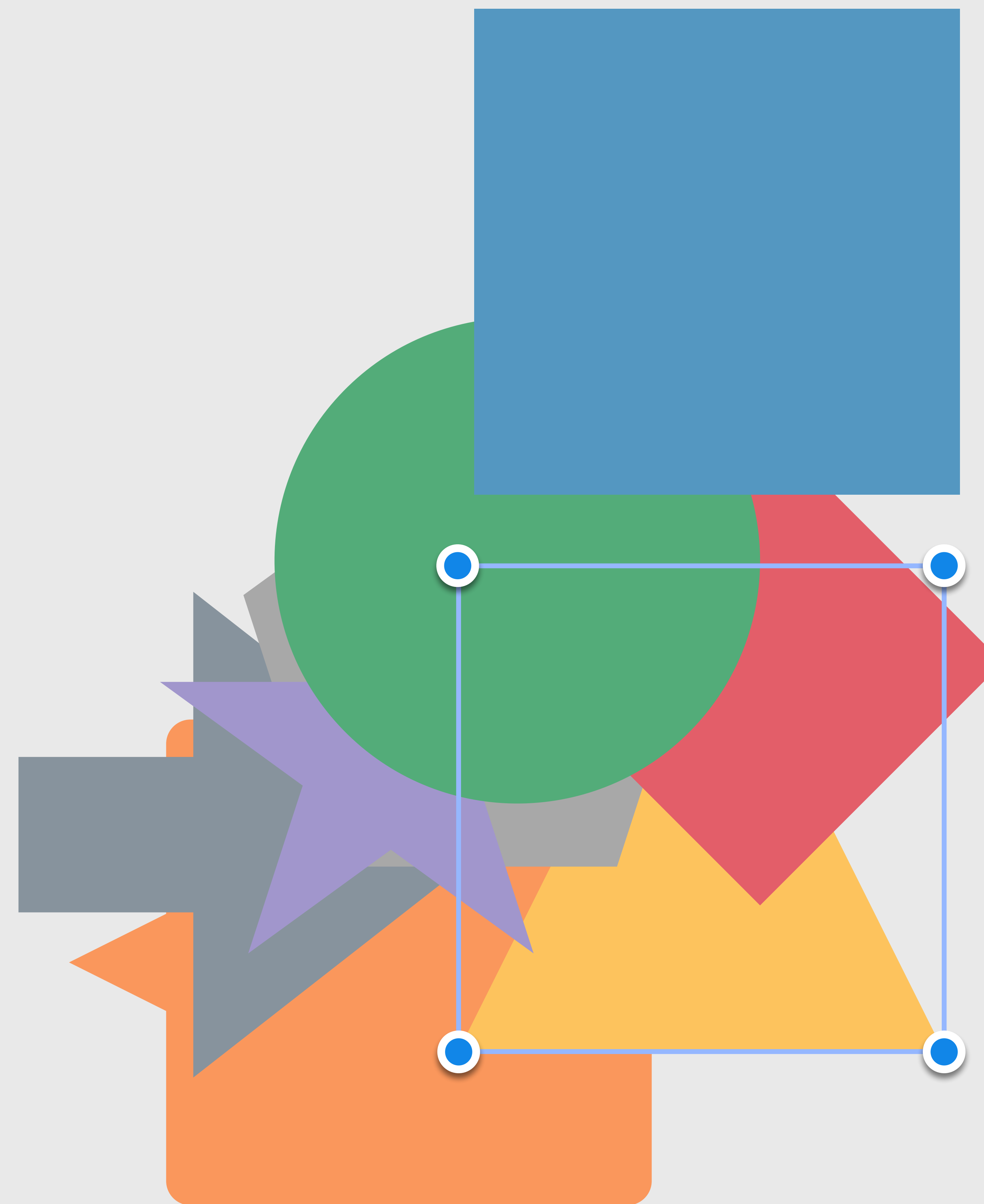


Bring to Front Send to Back

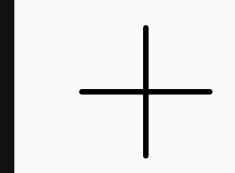
Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Triangle
-  Callout




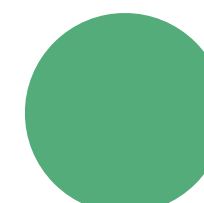
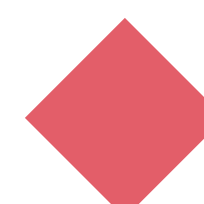
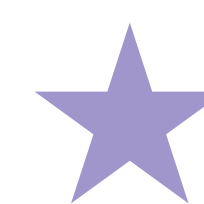
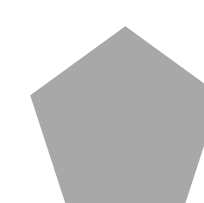
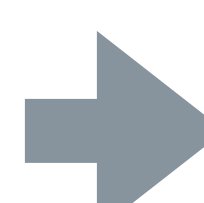




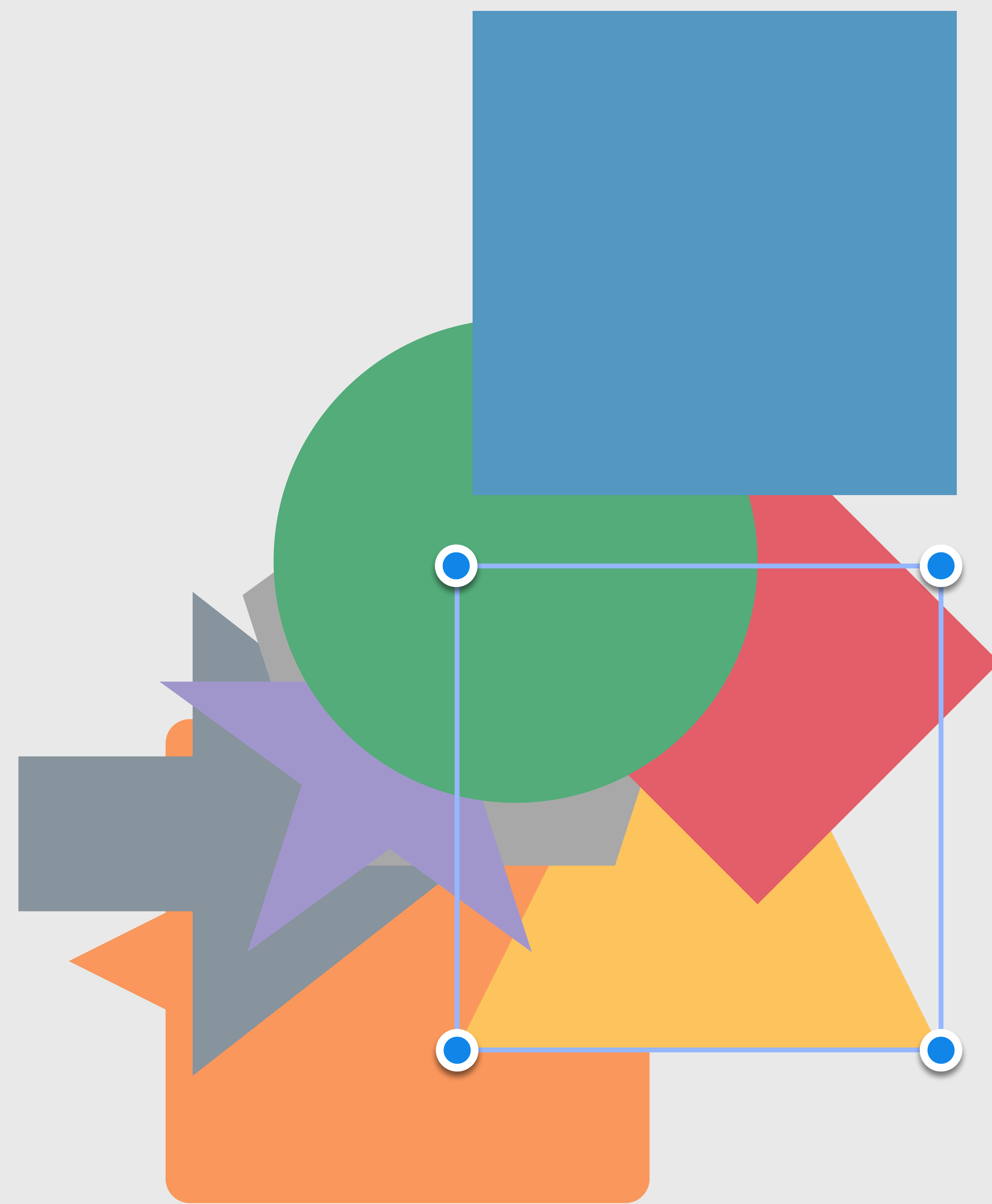


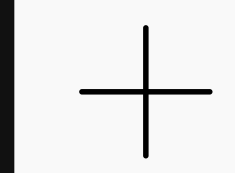
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Triangle
-  Callout


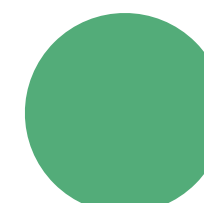
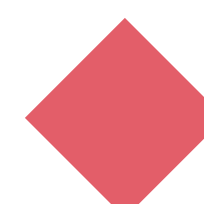
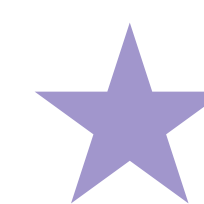
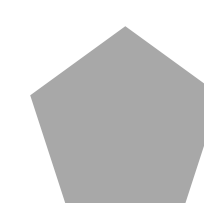

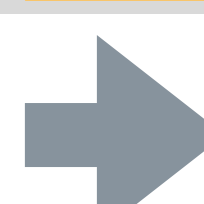



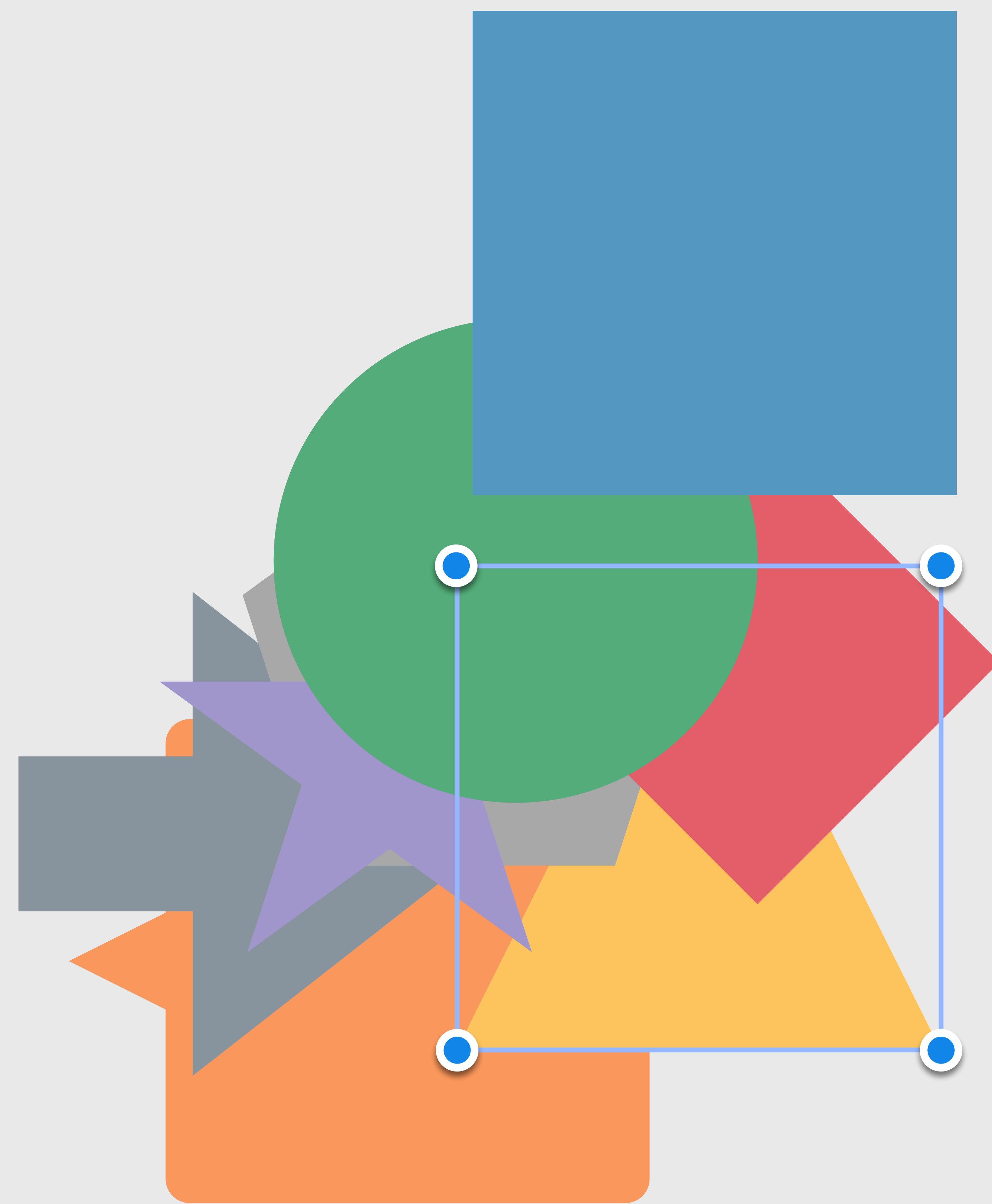


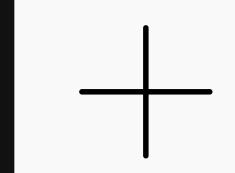
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Triangle
-  Arrow
-  Callout


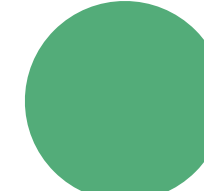
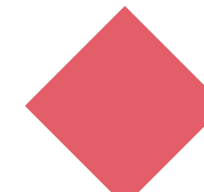

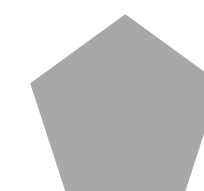

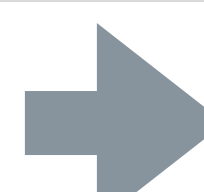



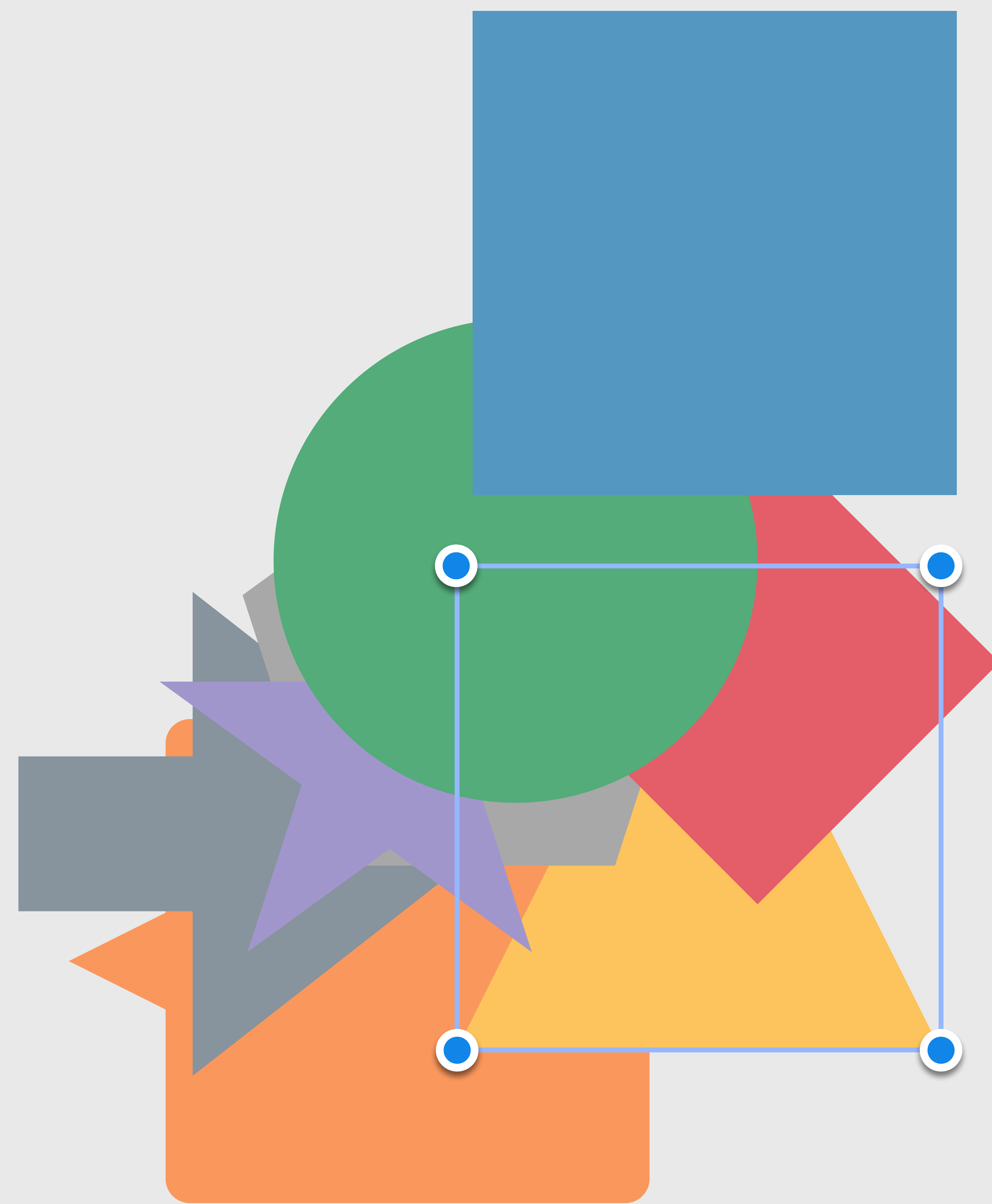


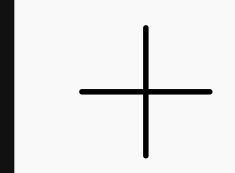
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Triangle
-  Arrow
-  Callout


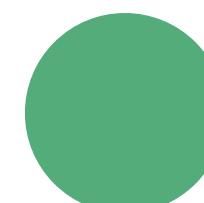
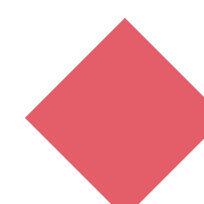
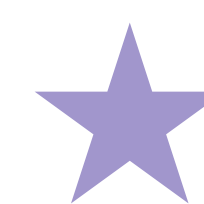
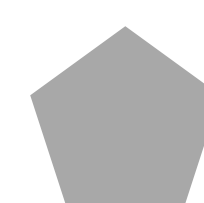

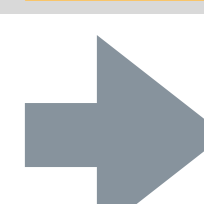



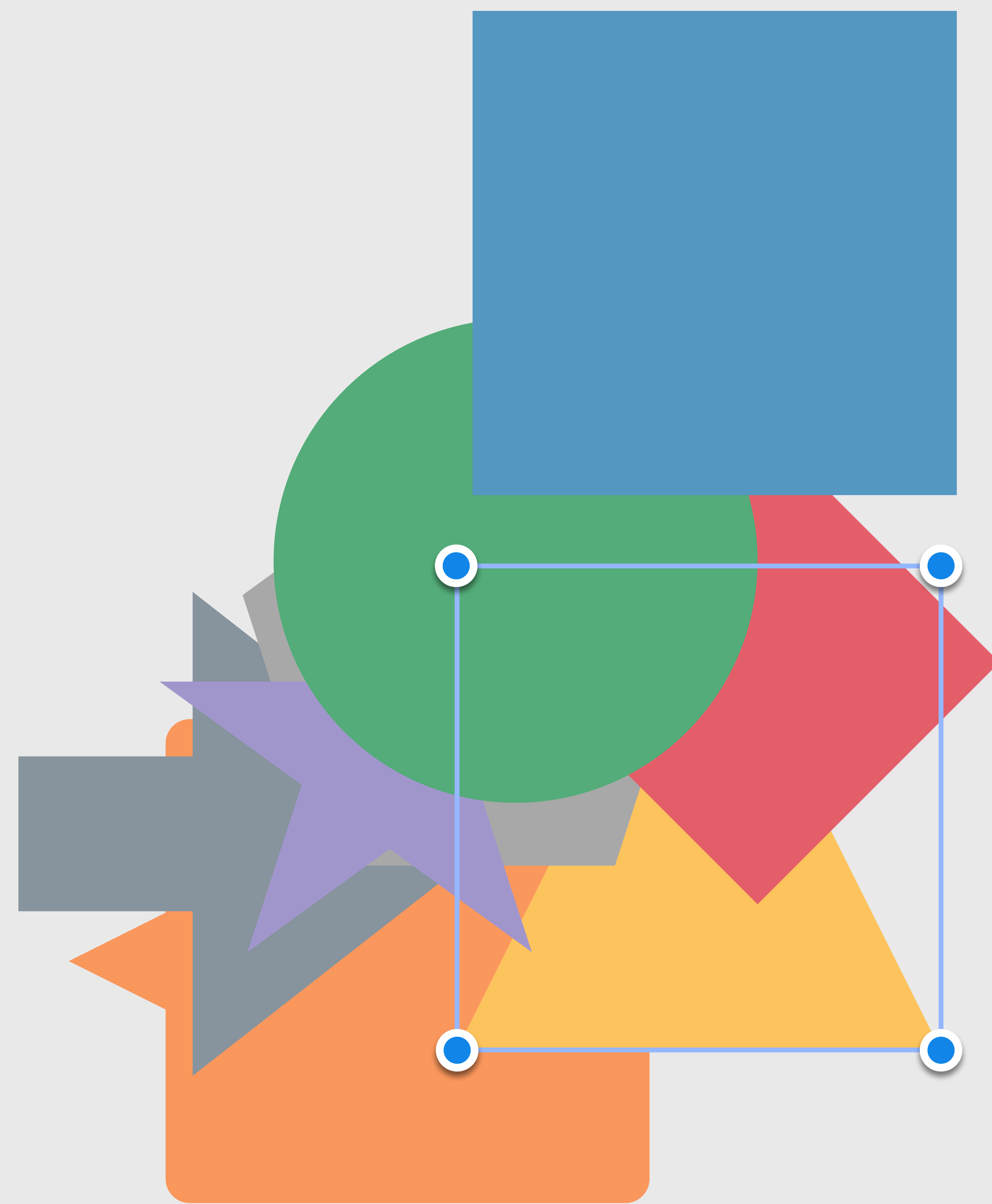


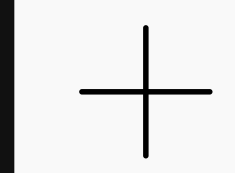
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Triangle
-  Arrow
-  Callout



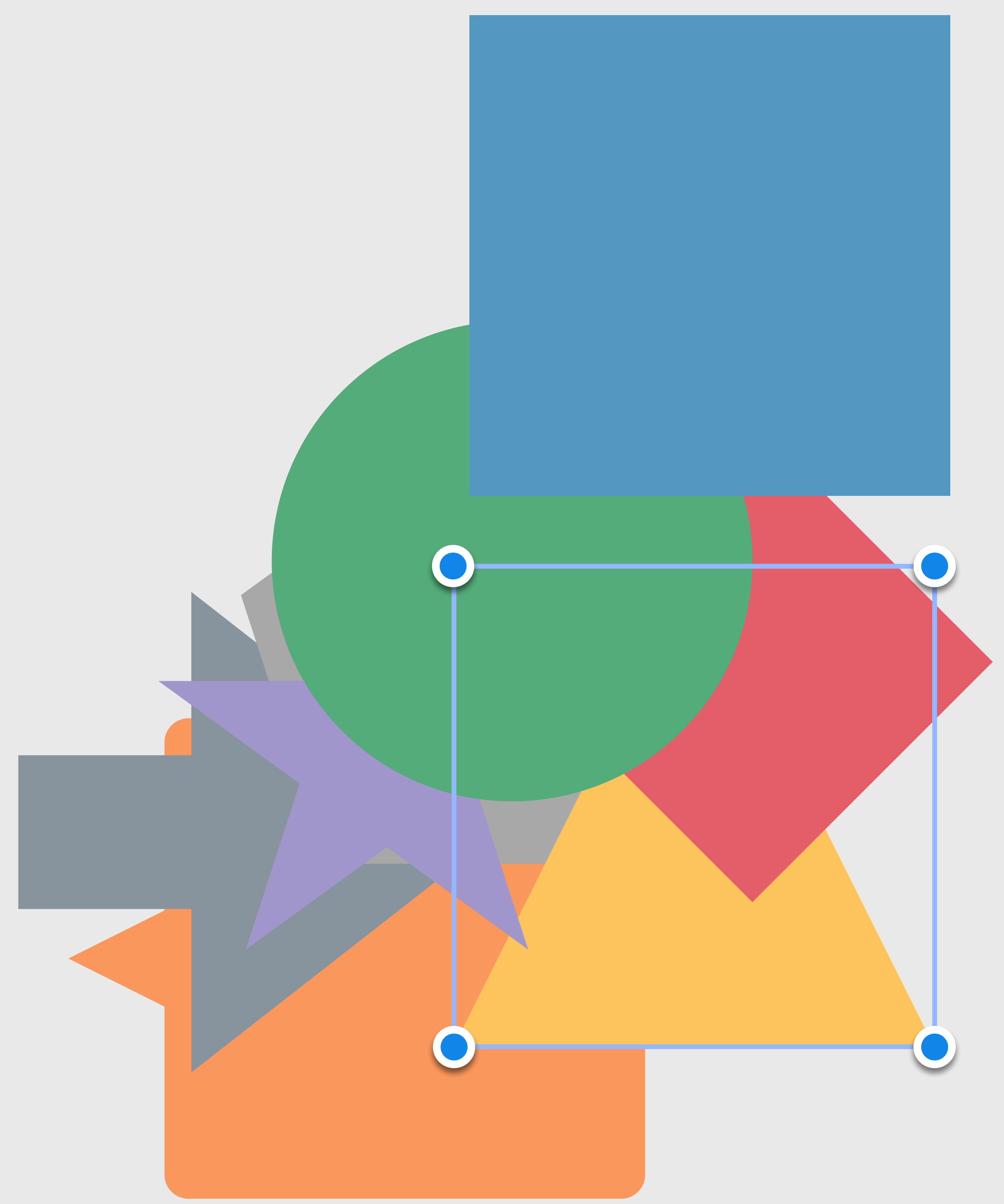


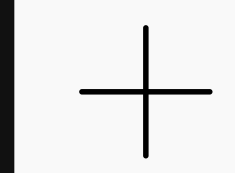
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Square
- Circle
- Diamond
- Star
- Triangle
- Polygon
- Arrow
- Callout


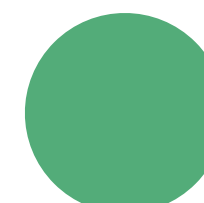
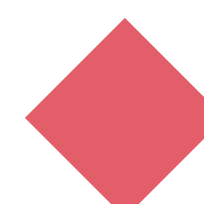
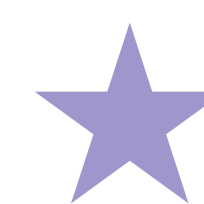

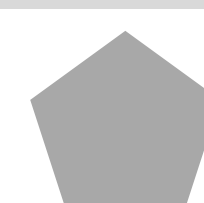
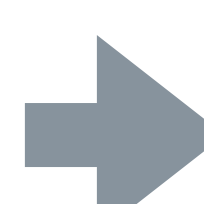



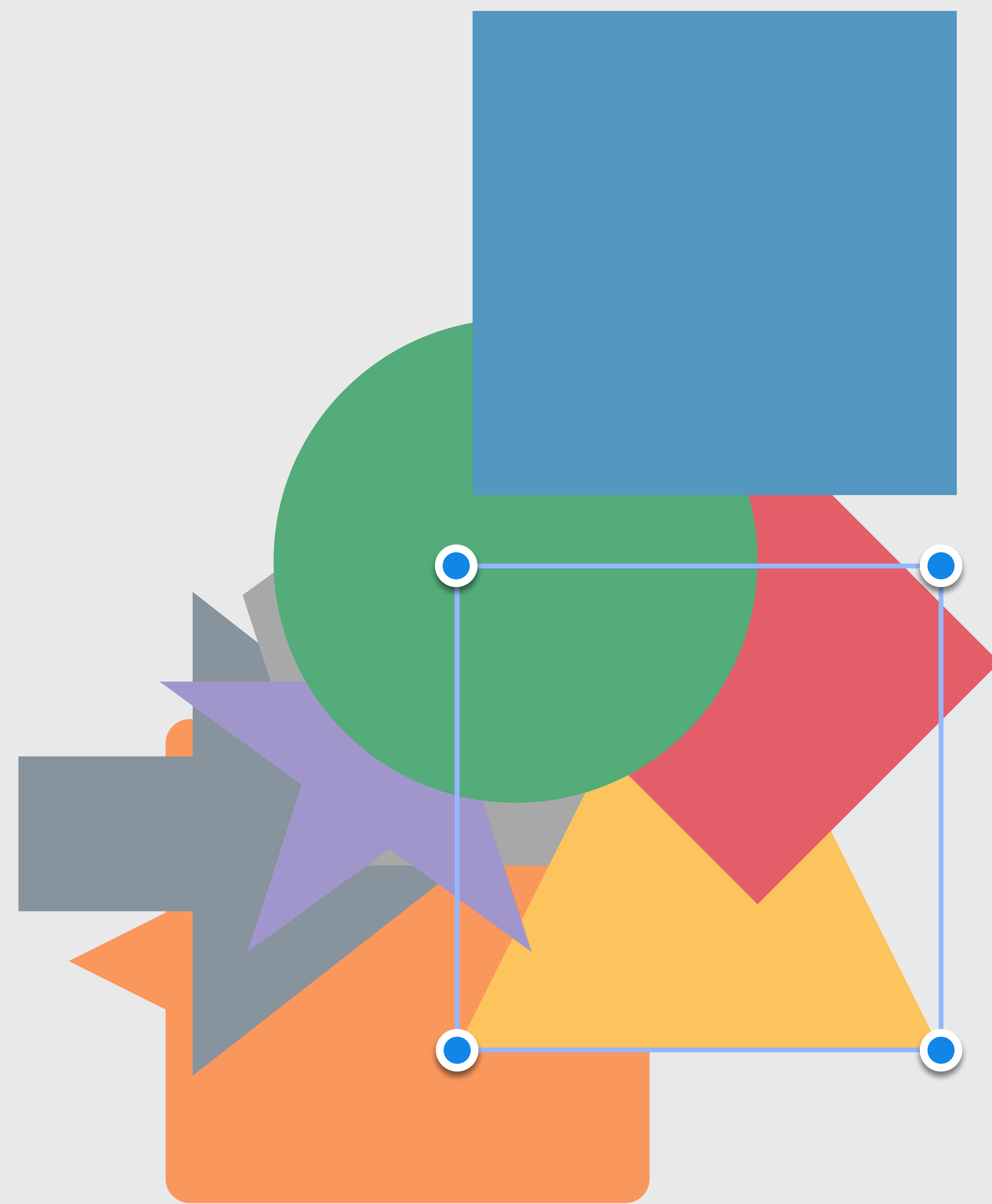


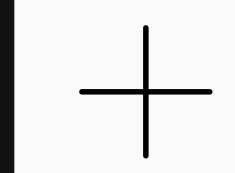
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Triangle
-  Polygon
-  Arrow
-  Callout


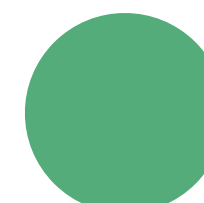
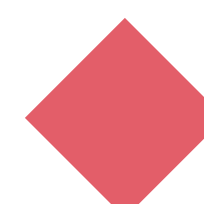
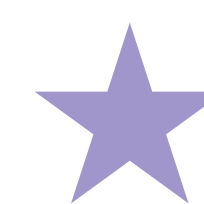

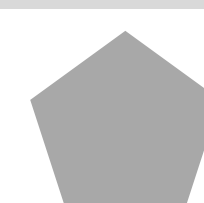
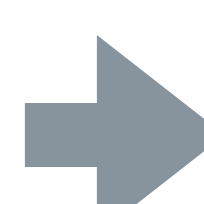



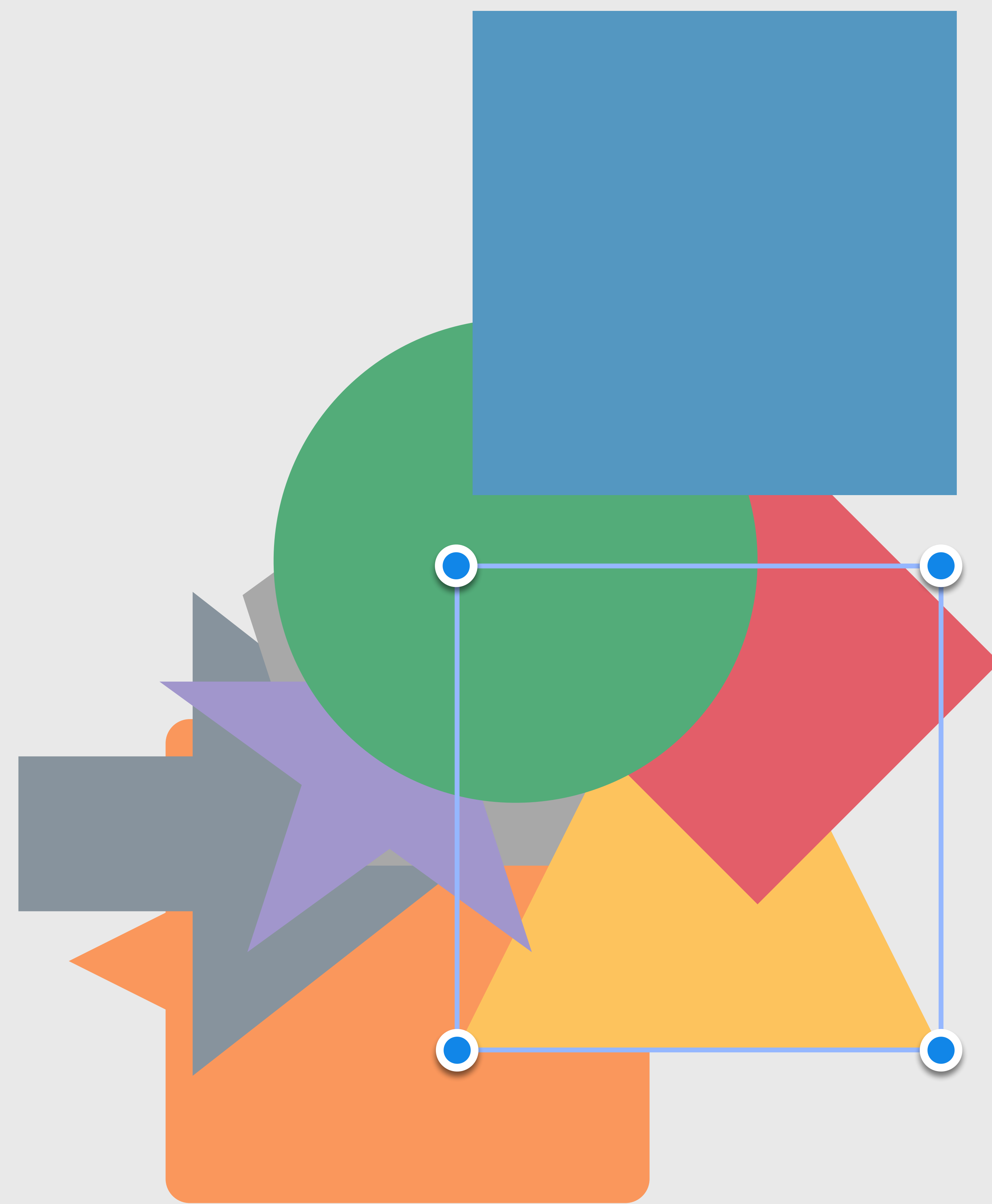


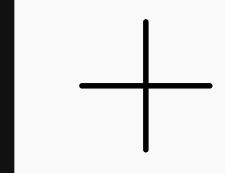
Bring to Front Send to Back

Shapes

Bring Forward Send Forward

-  Square
-  Circle
-  Diamond
-  Star
-  Triangle
-  Polygon
-  Arrow
-  Callout











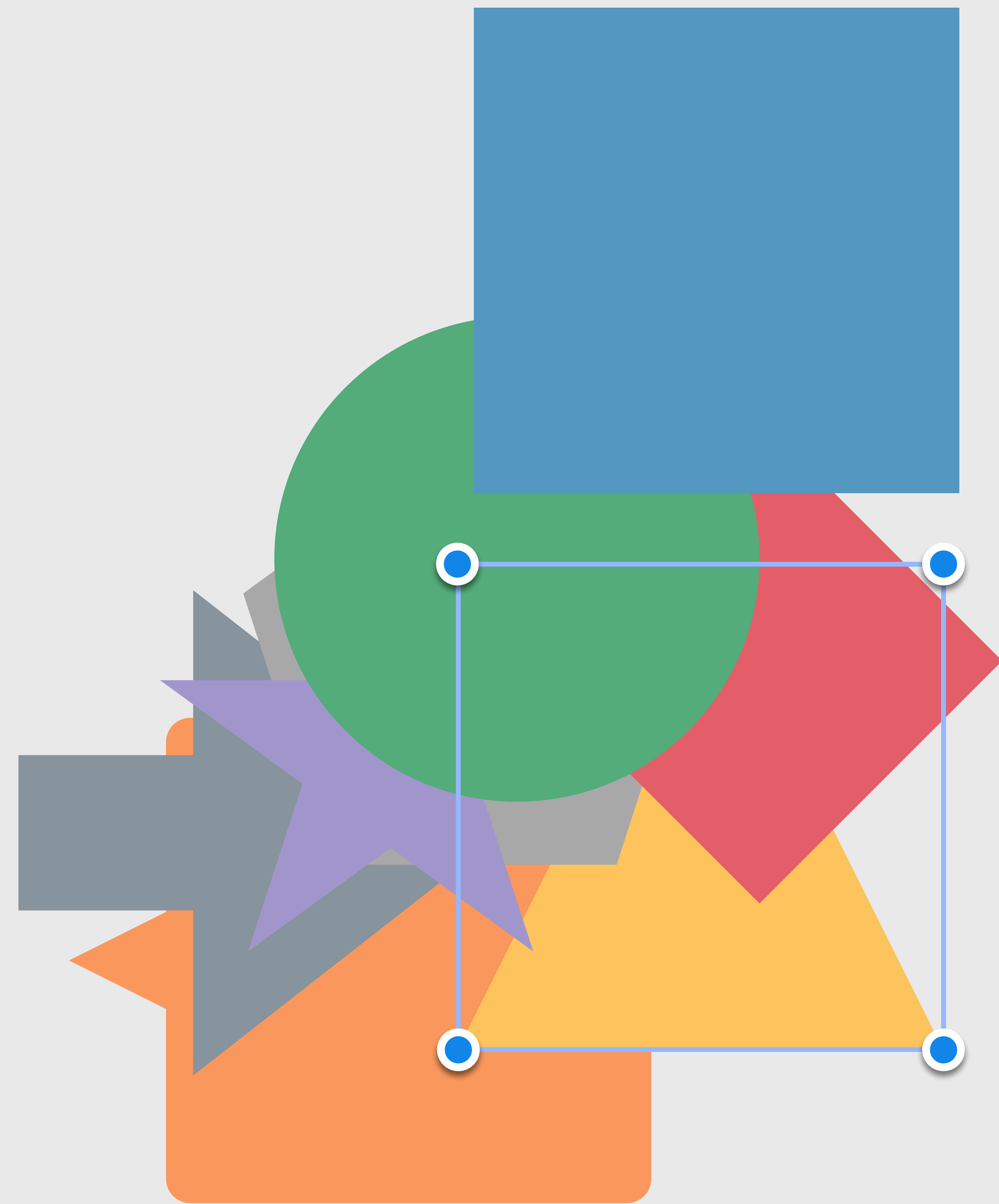


Bring to Front Send to Back

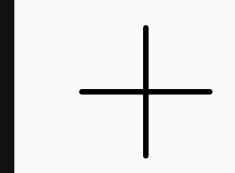
Shapes

Bring Forward Send Forward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Triangle
-  Arrow
-  Callout




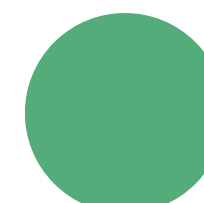
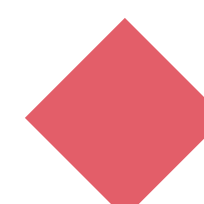
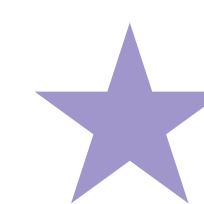
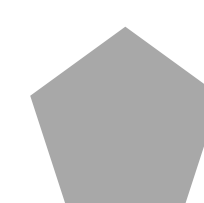

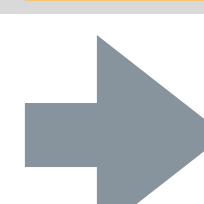



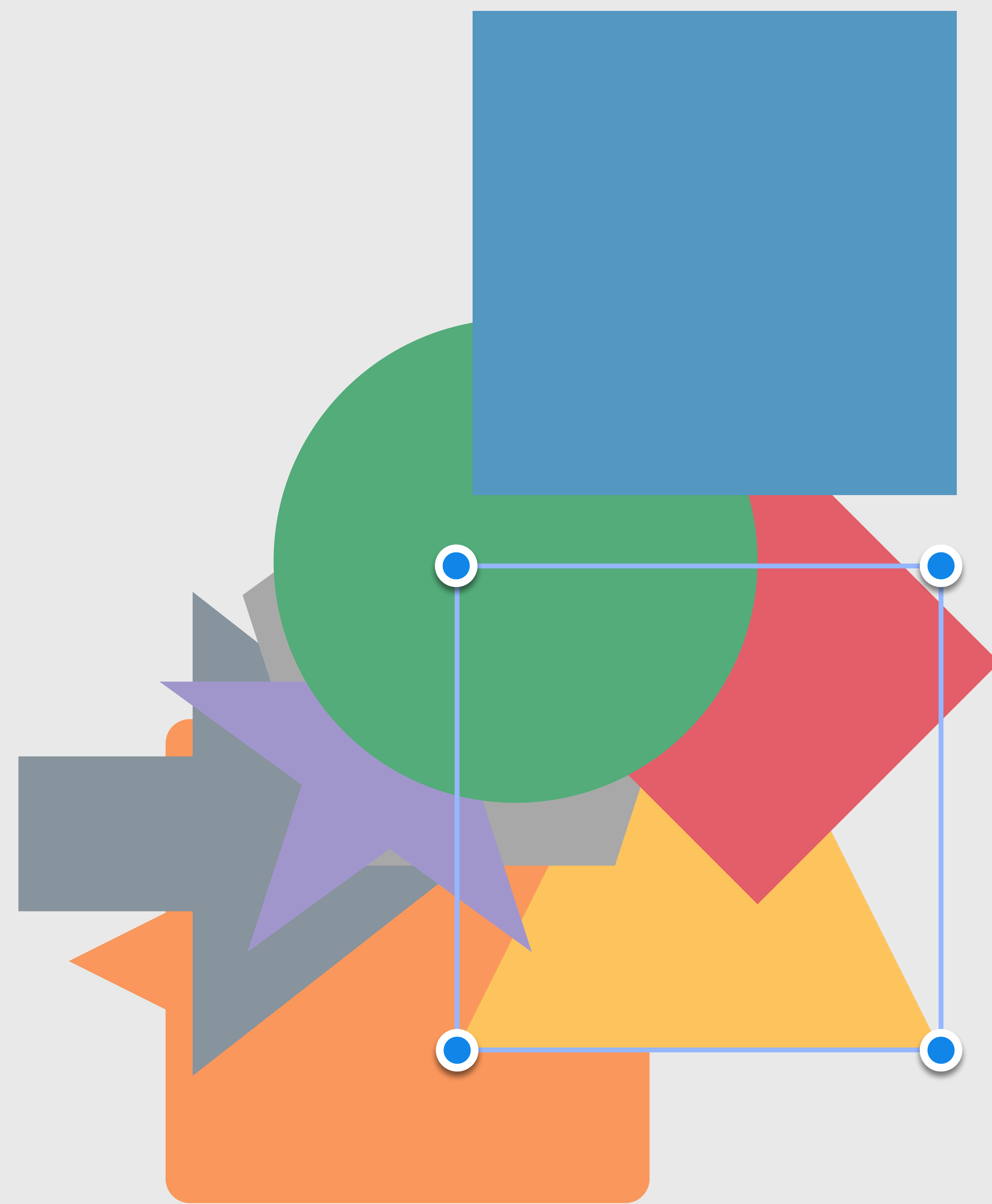


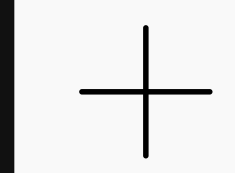
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Diamond
-  Star
-  Polygon
-  Triangle
-  Arrow
-  Callout



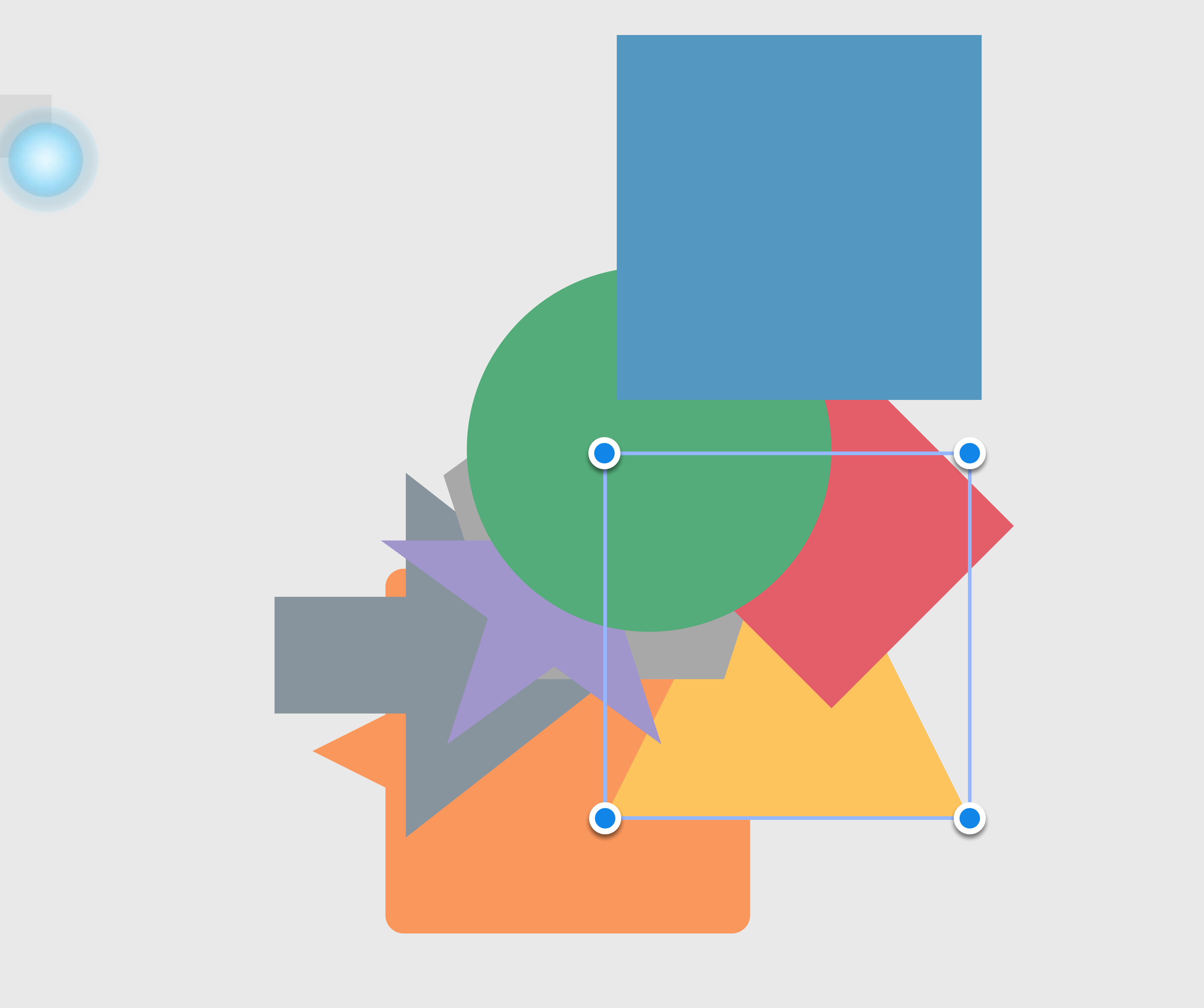


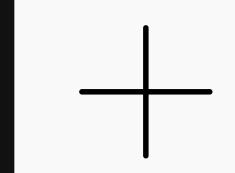
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Square
- Circle
- Triangle
- Diamond
- Star
- Polygon
- Arrow
- Callout



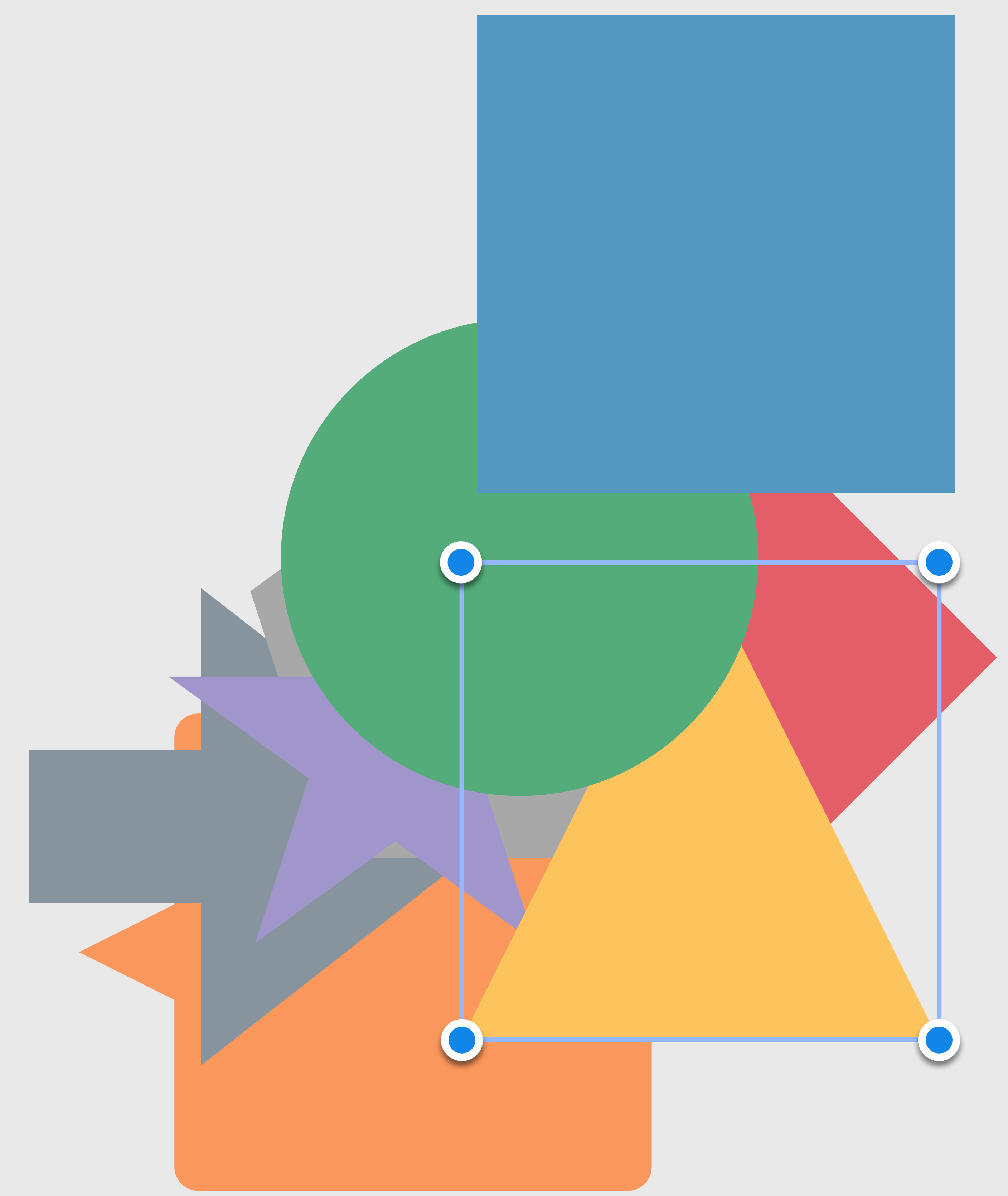
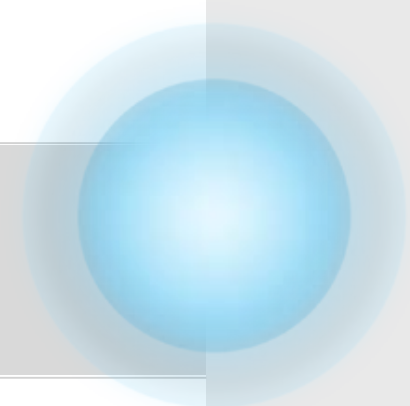


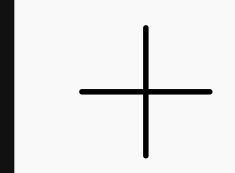
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Square
- Circle
- Triangle
- Diamond
- Star
- Polygon
- Arrow
- Callout


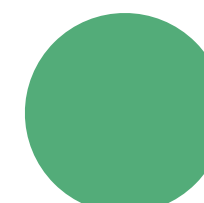

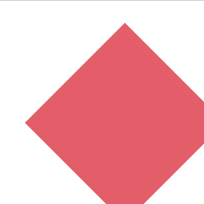

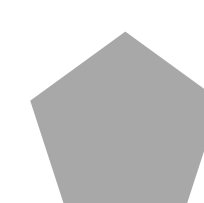
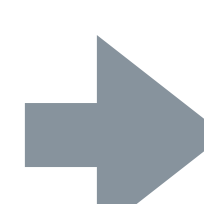



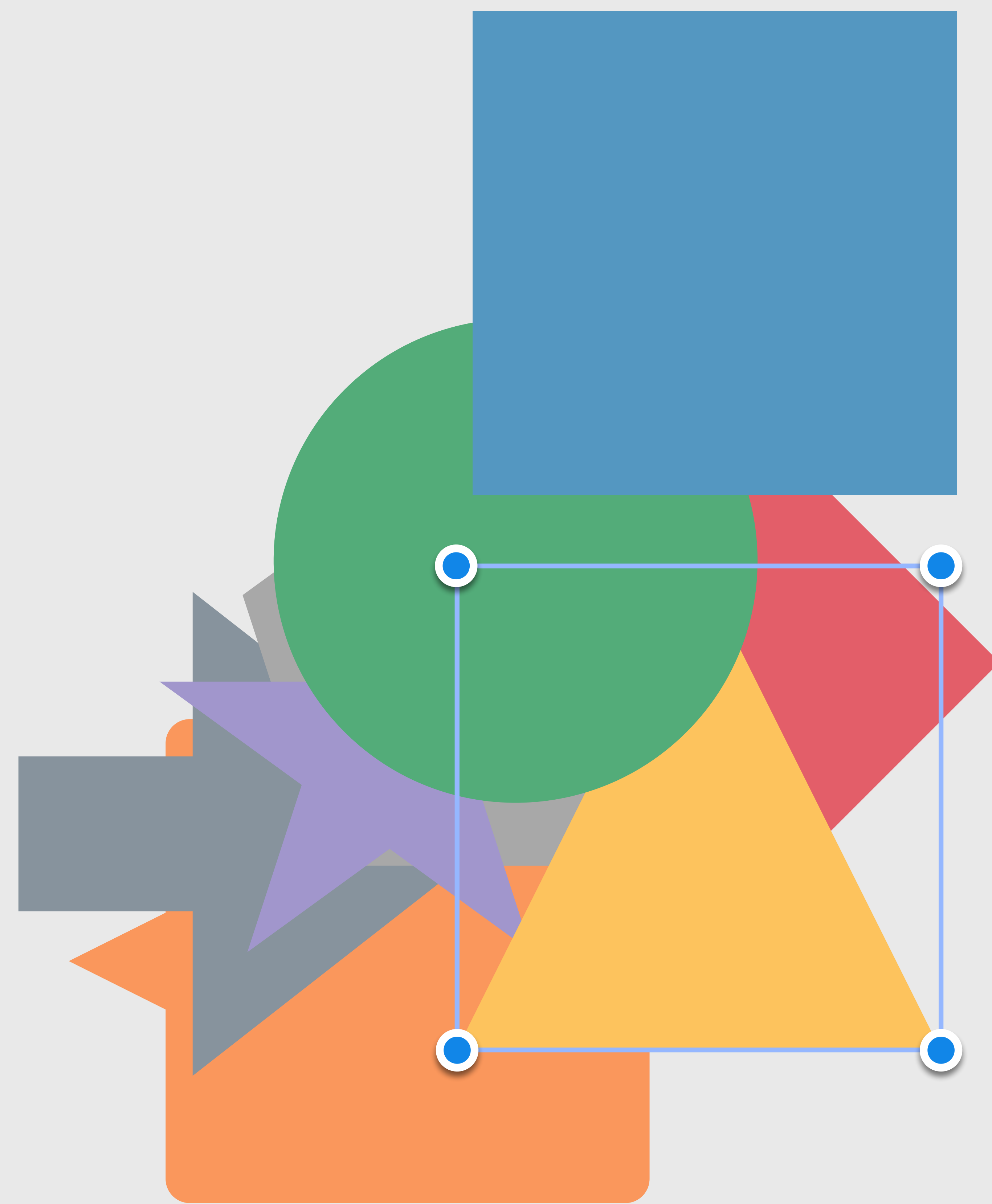


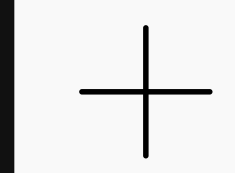
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Triangle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout



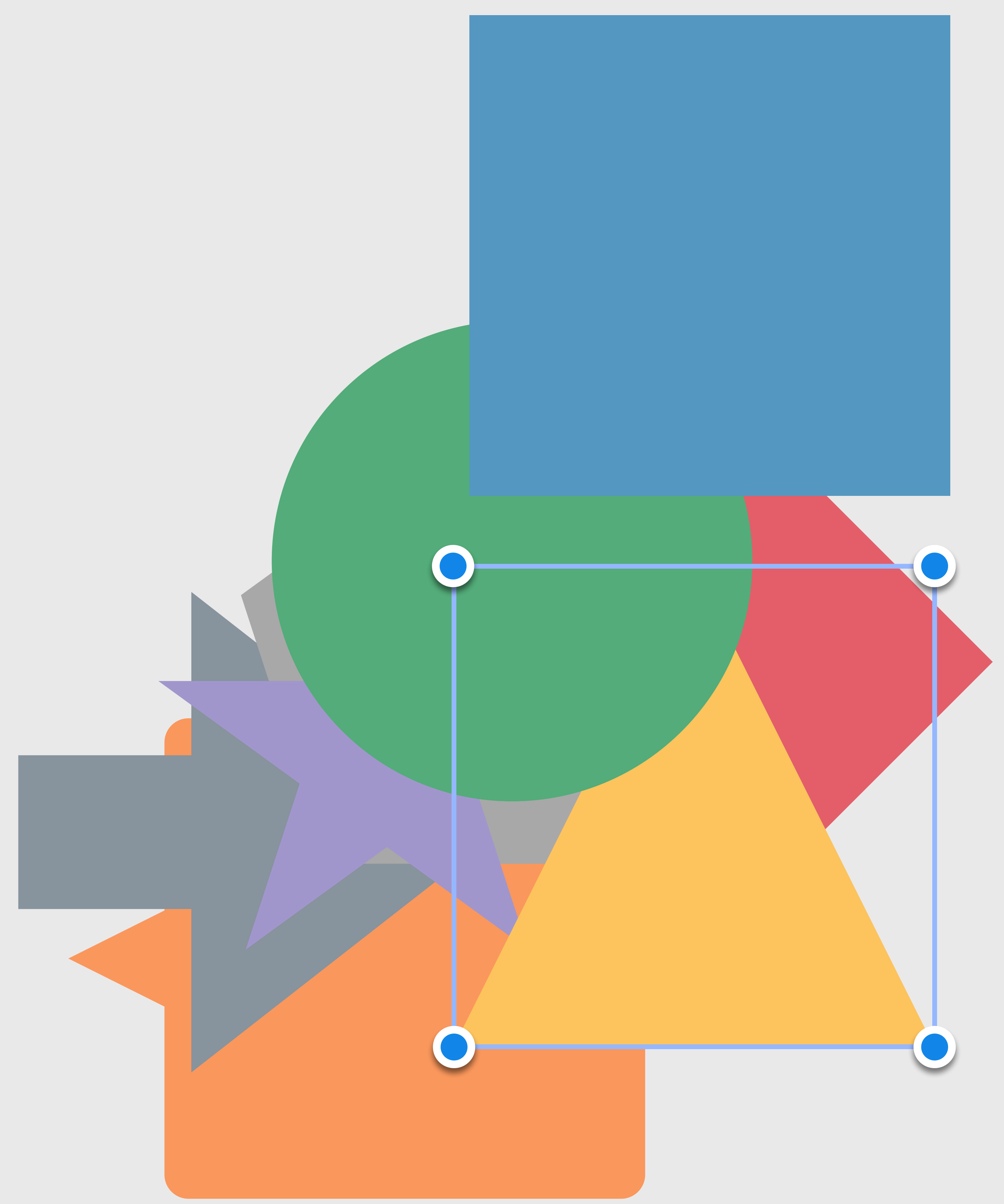


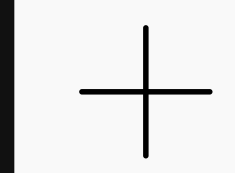
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Square
- Circle
- Triangle
- Diamond
- Star
- Polygon
- Arrow
- Callout


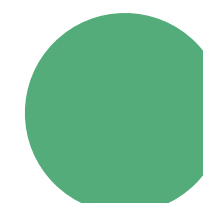

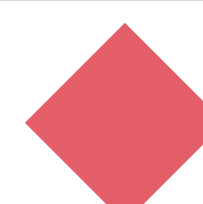

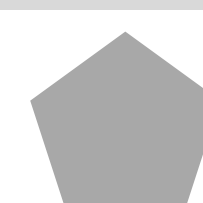
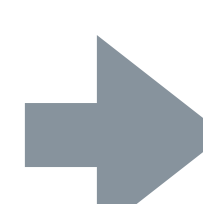



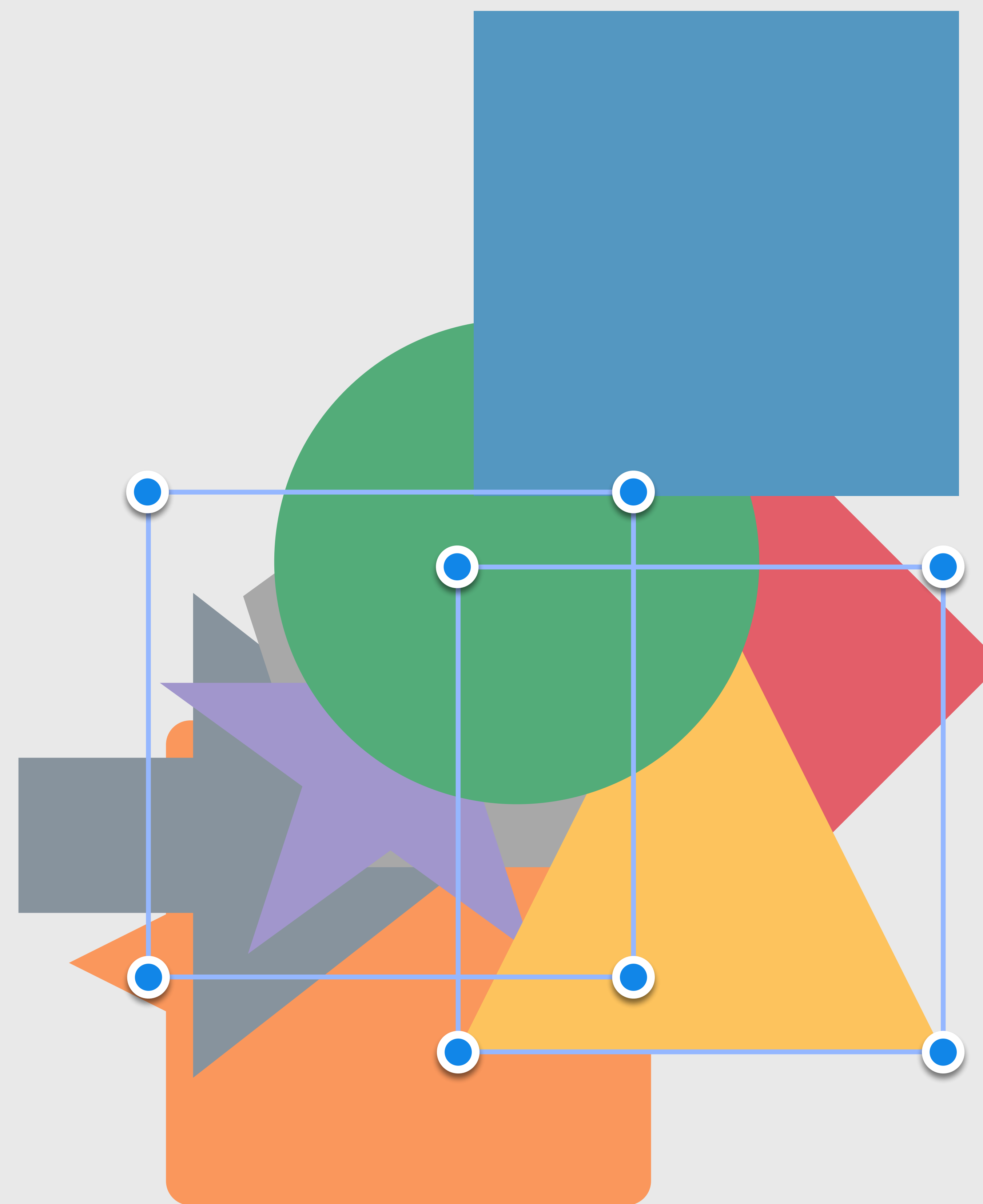


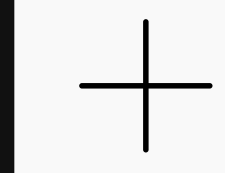
Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Triangle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout






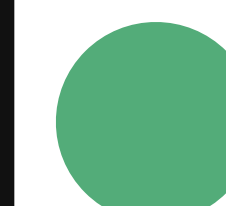

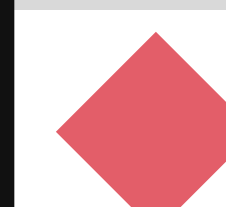
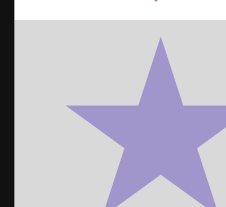

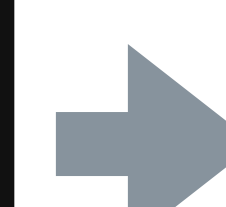

Bring to Front

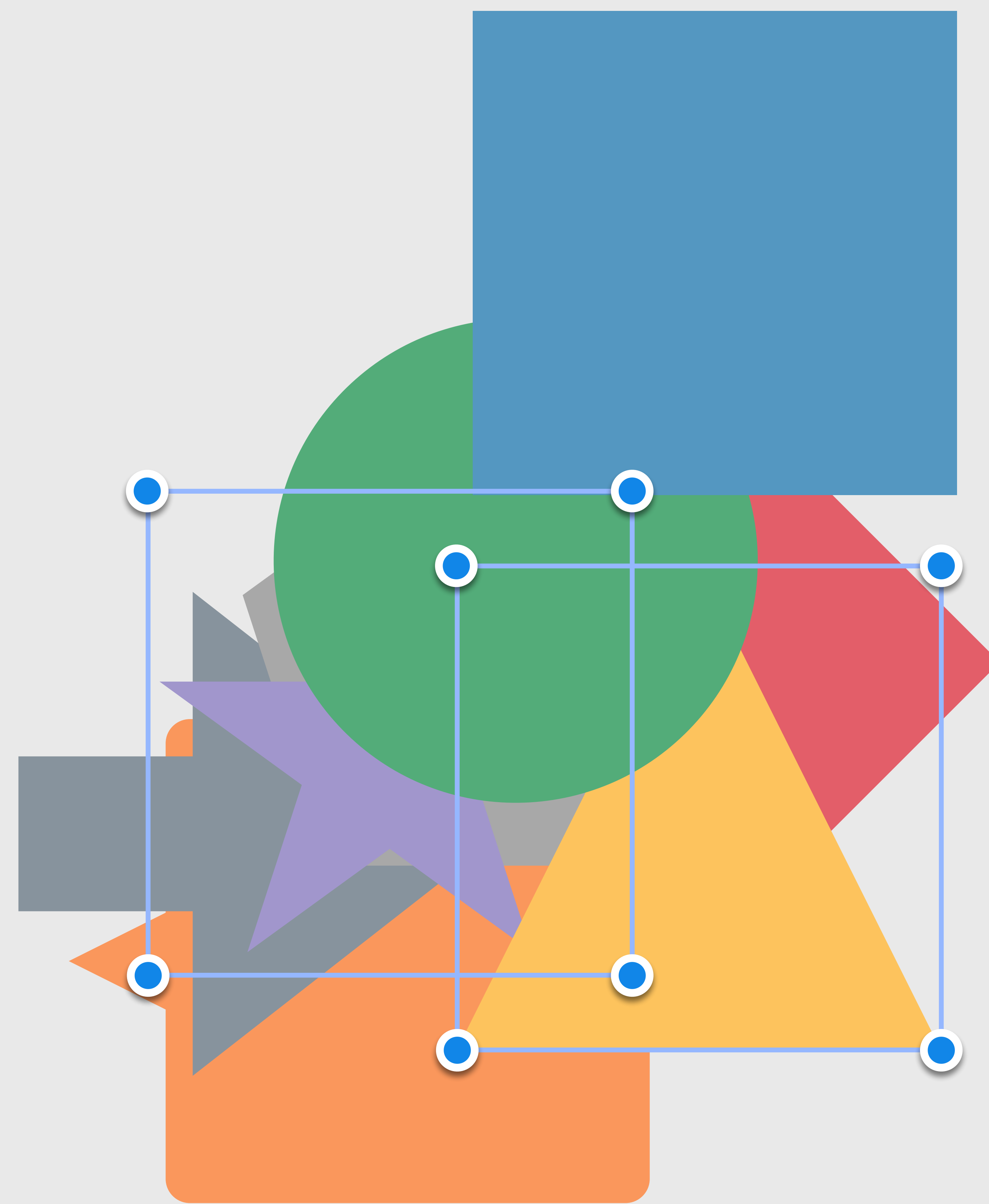
Send to Back

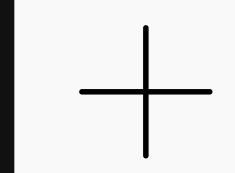
Shapes

Bring Forward

Send Backward

-  Square
-  Circle
-  Triangle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout



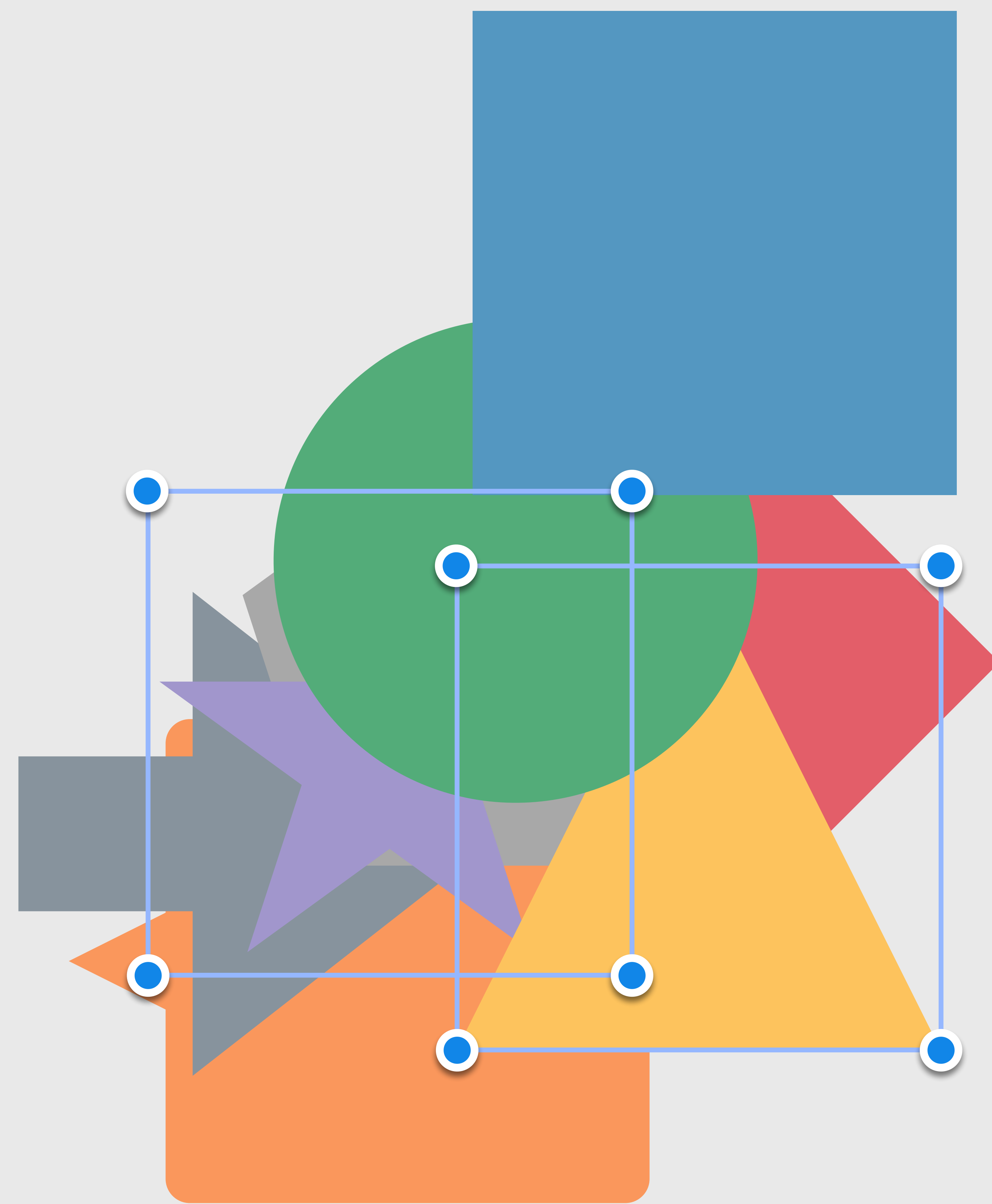


Bring to Front Send to Back

Shapes

Bring Forward Send Backward

-  Square
-  Circle
-  Triangle
-  Diamond
-  Star
-  Polygon
-  Arrow
-  Callout



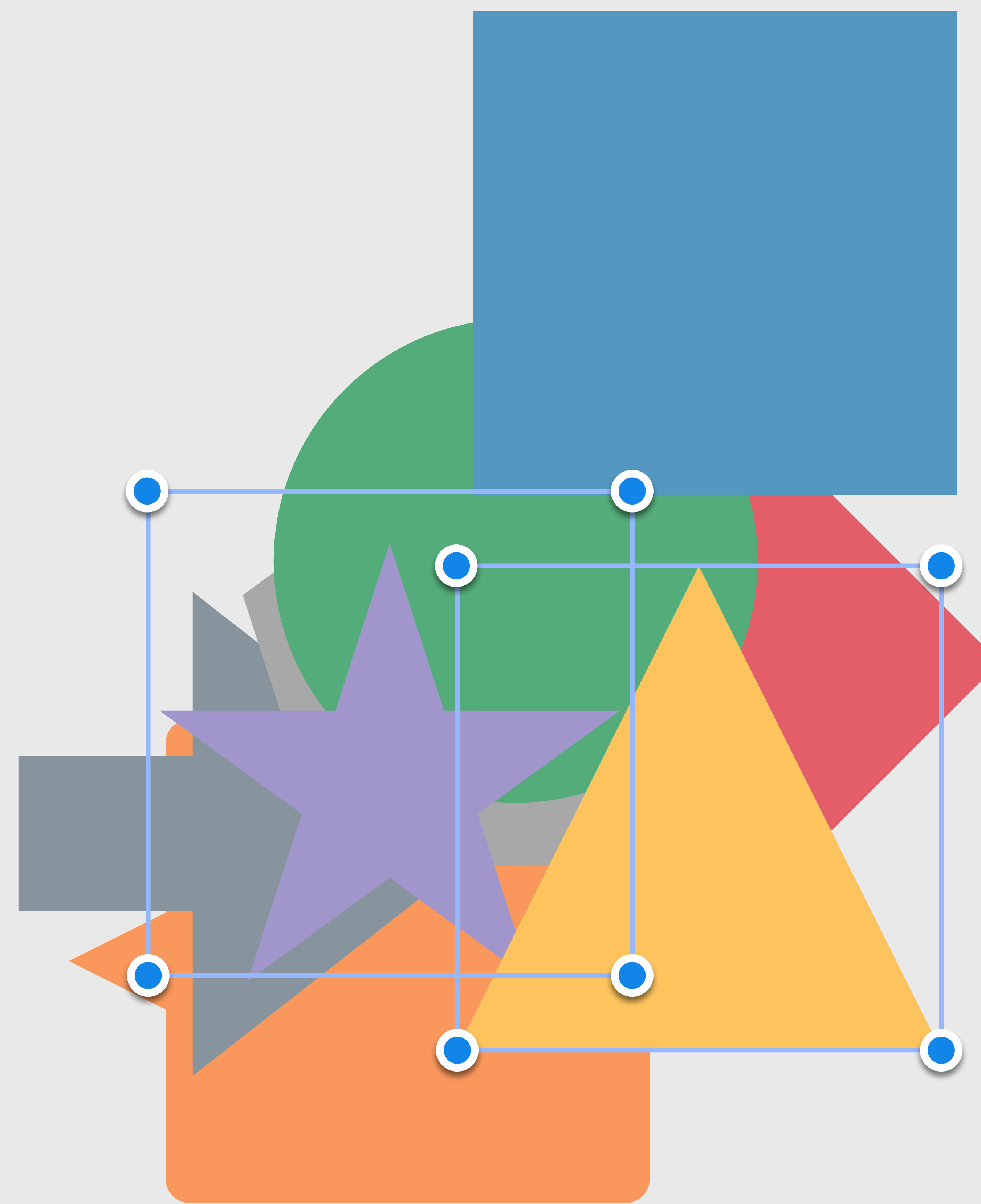


Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Square
- Triangle
- Star
- Circle
- Diamond
- Polygon
- Arrow
- Callout

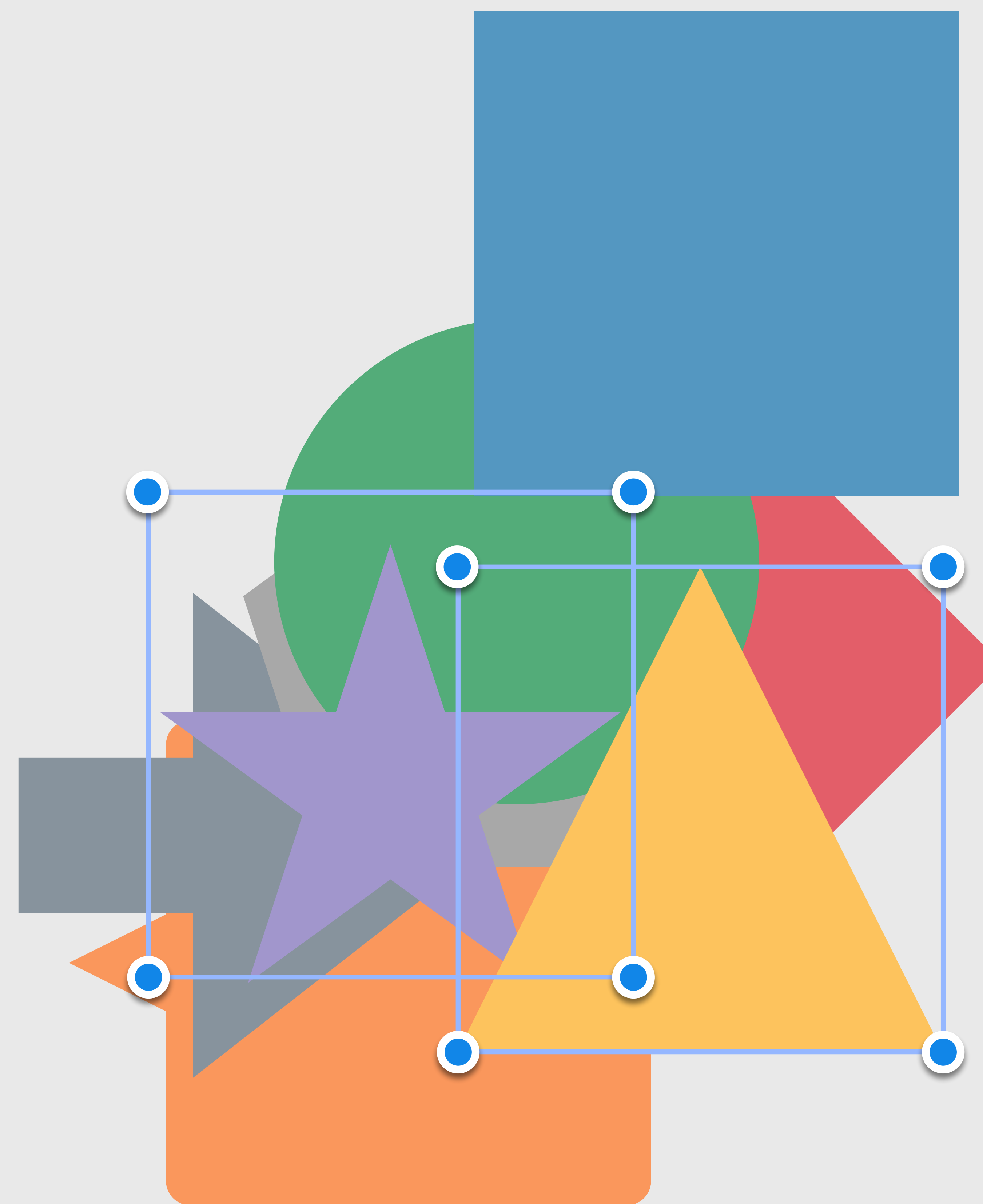


Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Square
- Triangle
- Star
- Circle
- Diamond
- Polygon
- Arrow
- Callout

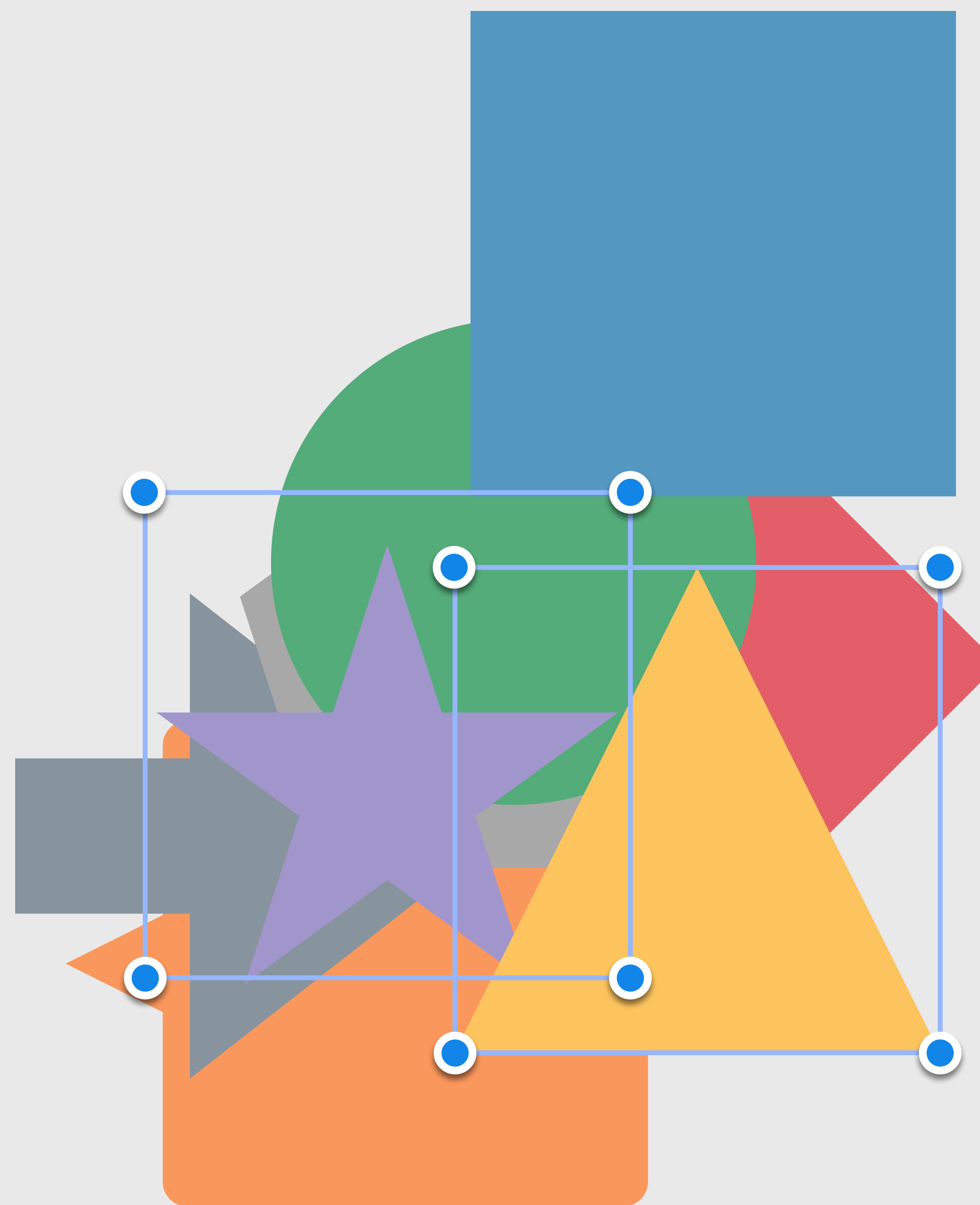


Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Square
- Triangle
- Star
- Circle
- Diamond
- Polygon
- Arrow
- Callout

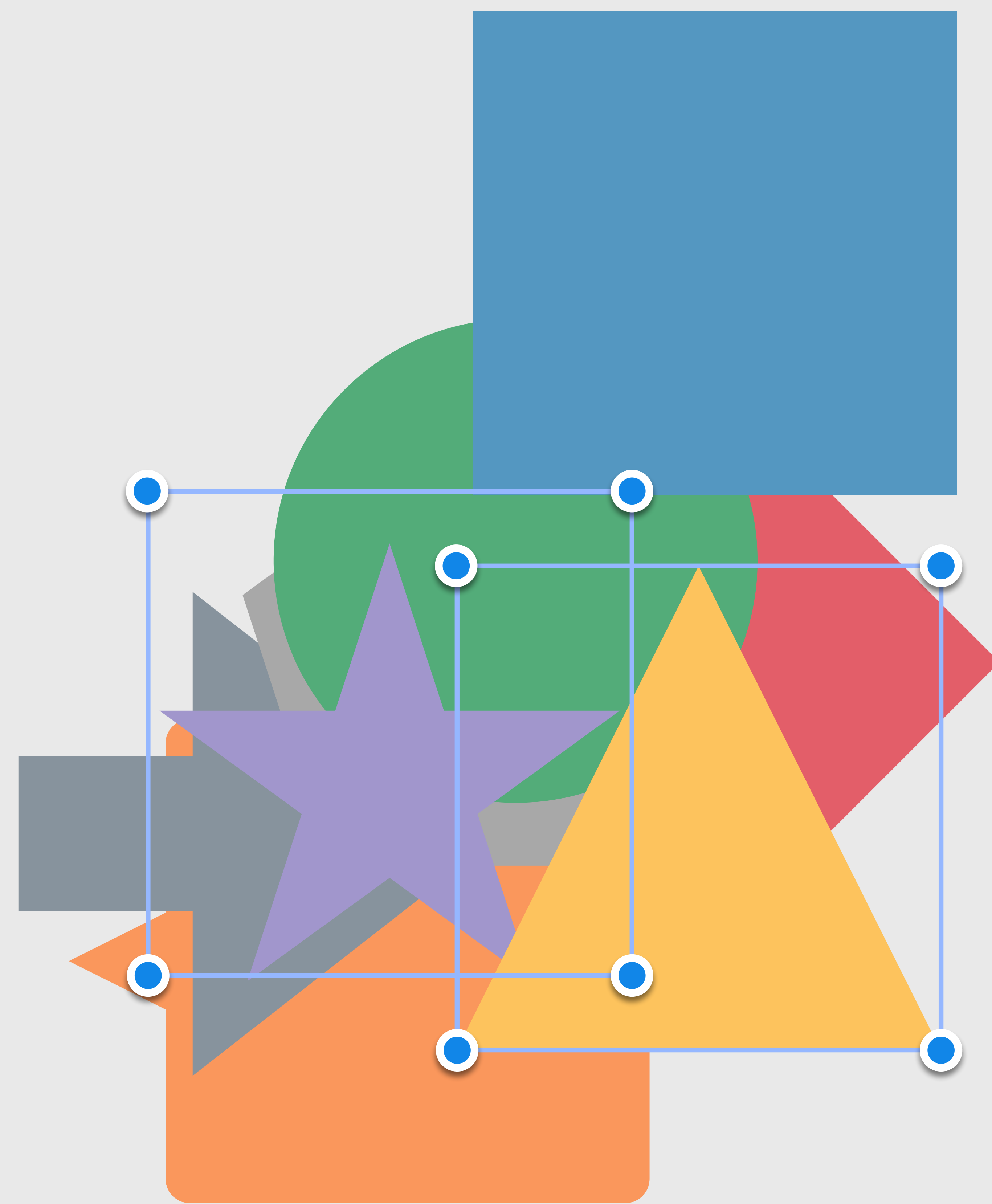


Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Triangle
- Star
- Square
- Circle
- Diamond
- Polygon
- Arrow
- Callout

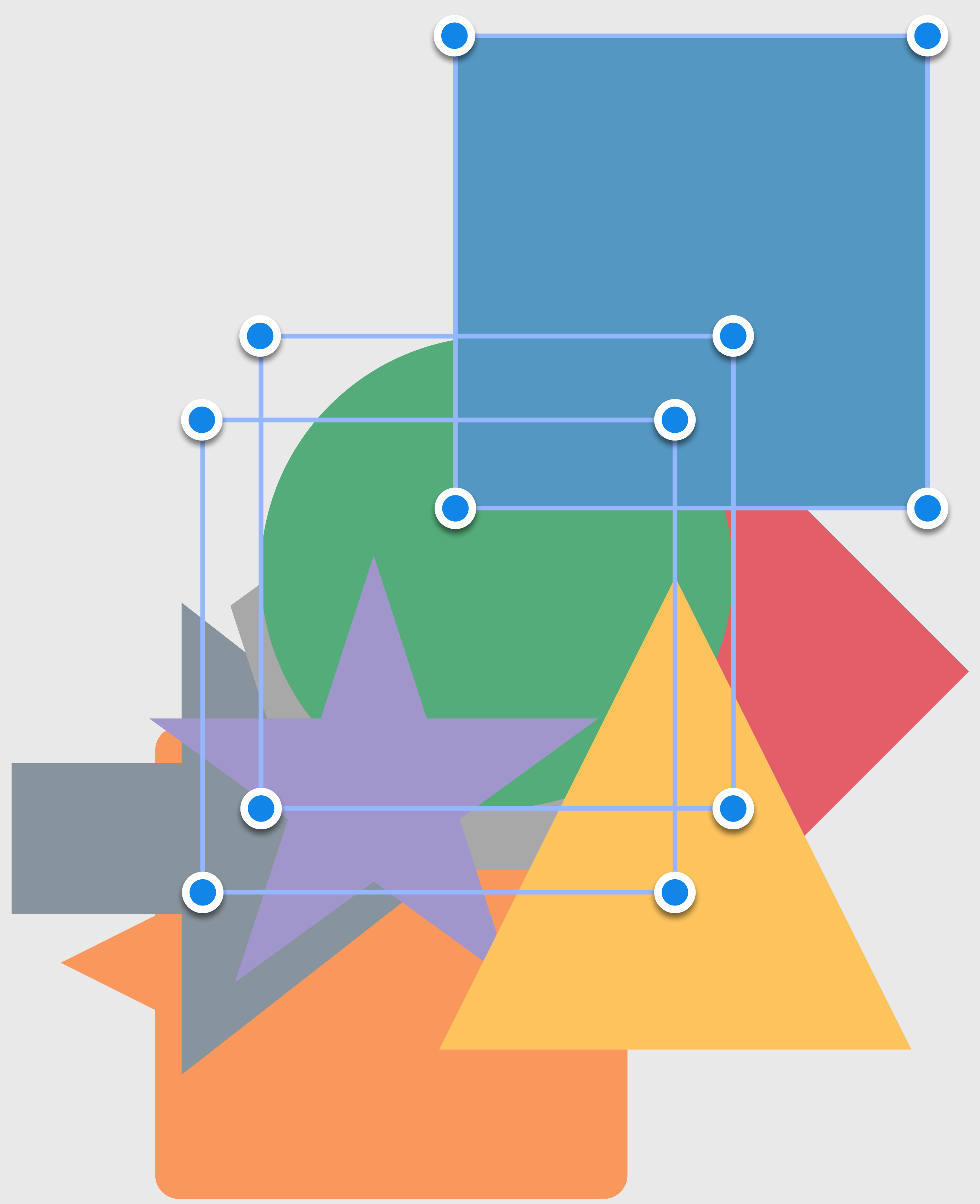


Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Triangle
- Star
- Square
- Circle
- Diamond
- Polygon
- Arrow
- Callout

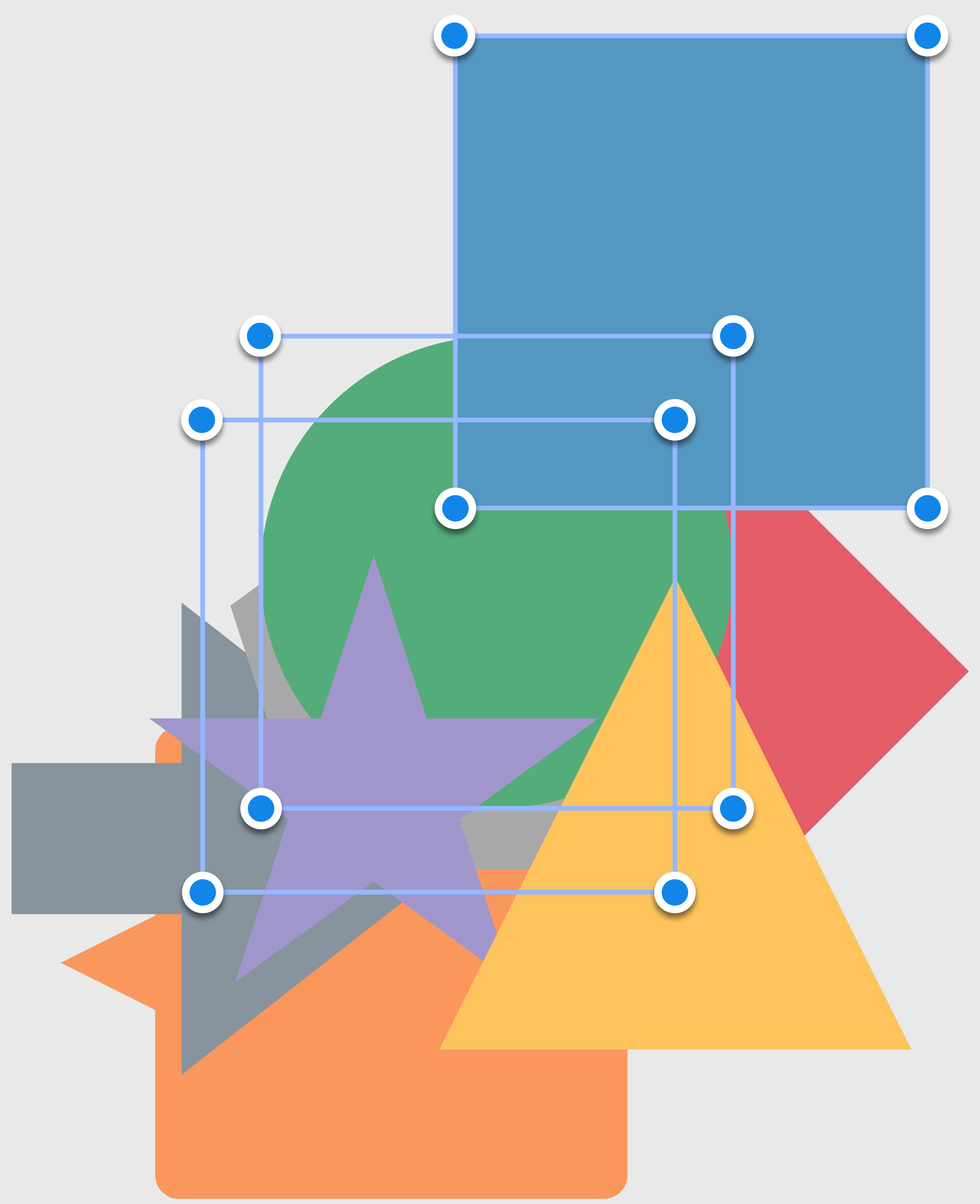


Bring to Front Send to Back

Shapes

Bring Forward Send Forward

- Triangle
- Star
- Square
- Circle
- Diamond
- Polygon
- Arrow
- Callout

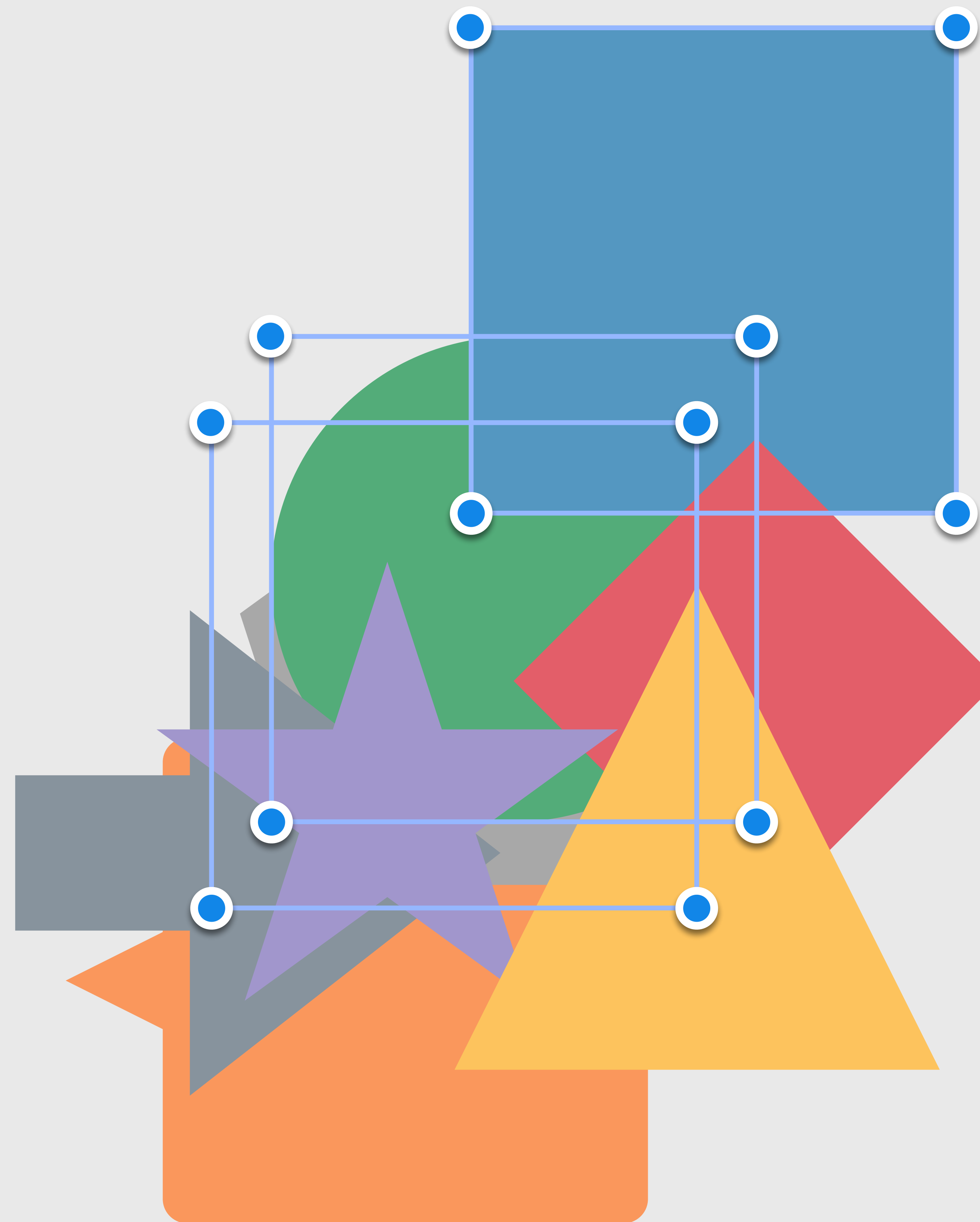


Bring to Front Send to Back

Shapes

Bring Forward Send Forward

- Triangle
- Star
- Diamond
- Arrow
- Square
- Circle
- Polygon
- Callout

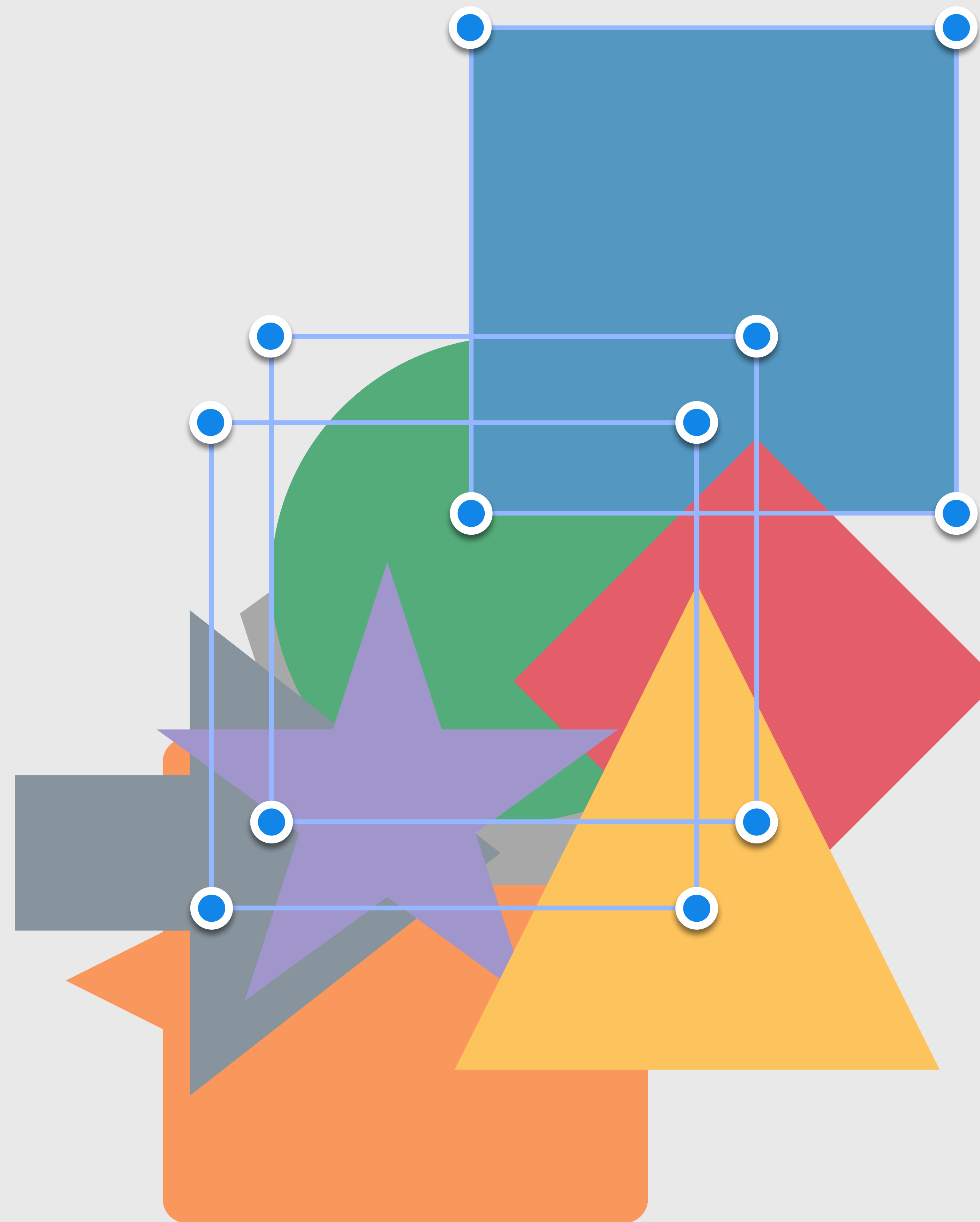


Bring to Front Send to Back

Shapes

Bring Forward Send Backward

- Triangle
- Star
- Diamond
- Arrow
- Square
- Circle
- Polygon
- Callout





```
// Layering Operations
// bringToFront()

extension Canvas {
  mutating func bringToFront() {
    var i = 0, j = 0
    while i < shapes.count {
      if shapes[i].isSelected {
        let selected = shapes.remove(at: i)
        shapes.insert(selected, at: j)
        j += 1
      }
      i += 1
    }
  }
}
```

```
// Layering Operations
// bringToFront()
```

```
extension Canvas {
  mutating func bringToFront() {
    var i = 0, j = 0
    while i < shapes.count {
      if shapes[i].isSelected {
        let selected = shapes.remove(at: i)
        shapes.insert(selected, at: j)
        j += 1
      }
      i += 1
    }
  }
}
```

O(n) loop, containing...

```
// Layering Operations
// bringToFront()
```

```
extension Canvas {
  mutating func bringToFront() {
    var i = 0, j = 0
    while i < shapes.count {
      if shapes[i].isSelected {
        let selected = shapes.remove(at: i)
        shapes.insert(selected, at: j)
        j += 1
      }
      i += 1
    }
  }
}
```

O(n) loop, containing...

O(n) operations

```
// Layering Operations
// bringToFront()
```

```
extension Canvas {
  mutating func bringToFront() {
    var i = 0, j = 0
    while i < shapes.count {
      if shapes[i].isSelected {
        let selected = shapes.remove(at: i)
        shapes.insert(selected, at: j)
        j += 1
      }
      i += 1
    }
  }
}
```

O(n) loop, containing...

O(n) operations

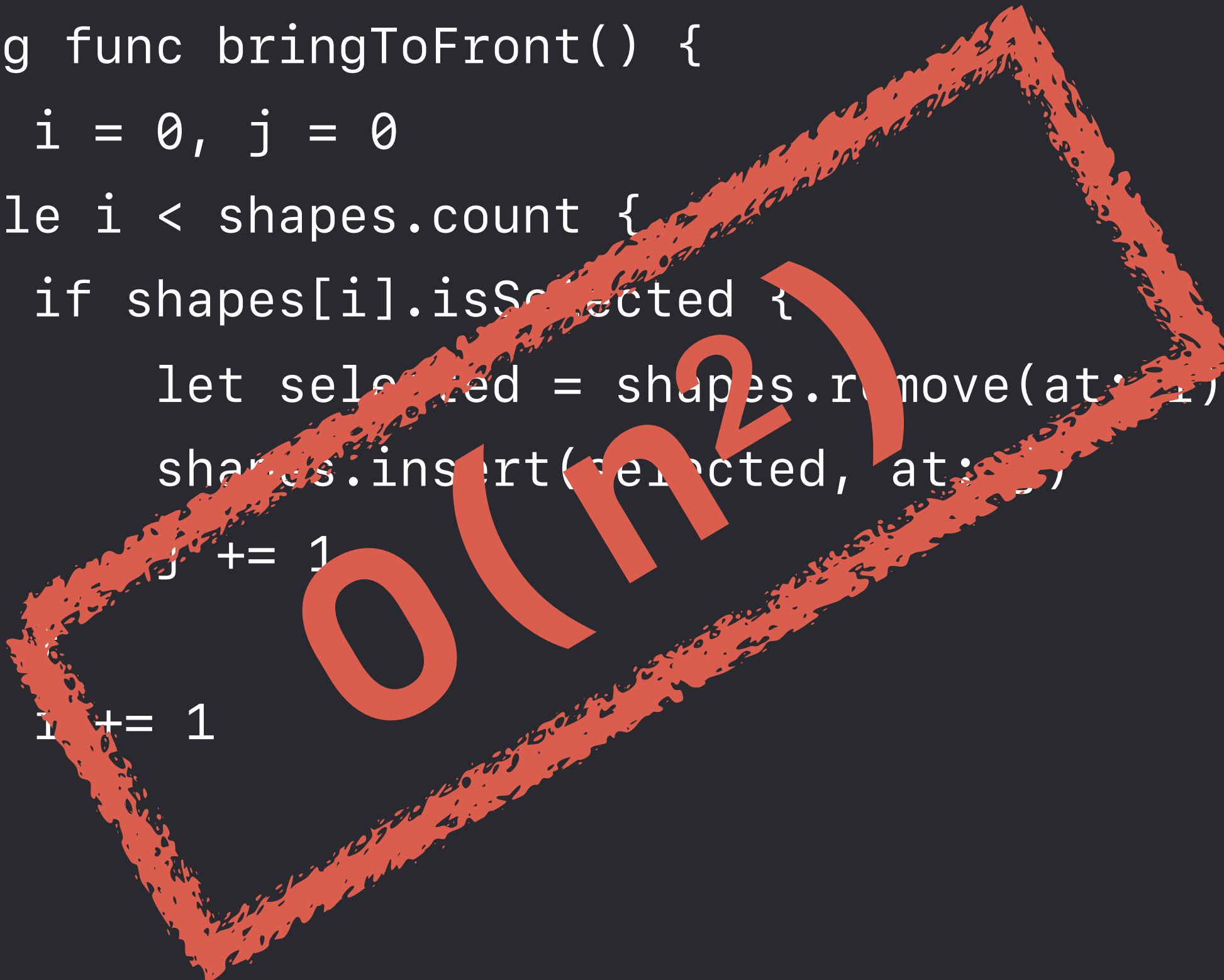
O(n<sup>2</sup>)

```

extension Canvas {
  mutating func bringToFront() {
    var i = 0, j = 0
    while i < shapes.count {
      if shapes[i].isSelected {
        let selected = shapes.remove(at: i)
        shapes.insert(selected, at: j)
        j += 1
      }
      i += 1
    }
  }

  mutating func sendToBack() {
    var i = 0, j = shapes.count
    while i < j {
      if shapes[i].isSelected {
        let selected = shapes.remove(at: i)
        shapes.append(selected)
        j -= 1
      }
      else {
        i += 1
      }
    }
  }
}

```



```

extension Canvas {
  mutating func sendBackward() {
    for i in shapes.indices.reversed()
    where shapes[i].isSelected
    {
      var insertionPoint = i + 1
      if insertionPoint == shapes.count { return }
      for j in (0...i).reversed()
      where shapes[j].isSelected
      {
        let x = shapes.remove(at: j)
        shapes.insert(x, at: insertionPoint)
        insertionPoint -= 1
      }
    }
    return
  }

  mutating func bringForward() {
    for i in shapes.indices
    where shapes[i].isSelected
    {
      if i == 0 { return }
      var insertionPoint = i - 1
      for j in i..<shapes.count
      where shapes[j].isSelected
      {
        let x = shapes.remove(at: j)
        shapes.insert(x, at: insertionPoint)
        insertionPoint += 1
      }
    }
    return
  }
}

```

```

extension Canvas {
  mutating func gatherSelected(at target: Int) {
    var shapesToInsert: [Shape] = []
    var insertionPoint = target
    var i = 0

    while i < insertionPoint {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        shapesToInsert.append(x)
        insertionPoint -= 1
      }
      else {
        i += 1
      }
    }

    while i < shapes.count {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        shapesToInsert.append(x)
      }
      else {
        i += 1
      }
    }

    shapes.insert(
      contentsOf: shapesToInsert,
      at: insertionPoint)
  }
}

```

```

extension Canvas {
  mutating func bringToFront() {
    var i = 0, j = 0
    while i < shapes.count {
      if shapes[i].isSelected {
        let selected = shapes.remove(at: i)
        shapes.insert(selected, at: j)
        j += 1
      }
      i += 1
    }
  }

  mutating func sendToBack() {
    var i = 0, j = shapes.count - 1
    while i < j {
      if shapes[i].isSelected {
        let selected = shapes.remove(at: i)
        shapes.append(selected)
        j -= 1
      } else {
        i += 1
      }
    }
  }
}

```

$O(n^2)$

$O(n^2)$

```

extension Canvas {
  mutating func sendBackward() {
    for i in shapes.indices.reversed() {
      where shapes[i].isSelected {
        var insertionPoint = i + 1
        if insertionPoint == shapes.count { return }
        for j in 0..i.reversed() {
          where shapes[j].isSelected {
            let x = shapes.remove(at: j)
            shapes.insert(x, at: insertionPoint)
            insertionPoint -= 1
          }
        }
        return
      }
    }

    mutating func bringForward() {
      for i in shapes.indices {
        where shapes[i].isSelected {
          if i == 0 { return }
          var insertionPoint = i - 1
          for j in i..shapes.count {
            where shapes[j].isSelected {
              let x = shapes.remove(at: j)
              shapes.insert(x, at: insertionPoint)
              insertionPoint += 1
            }
          }
          return
        }
      }
    }
  }
}

```

$O(n^2)$

$O(n^2)$

```

extension Canvas {
  mutating func gatherSelected(at target: Int) {
    var shapesToInsert: [Shape] = []
    var insertionPoint = target
    var i = 0

    while i < insertionPoint {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        shapesToInsert.append(x)
        insertionPoint -= 1
      } else {
        i += 1
      }
    }

    while i < shapes.count {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        shapesToInsert.append(x)
      } else {
        i += 1
      }
    }

    shapes.insert(
      contentsOf: shapesToInsert,
      at: insertionPoint)
  }
}

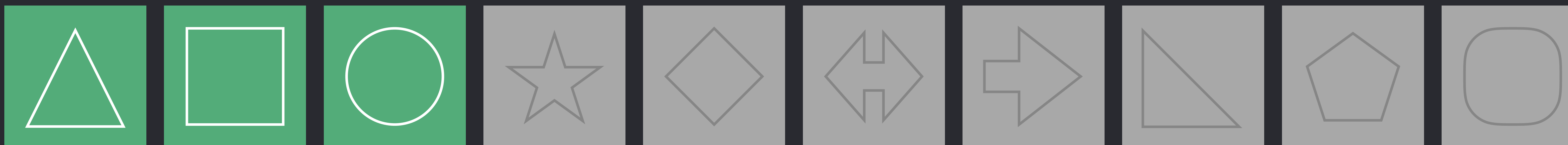
```

$O(n^2)$

```
extension Canvas {  
  mutating func bringToFront() {  
    var i = 0, j = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        let selected = shapes.remove(at: i)  
        shapes.insert(selected, at: j)  
        j += 1  
      }  
      i += 1  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func bringToFront() {  
    var i = 0, j = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        let selected = shapes.remove(at: i)  
        shapes.insert(selected, at: j)  
        j += 1  
      }  
      i += 1  
    }  
  }  
}
```





```
extension Canvas {  
  
  mutating func bringToFront() {  
    var i = 0, j = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        let selected = shapes.remove(at: i)  
        shapes.insert(selected, at: j)  
        j += 1  
      }  
      i += 1  
    }  
  }  
}
```



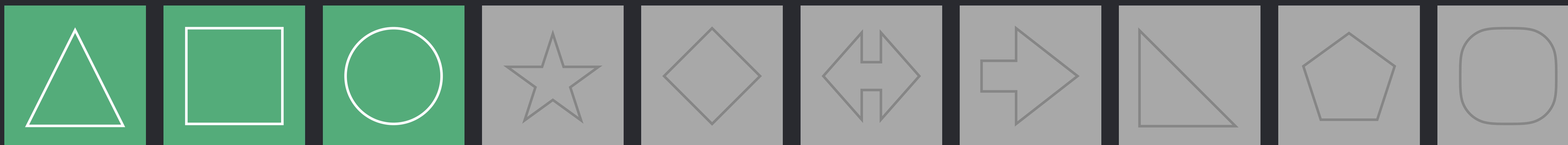
```
extension Canvas {  
  /// Moves the selected shapes to the front, maintaining their relative order.  
  mutating func bringToFront() {  
    var i = 0, j = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        let selected = shapes.remove(at: i)  
        shapes.insert(selected, at: j)  
        j += 1  
      }  
      i += 1  
    }  
  }  
}
```



```
extension Canvas {  
  /// Moves the selected shapes to the front, maintaining their relative order.  
  mutating func bringToFront() {  
    var i = 0, j = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        let selected = shapes.remove(at: i)  
        shapes.insert(selected, at: j)  
        j += 1  
      }  
      i += 1  
    }  
  }  
}
```



```
extension Canvas {  
  /// Moves the selected shapes to the front, maintaining their relative order.  
  mutating func bringToFront() {  
    var i = 0, j = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        let selected = shapes.remove(at: i)  
        shapes.insert(selected, at: j)  
        j += 1  
      }  
      i += 1  
    }  
  }  
}
```



GitHub, Inc.

Features Business Explore Marketplace Pricing This repository Search Sign in or Sign up

apple / swift Watch 2,641 Star 44,011 Fork 6,936

Code Pull requests 328 Insights

Branch: master swift / test / Prototypes / Algorithms.swift Find file Copy path

natecook1000 [test] Update prototypes to use conditional conformance (#14274) 3543625 on Feb 1

3 contributors

780 lines (679 sloc) 23.3 KB Raw Blame History

```
1 //===--- Algorithms.swift.gyb -----*- swift -*-===//
2 //
3 // This source file is part of the Swift.org open source project
4 //
5 // Copyright (c) 2014 - 2017 Apple Inc. and the Swift project authors
6 // Licensed under Apache License v2.0 with Runtime Library Exception
7 //
8 // See https://swift.org/LICENSE.txt for license information
```

<https://github.com/apple/swift/blob/master/test/Prototypes/Algorithms.swift>

```
extension Canvas {  
  /// Moves the selected shapes to the front, maintaining their relative order.  
  mutating func bringToFront() {  
    var i = 0, j = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        let selected = shapes.remove(at: i)  
        shapes.insert(selected, at: j)  
        j += 1  
      }  
      i += 1  
    }  
  }  
}
```



```
extension Canvas {  
  /// Moves the selected shapes to the front, maintaining their relative order.  
  mutating func bringToFront() {  
    var i = 0, j = 0  
    while i < shapes.count {  
      if shapes[i].isSelected {  
        let selected = shapes.remove(at: i)  
        shapes.insert(selected, at: j)  
        j += 1  
      }  
      i += 1  
    }  
  }  
}
```



```
extension Canvas {  
  /// Moves the selected shapes to the front, maintaining their relative order.  
  mutating func bringToFront() {  
      
  }  
}
```





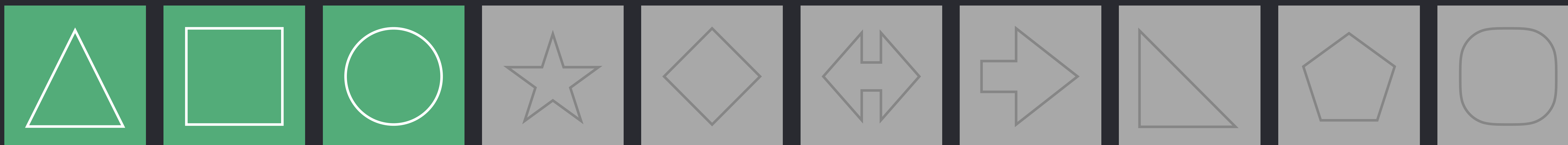
```
extension Canvas {  
  /// Moves the selected shapes to the front, maintaining their relative order.  
  mutating func bringToFront() {  
    shapes.stablePartition(isSuffixElement: (Shape) -> Bool)  
  }  
}
```



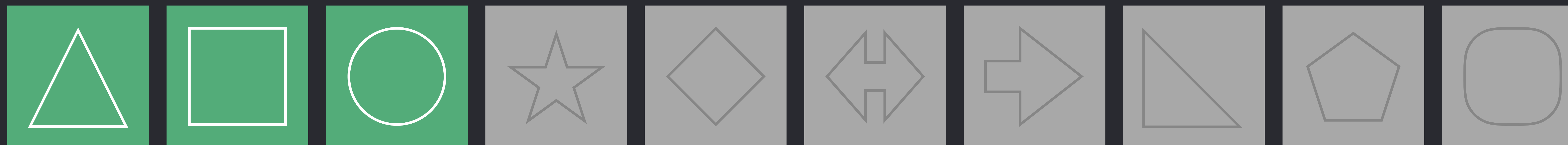
```
extension Canvas {  
  /// Moves the selected shapes to the front, maintaining their relative order.  
  mutating func bringToFront() {  
    shapes.stablePartition(isSuffixElement: (Shape) -> Bool)  
  }  
}
```



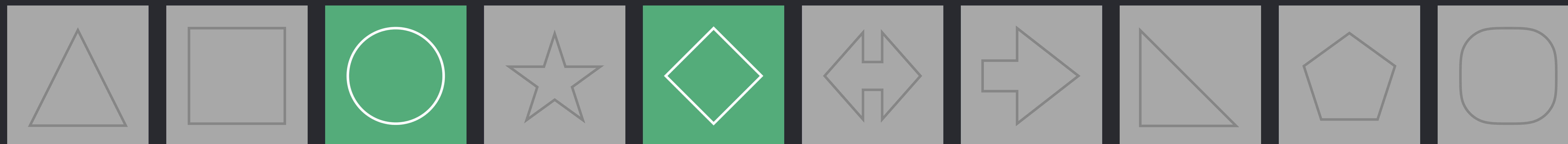
```
extension Canvas {  
  /// Moves the selected shapes to the front, maintaining their relative order.  
  mutating func bringToFront() {  
    shapes.stablePartition(isSuffixElement: (Shape) -> Bool)  
  }  
}
```



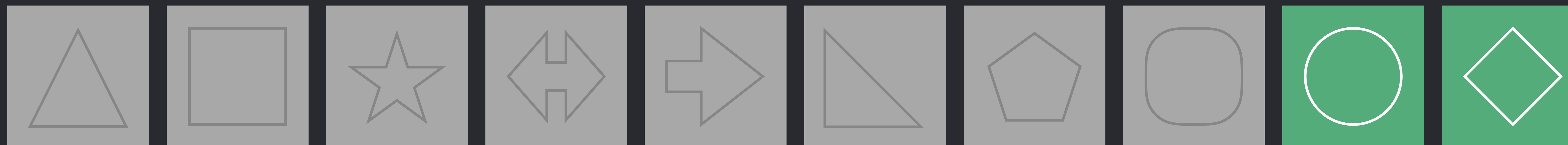
```
extension Canvas {  
    /// Moves the selected shapes to the front, maintaining their relative order.  
    mutating func bringToFront() {  
        shapes.stablePartition(isSuffixElement: { !$0.isSelected })  
    }  
}
```



```
extension Canvas {  
    /// Moves the selected shapes to the front, maintaining their relative order.  
    mutating func bringToFront() {  
        shapes.stablePartition(isSuffixElement: { !$0.isSelected })  
    }  
  
    /// Moves the selected shapes to the back, maintaining their relative order.  
    mutating func sendToBack() {  
        shapes.stablePartition(isSuffixElement: { $0.isSelected })  
    }  
}
```



```
extension Canvas {  
    /// Moves the selected shapes to the front, maintaining their relative order.  
    mutating func bringToFront() {  
        shapes.stablePartition(isSuffixElement: { !$0.isSelected })  
    }  
  
    /// Moves the selected shapes to the back, maintaining their relative order.  
    mutating func sendToBack() {  
        shapes.stablePartition(isSuffixElement: { $0.isSelected })  
    }  
}
```



```
extension Canvas {  
    /// Moves the selected shapes to the front, maintaining their relative order.  
    mutating func bringToFront() {  
        shapes.stablePartition(isSuffixElement: { !$0.isSelected })  
    }  
  
    /// Moves the selected shapes to the back, maintaining their relative order.  
    mutating func sendToBack() {  
        shapes.stablePartition(isSuffixElement: { $0.isSelected })  
    }  
}
```

```
extension Canvas {  
    /// Moves the selected shapes to the front, maintaining their relative order.  
    mutating func bringToFront() {  
        shapes.stablePartition(isSuffixElement: { !$0.isSelected })  
    }  
}
```

### Summary

Moves all elements satisfying `isSuffixElement` into a suffix of the collection, preserving their relative order, and returns the start of the resulting suffix.

### Declaration

```
mutating func stablePartition(isSuffixElement: (Element) -> Bool)  
-> Index
```

### Discussion

### Complexity

$O(n \log n)$  where  $n$  is the number of elements.

### Declared In

[Example.playground](#)



```
extension Canvas {  
    /// Moves the selected shapes to the front, maintaining their relative order.  
    mutating func bringToFront() {  
        shapes.stablePartition(isSuffixElement: { !$0.isSelected })  
    }  
}
```

### Summary

Moves all elements satisfying `isSuffixElement` into a suffix of the collection, preserving their relative order, and returns the start of the resulting suffix.

### Declaration

```
mutating func stablePartition(isSuffixElement: (Element) -> Bool)  
-> Index
```

### Discussion

#### Complexity

$O(n \log n)$  where  $n$  is the number of elements.

### Declared In

[Example.playground](#)

```
/// Moves the selected shapes to the back, maintaining their relative order.
```

## Summary

Moves all elements satisfying `isSuffixElement` into a suffix of the collection, preserving their relative order, and returns the start of the resulting suffix.

## Declaration

```
mutating func stablePartition(isSuffixElement: (Element) -> Bool)  
-> Index
```

## Discussion

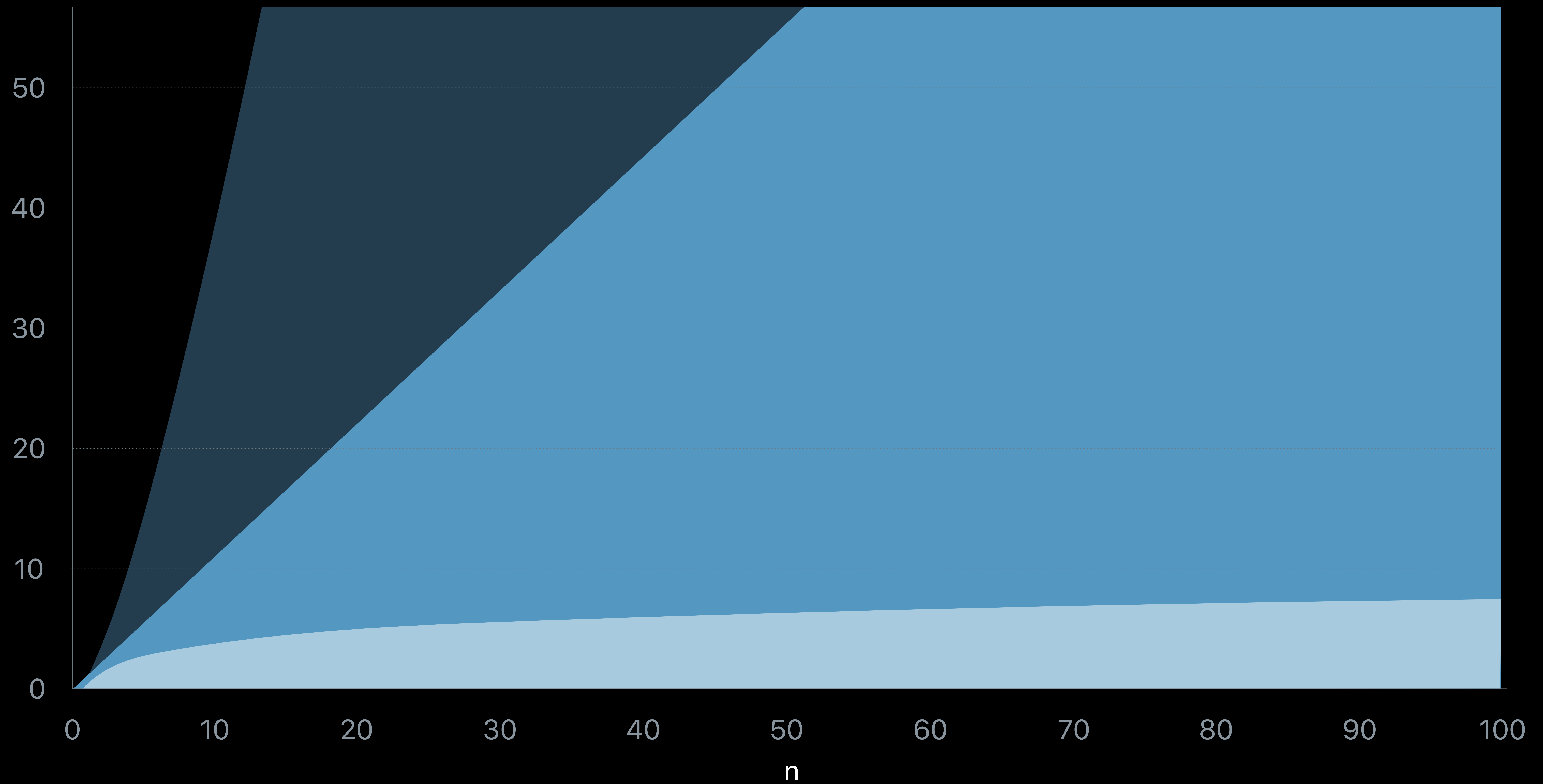
### Complexity

$O(n \log n)$  where  $n$  is the number of elements.

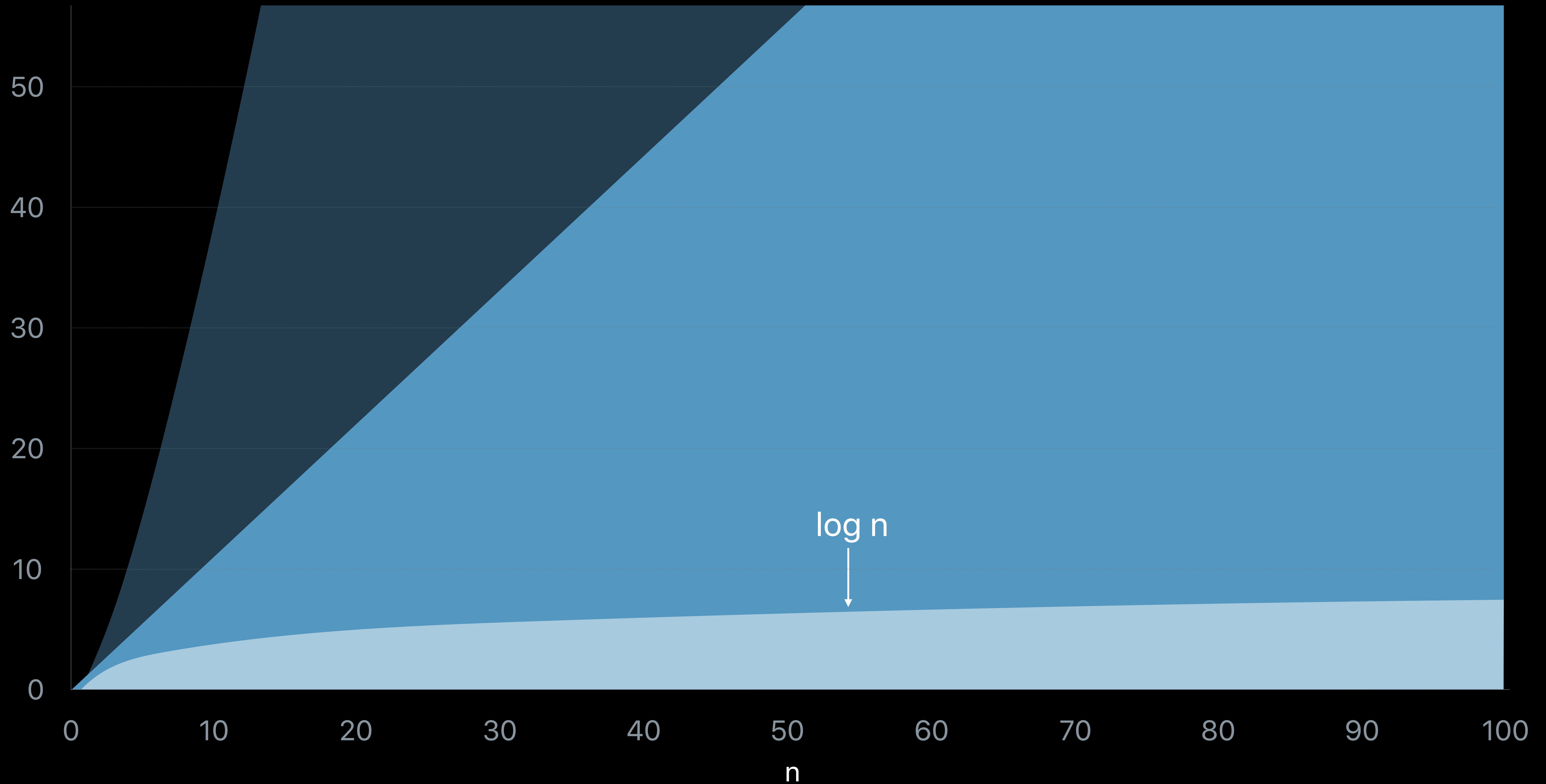
## Declared In

[Example.playground](#)

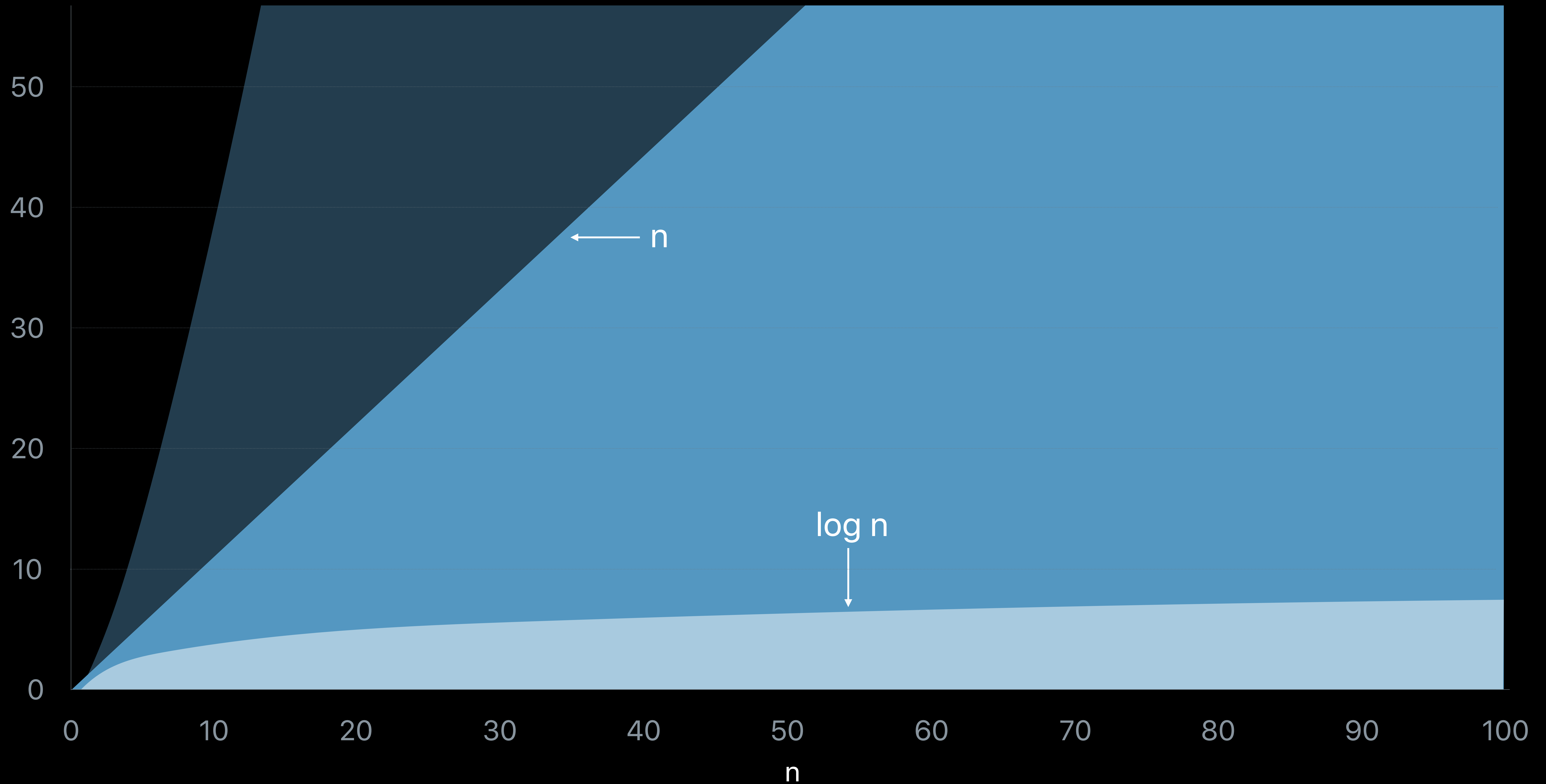
# Scalability Redux



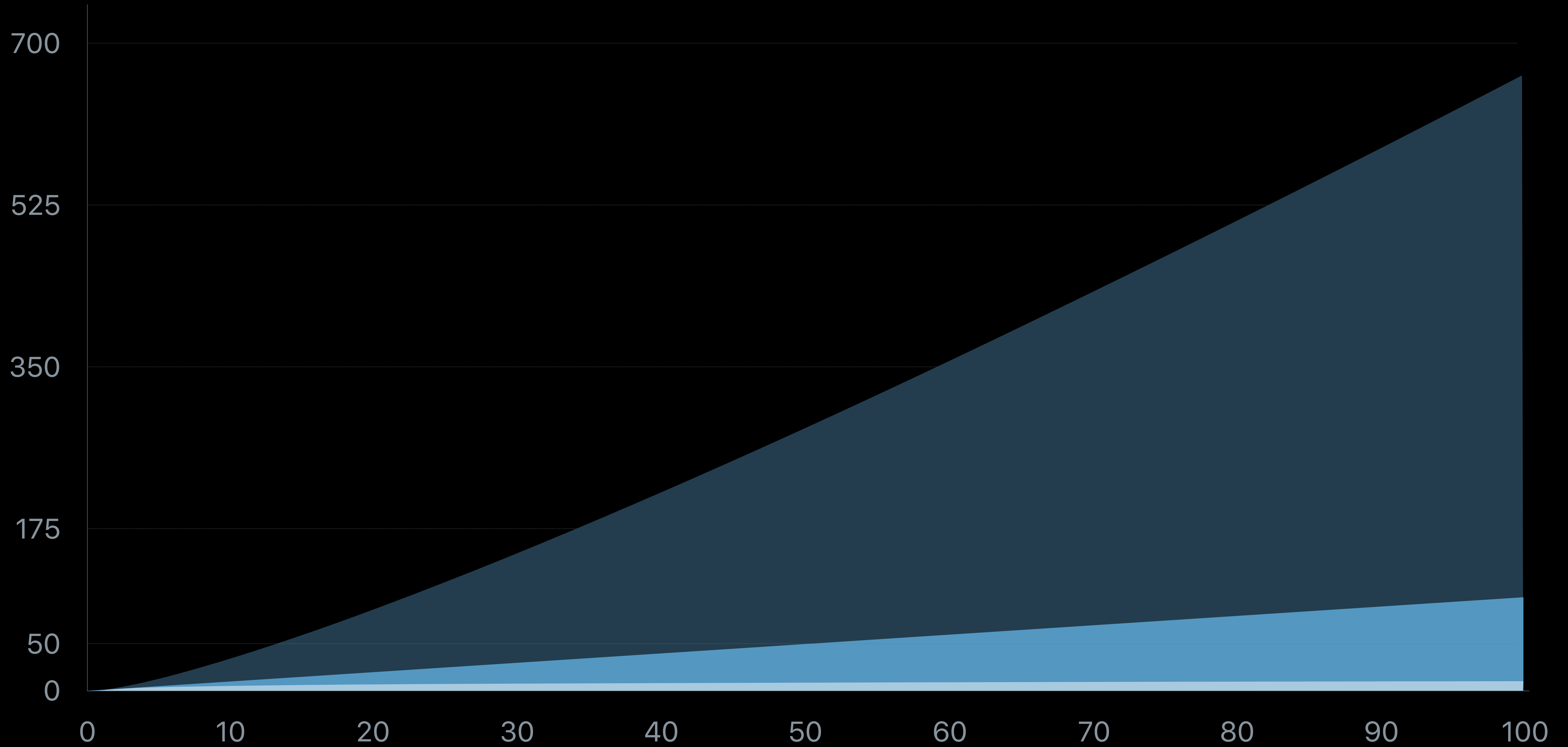
# Scalability Redux



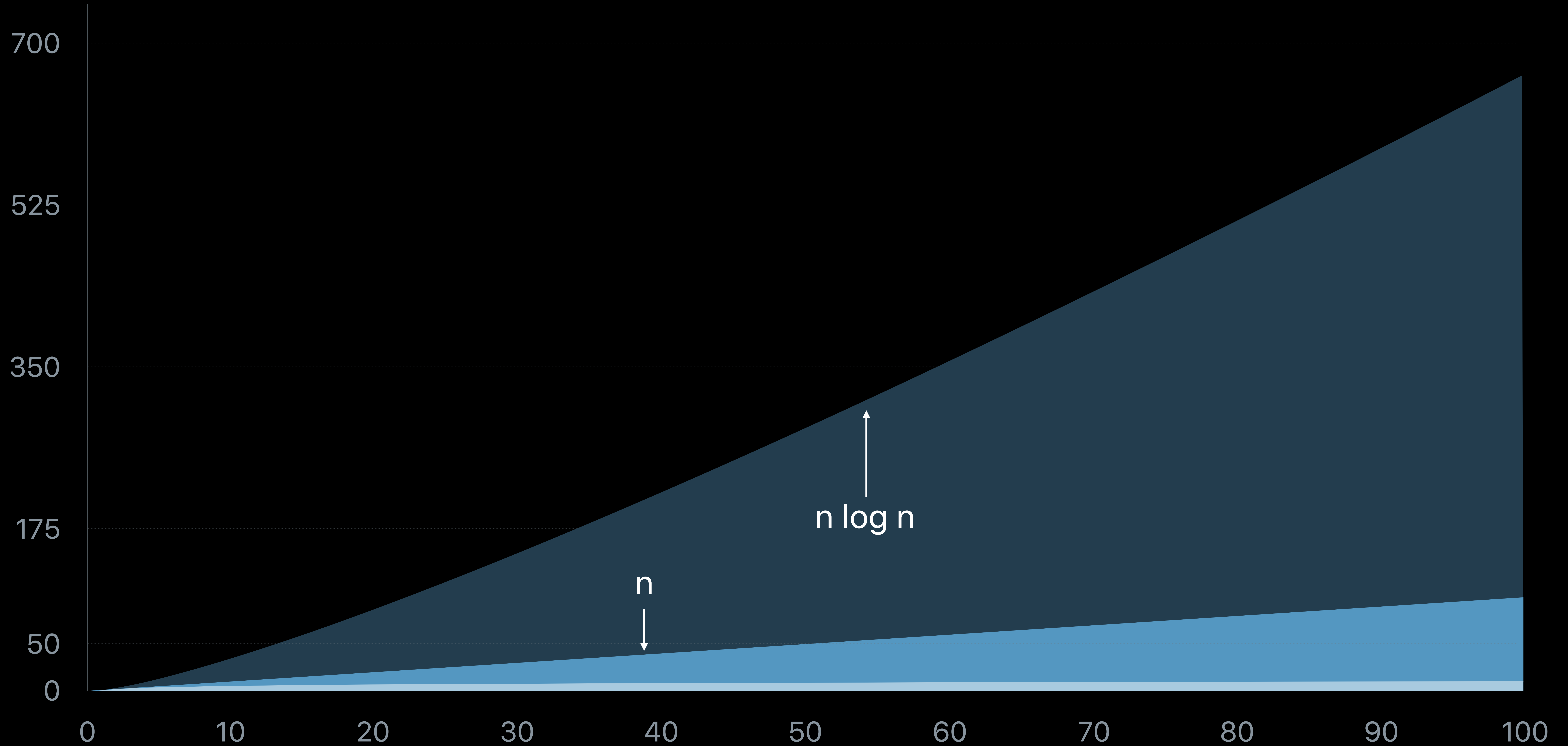
# Scalability Redux



# Scalability Redux



# Scalability Redux



```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```





```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```



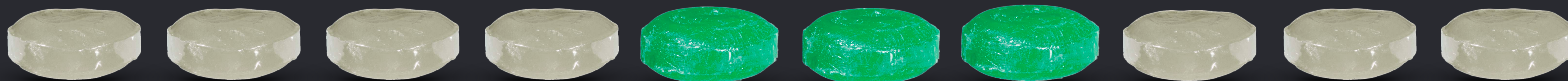
```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```



```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```



```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```



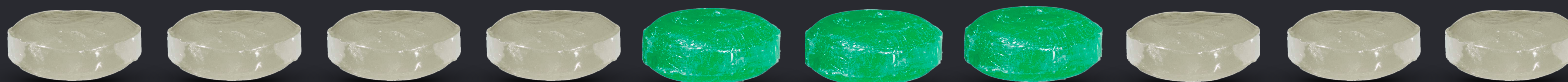
```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```



```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```



```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```



```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```

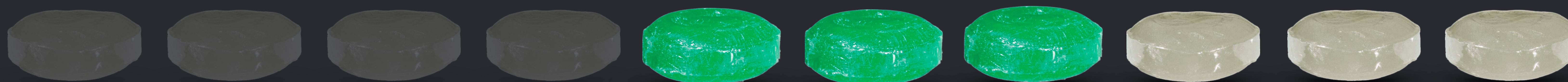




```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
  }  
}
```



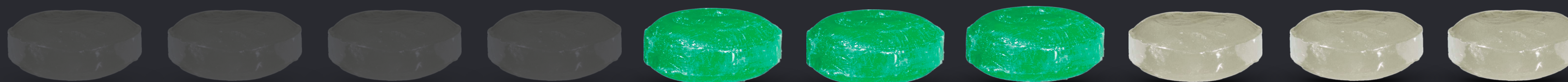
```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func bringForward() {  
    for i in shapes.indices where shapes[i].isSelected {  
      if i == 0 { return }  
      var insertionPoint = i - 1  
      for j in i..  
shapes.count where shapes[j].isSelected {  
        let x = shapes.remove(at: j)  
        shapes.insert(x, at: insertionPoint)  
        insertionPoint += 1  
      }  
    }  
    return  
  }  
}
```

```
}
```

```
}
```



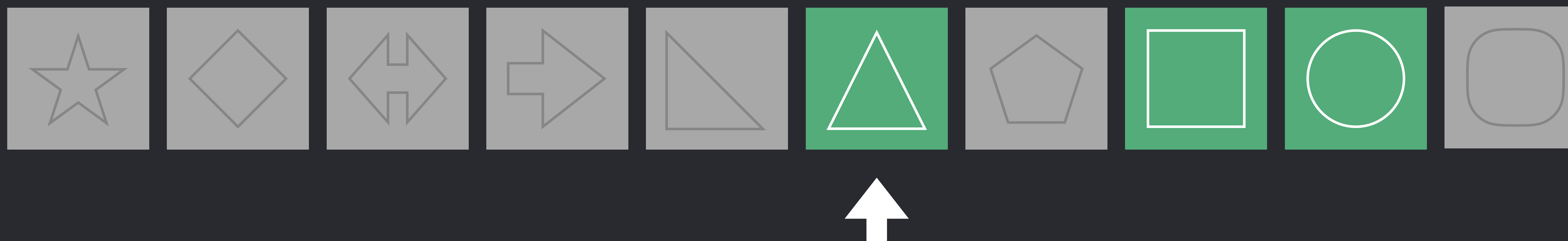
```
extension Canvas {  
  mutating func bringForward() {  
  
  }  
}
```



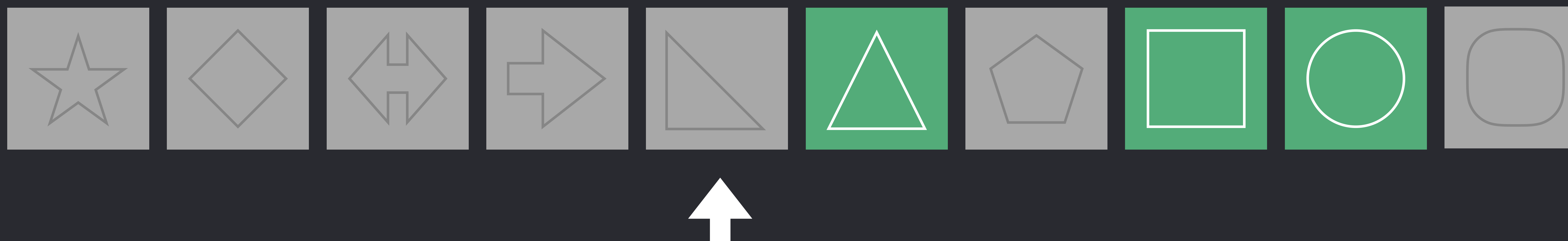
```
extension Canvas {  
  mutating func bringForward() {  
    if let i = shapes.firstIndex(where: { $0.isSelected }) {  
      if i == 0 { return }  
      let predecessor = i - 1  
    }  
  }  
}
```



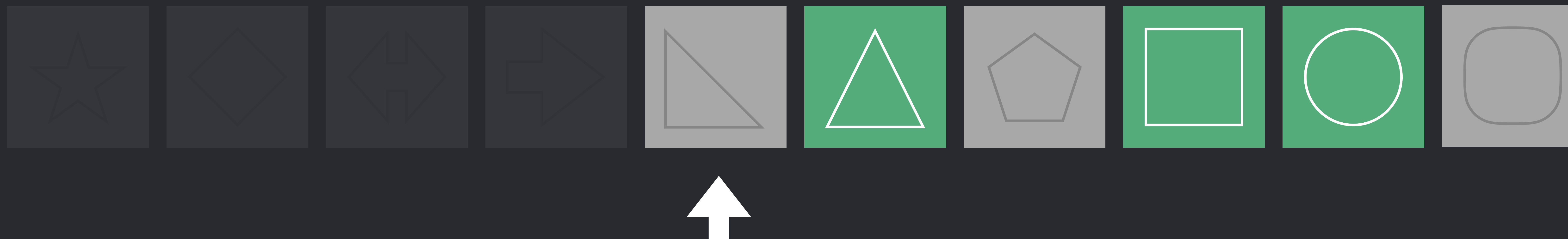
```
extension Canvas {  
  mutating func bringForward() {  
    if let i = shapes.firstIndex(where: { $0.isSelected }) {  
      if i == 0 { return }  
      let predecessor = i - 1  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func bringForward() {  
    if let i = shapes.firstIndex(where: { $0.isSelected }) {  
      if i == 0 { return }  
      let predecessor = i - 1  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func bringForward() {  
    if let i = shapes.firstIndex(where: { $0.isSelected }) {  
      if i == 0 { return }  
      let predecessor = i - 1  
    }  
  }  
}
```





```
extension Canvas {  
  mutating func bringForward() {  
    if let i = shapes.firstIndex(where: { $0.isSelected }) {  
      if i == 0 { return }  
      let predecessor = i - 1  
    }  
  }  
}
```

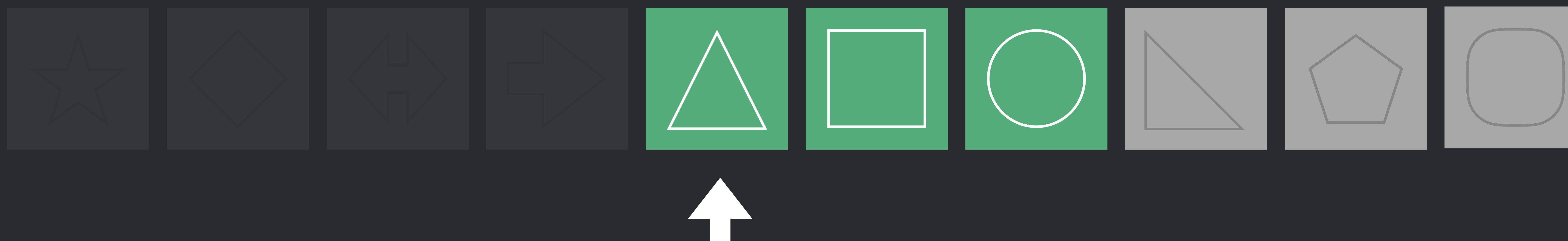


```
extension Canvas {  
  mutating func bringForward() {  
    if let i = shapes.firstIndex(where: { $0.isSelected }) {  
      if i == 0 { return }  
      let predecessor = i - 1  
      shapes[predecessor...]  
    }  
  }  
}
```

Means the same as  
`shapes[predecessor..  
endIndex]`



```
extension Canvas {  
  mutating func bringForward() {  
    if let i = shapes.firstIndex(where: { $0.isSelected }) {  
      if i == 0 { return }  
      let predecessor = i - 1  
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })  
    }  
  }  
}
```



# Discover Algorithms

```
extension Canvas {
  mutating func bringForward() {
    if let i = shapes.firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

```
extension Canvas {
  mutating func bringForward() {
    if let i = shapes.firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

A diagram consisting of a dark grey rectangular box containing the text "ArraySlice<Shape>". A small white triangle points from the top center of this box to the range "[predecessor...]" in the code line "shapes[predecessor...].stablePartition(isSuffixElement: { !\$0.isSelected })".

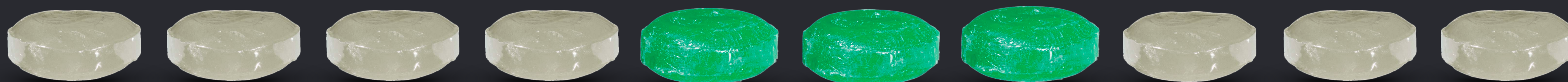
```
extension Canvas {  
  mutating func bringForward() {  
    if let i = shapes.firstIndex(where: { $0.isSelected }) {  
      if i == 0 { return }  
      let predecessor = i - 1  
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })  
    }  
  }  
}
```

ArraySlice<Shape>



```
extension Canvas {  
  mutating func bringForward() {  
    if let i = shapes.firstIndex(where: { $0.isSelected }) {  
      if i == 0 { return }  
      let predecessor = i - 1  
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })  
    }  
  }  
}
```

ArraySlice<Shape>





```
extension Canvas {
  mutating func bringForward() {
    if let i = shapes.firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

```
extension Canvas {
  mutating func bringForward() {
    if let i = shapes.firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

```
extension Canvas {
  mutating func bringForward() {
    if let i = shapes.firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

468

469

470

```
extension Canvas {
  mutating func bringForward() {
    if let i = shapes.firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

468

469

470

```
extension Array where Element == Shape {
  mutating func bringForward() {
    if let i = shapes.firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

```
extension Array where Element == Shape {
  mutating func bringForward() {
    if let i = shapes.firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      shapes[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

```
extension Array where Element == Shape {
  mutating func bringForward() {
    if let i = firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

```
extension Array where Element == Shape {
  mutating func bringForward() {
    if let i = firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```



```
extension Array where Element == Shape {
  mutating func bringForward() {
    if let i = firstIndex(where: { $0.isSelected }) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !$0.isSelected })
    }
  }
}
```

```
extension Array where Element == Shape {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let i = firstIndex(where: predicate) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```

```
extension Array where Element == Shape {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let i = firstIndex(where: predicate) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```

```
extension Array where Element == Shape {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let i = firstIndex(where: predicate) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```

```
extension Array {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == 0 { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```

```
extension Array {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let i = firstIndex(where: predicate) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```

```
extension Array {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let i = firstIndex(where: predicate) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```

```
extension MutableCollection {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let i = firstIndex(where: predicate) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```



```
extension MutableCollection {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let i = firstIndex(where: predicate) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```



```
extension MutableCollection where Index == Int {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == 0 { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```

! Binary operator '==' cannot be applied to operands of type 'Self.Index' and 'Int'  
! Binary operator '-' cannot be applied to operands of type 'Self.Index' and 'Int'



```
extension MutableCollection where Index == Int {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == 0 { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



```
extension MutableCollection where Index == Int {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let i = firstIndex(where: predicate) {
      if i == 0 { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```



```
extension MutableCollection where Index == Int {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let i = firstIndex(where: predicate) {
      if i == startIndex { return }
      let predecessor = i - 1
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```



```
extension MutableCollection where Index == Int {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == startIndex { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



```
extension MutableCollection where Index == Int {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == startIndex { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



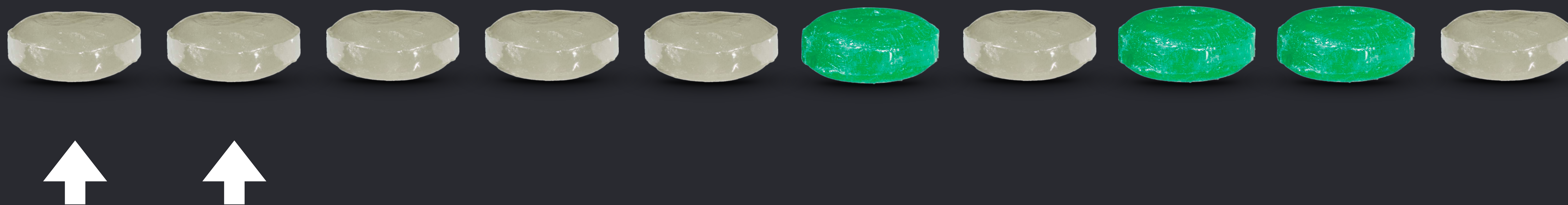
```
extension MutableCollection where Index == Int {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == startIndex { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```







```
extension MutableCollection where Index == Int {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == startIndex { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```





```
extension MutableCollection where Index == Int {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == startIndex { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



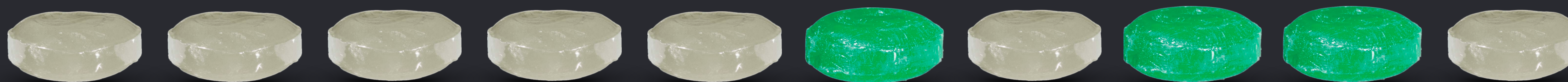
```
extension MutableCollection where Index == Int {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == startIndex { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



```
extension MutableCollection {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == startIndex { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



```
extension MutableCollection {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      if i == startIndex { return }  
      let predecessor = i - 1  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



```
extension MutableCollection {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



```
extension MutableCollection {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let i = firstIndex(where: predicate) {  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



```
extension MutableCollection {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let {  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```





```
extension MutableCollection {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let {  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



```
extension MutableCollection {  
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {  
    if let predecessor = indexBeforeFirst(where: predicate) {  
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })  
    }  
  }  
}
```



```
extension MutableCollection {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let predecessor = indexBeforeFirst(where: predicate) {
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```

```
extension MutableCollection {
    mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
        if let predecessor = indexBeforeFirst(where: predicate) {
            self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
        }
    }
}
```

```
extension Collection {
    func indexBeforeFirst(where predicate: (Element) -> Bool) -> Index? {
        return indices.first {
            let successor = index(after: $0)
            return successor != endIndex && predicate(self[successor])
        }
    }
}
```

```
extension MutableCollection {
    mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
        if let predecessor = indexBeforeFirst(where: predicate) {
            self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
        }
    }
}
```

```
extension Collection {
    func indexBeforeFirst(where predicate: (Element) -> Bool) -> Index? {
        return indices.first {
            let successor = index(after: $0)
            return successor != endIndex && predicate(self[successor])
        }
    }
}
```

```
extension MutableCollection {
    mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
        if let predecessor = indexBeforeFirst(where: predicate) {
            self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
        }
    }
}
```

```
extension Collection {
    func indexBeforeFirst(where predicate: (Element) -> Bool) -> Index? {
        return indices.first {
            let successor = index(after: $0)
            return successor != endIndex && predicate(self[successor])
        }
    }
}
```

```
extension MutableCollection {
    mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
        if let predecessor = indexBeforeFirst(where: predicate) {
            self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
        }
    }
}
```

```
extension Collection {
    func indexBeforeFirst(where predicate: (Element) -> Bool) -> Index? {
        return indices.first {
            let successor = index(after: $0)
            return successor != endIndex && predicate(self[successor])
        }
    }
}
```

```
extension MutableCollection {
    mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
        if let predecessor = indexBeforeFirst(where: predicate) {
            self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
        }
    }
}
```

```
extension Collection {
    func indexBeforeFirst(where predicate: (Element) -> Bool) -> Index? {
        return indices.first {
            let successor = index(after: $0)
            return successor != endIndex && predicate(self[successor])
        }
    }
}
```



# Building Towers Of Abstraction

# Building Towers Of Abstraction



```
extension MutableCollection {
    mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
        if let predecessor = indexBeforeFirst(where: predicate) {
            self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
        }
    }
}
```

```
extension Collection {
    func indexBeforeFirst(where predicate: (Element) -> Bool) -> Index? {
        return indices.first {
            let successor = index(after: $0)
            return successor != endIndex && predicate(self[successor])
        }
    }
}
```

```

extension MutableCollection {
    mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
        if let predecessor = indexBeforeFirst(where: predicate) {
            self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
        }
    }
}

extension Collection {
    /// Returns the `index` before the first one whose element satisfies `predicate`.
    ///
    /// - Complexity:  $O(n)$  where  $n$  is the length of the collection.
    func indexBeforeFirst(where predicate: (Element) -> Bool) -> Index? {
        return indices.first {
            let successor = index(after: $0)
            return successor != endIndex && predicate(self[successor])
        }
    }
}

```

```
extension MutableCollection {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let predecessor = indexBeforeFirst(where: predicate) {
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```

```
extension MutableCollection {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let predecessor = indexBeforeFirst(where: predicate) {
      self.moveElement(at: predecessor, to: indexBeforeFirst(where: { !predicate($0) }))
    }
  }
}
```

### Summary

Returns the index before the first one whose element satisfies predicate.

### Declaration

```
func indexBeforeFirst(where predicate: (Self.Element) -> Bool) -> Self.Index?
```

### Discussion

### Complexity

$O(n)$  where  $n$  is the length of the collection.

### Parameters

`predicate` No description.

### Declared In

[Example.playground](#)

```
extension MutableCollection {
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let predecessor = indexBeforeFirst(where: predicate) {
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```

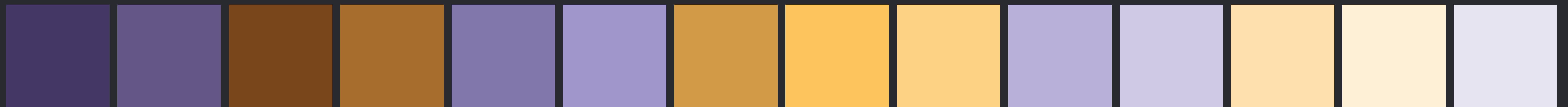
```
extension MutableCollection {
  /// Gathers the elements satisfying `predicate` at the position preceding
  /// the first element satisfying `predicate`.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the length of the collection.
  mutating func bringForward(elementsSatisfying predicate: (Element) -> Bool) {
    if let predecessor = indexBeforeFirst(where: predicate) {
      self[predecessor...].stablePartition(isSuffixElement: { !predicate($0) })
    }
  }
}
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

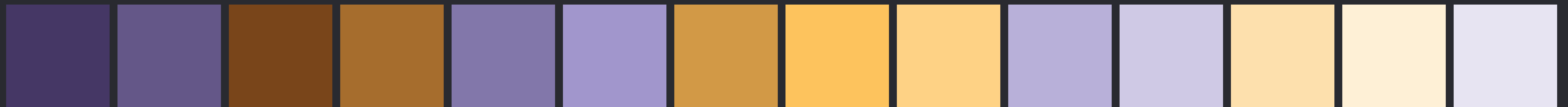
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

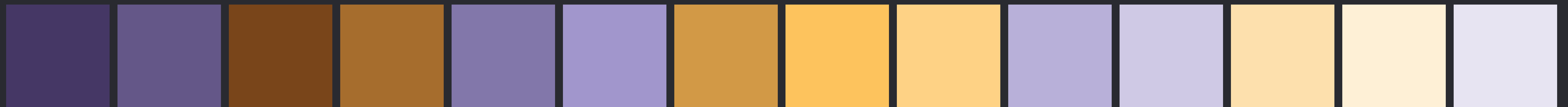
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

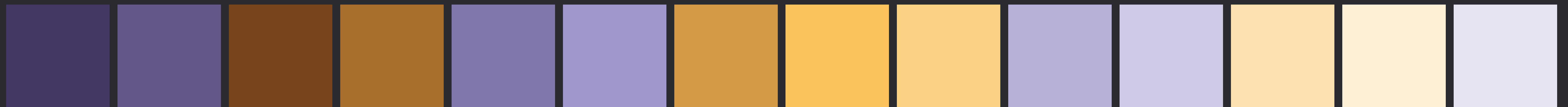
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

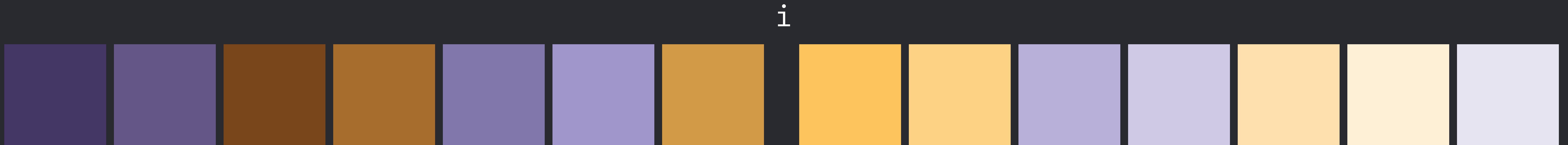
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

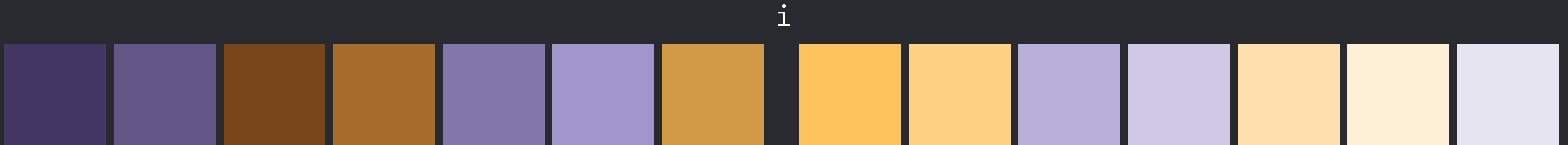
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

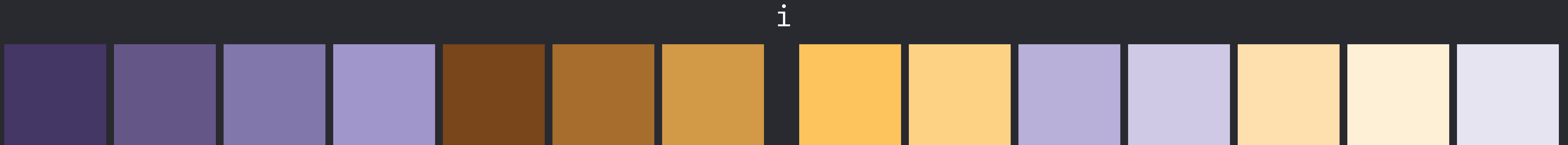
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

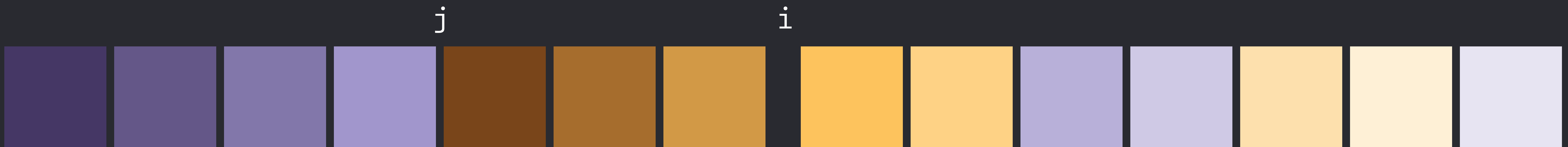
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

```

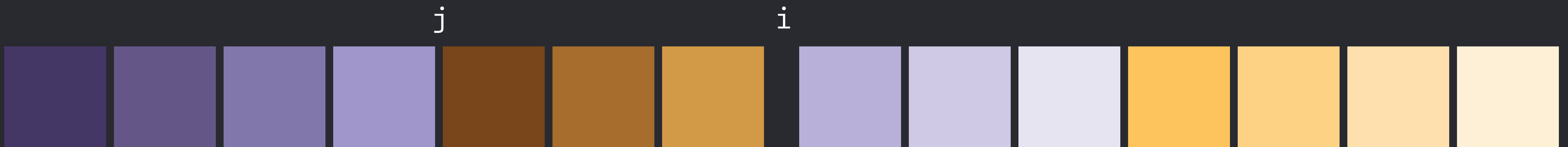




```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

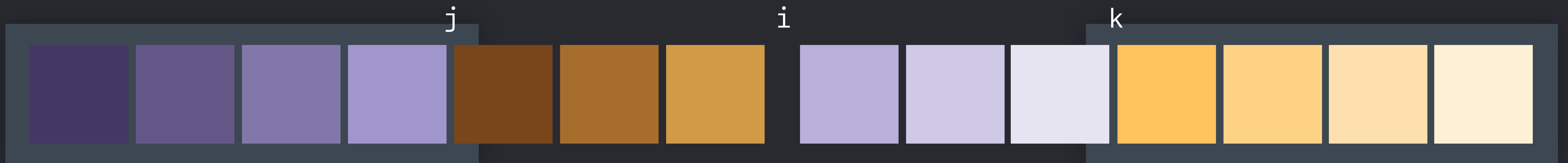
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

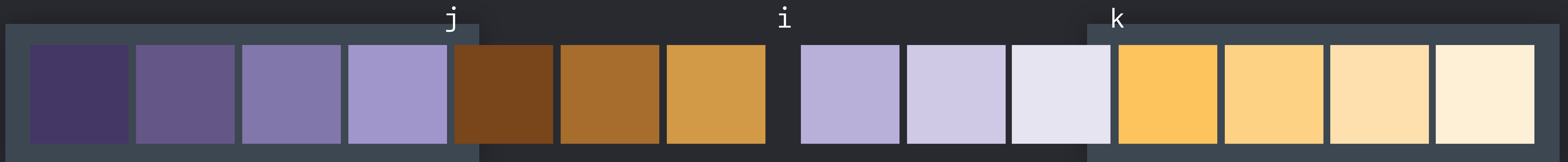
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

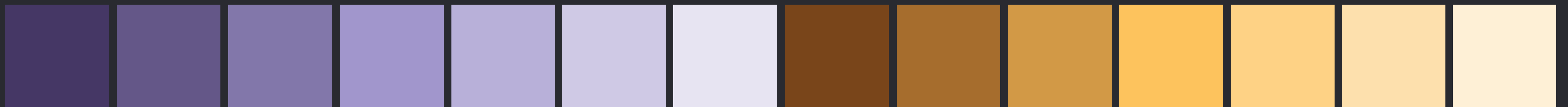
```



```

extension MutableCollection {
  /// Moves all elements satisfying `isSuffixElement` into a suffix of the collection,
  /// preserving their relative order, returning the start of the resulting suffix.
  ///
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
  /// - Precondition: `n == self.count`
  mutating func stablePartition(count n: Int, isSuffixElement: (Element) -> Bool) -> Index {
    if n == 0 { return startIndex }
    if n == 1 { return isSuffixElement(self[startIndex]) ? startIndex : endIndex }
    let h = n / 2, i = index(startIndex, offsetBy: h)
    let j = self[..<i].stablePartition(count: h, isSuffixElement: isSuffixElement)
    let k = self[i...].stablePartition(count: n - h, isSuffixElement: isSuffixElement)
    return self[j..<k].rotate(shiftingToStart: i)
  }
}

```



```
extension Canvas {
  mutating func gatherSelected(at target: Int) {
    var buffer: [Shape] = []
    var insertionPoint = target
    var i = 0

    while i < insertionPoint {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        buffer.append(x)
        insertionPoint -= 1
      }
      else {
        i += 1
      }
    }

    while i < shapes.count {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        buffer.append(x)
      }
      else {
        i += 1
      }
    }

    shapes.insert(contentsOf: buffer, at: insertionPoint)
  }
}
```





```
extension Canvas {  
  mutating func gatherSelected(at target: Int) {  
    var buffer: [Shape] = []  
    var insertionPoint = target  
    var i = 0  
  
    while i < insertionPoint {  
      if shapes[i].isSelected {  
        let x = shapes.remove(at: i)  
        buffer.append(x)  
        insertionPoint -= 1  
      }  
      else {  
        i += 1  
      }  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func gatherSelected(at target: Int) {  
    var buffer: [Shape] = []  
    var insertionPoint = target  
    var i = 0  
  
    while i < insertionPoint {  
      if shapes[i].isSelected {  
        let x = shapes.remove(at: i)  
        buffer.append(x)  
        insertionPoint -= 1  
      }  
      else {  
        i += 1  
      }  
    }  
  }  
}
```



```
extension Canvas {  
  mutating func gatherSelected(at target: Int) {  
    var buffer: [Shape] = []  
    var insertionPoint = target  
    var i = 0  
  
    while i < insertionPoint {  
      if shapes[i].isSelected {  
        let x = shapes.remove(at: i)  
        buffer.append(x)  
        insertionPoint -= 1  
      }  
      else {  
        i += 1  
      }  
    }  
  }  
}
```



```
mutating func gatherSelected(at target: Int) {  
    var buffer: [Shape] = []  
    var insertionPoint = target  
    var i = 0  
  
    while i < insertionPoint {  
        if shapes[i].isSelected {  
            let x = shapes.remove(at: i)  
            buffer.append(x)  
            insertionPoint -= 1  
        }  
        else {  
            i += 1  
        }  
    }  
}
```



```
mutating func gatherSelected(at target: Int) {
```

```
  var buffer: [Shape] = []
```

```
  var insertionPoint = target
```

```
  var i = 0
```

```
  while i < insertionPoint {
```

```
    if shapes[i].isSelected {
```

```
      let x = shapes.remove(at: i)
```

```
      buffer.append(x)
```

```
      insertionPoint -= 1
```

```
    }
```

```
  else {
```

```
    i += 1
```

```
  }
```

```
}
```



```
mutating func gatherSelected(at target: Int) {  
    var buffer: [Shape] = []  
    var insertionPoint = target  
    var i = 0
```

```
    while i < insertionPoint {  
        if shapes[i].isSelected {  
            let x = shapes.remove(at: i)  
            buffer.append(x)  
            insertionPoint -= 1  
        }  
        else {  
            i += 1  
        }  
    }  
}
```



```
mutating func gatherSelected(at target: Int) {  
  var buffer: [Shape] = []  
  var insertionPoint = target  
  var i = 0
```

```
  while i < insertionPoint {  
    if shapes[i].isSelected {  
      let x = shapes.remove(at: i)  
      buffer.append(x)  
      insertionPoint -= 1  
    }  
    else {  
      i += 1  
    }  
  }  
}
```



```
mutating func gatherSelected(at target: Int) {  
  var buffer: [Shape] = []  
  var insertionPoint = target  
  var i = 0
```

```
  while i < insertionPoint {  
    if shapes[i].isSelected {  
      let x = shapes.remove(at: i)  
      buffer.append(x)  
      insertionPoint -= 1  
    }  
    else {  
      i += 1  
    }  
  }  
}
```





```
else {  
    i += 1  
}  
}
```

```
while i < shapes.count {  
    if shapes[i].isSelected {  
        let x = shapes.remove(at: i)  
        buffer.append(x)  
    }  
    else {  
        i += 1  
    }  
}
```



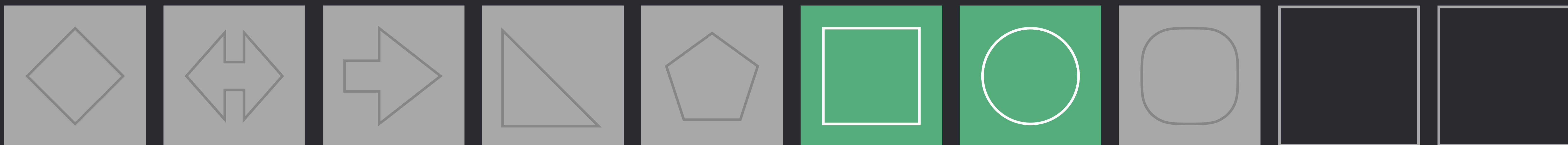
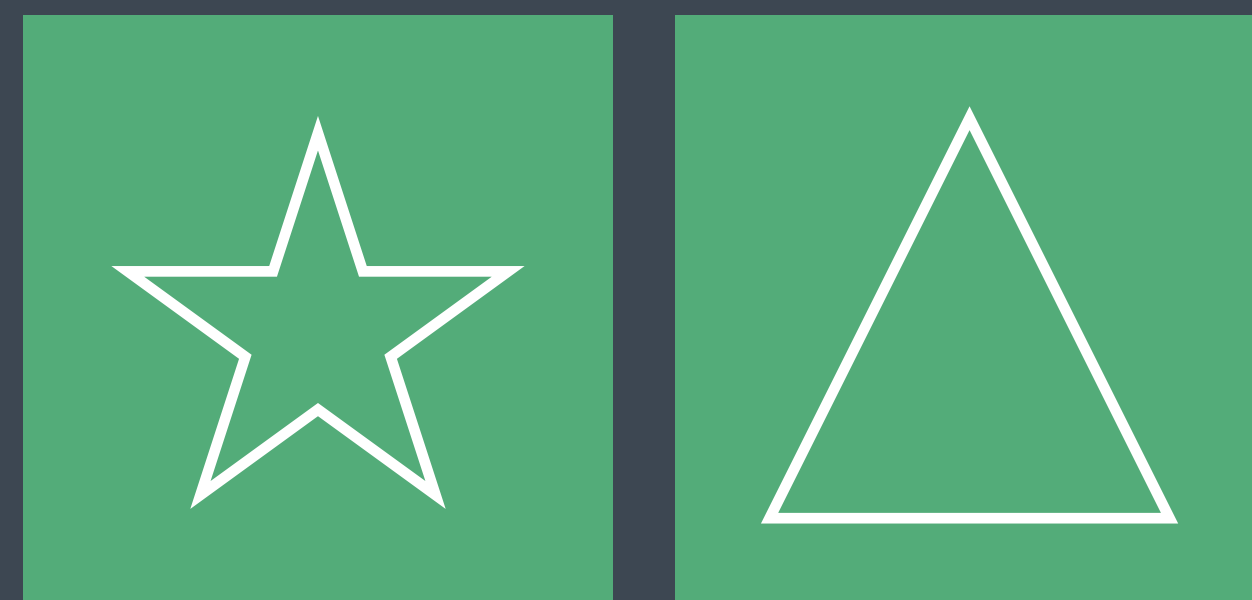
```
else {  
    i += 1  
}  
}
```

```
while i < shapes.count {  
    if shapes[i].isSelected {  
        let x = shapes.remove(at: i)  
        buffer.append(x)  
    }  
    else {  
        i += 1  
    }  
}
```



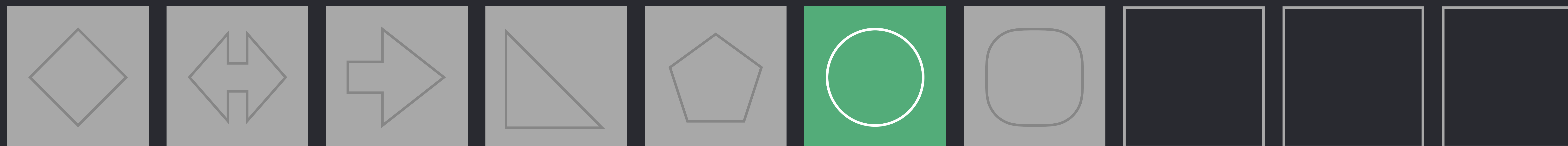
```
else {  
    i += 1  
}  
}
```

```
while i < shapes.count {  
    if shapes[i].isSelected {  
        let x = shapes.remove(at: i)  
        buffer.append(x)  
    }  
    else {  
        i += 1  
    }  
}
```



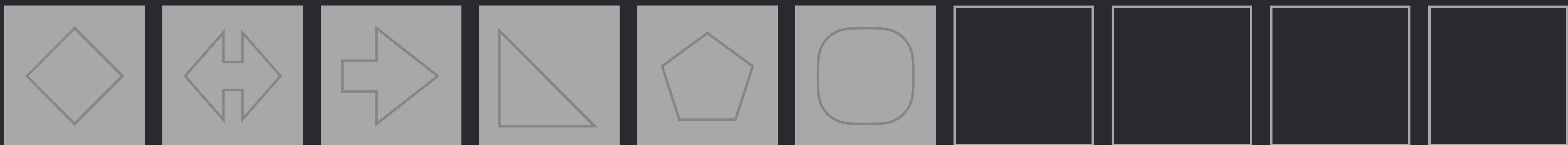
```
else {  
    i += 1  
}  
}
```

```
while i < shapes.count {  
    if shapes[i].isSelected {  
        let x = shapes.remove(at: i)  
        buffer.append(x)  
    }  
    else {  
        i += 1  
    }  
}
```



```
else {  
    i += 1  
}  
}
```

```
while i < shapes.count {  
    if shapes[i].isSelected {  
        let x = shapes.remove(at: i)  
        buffer.append(x)  
    }  
    else {  
        i += 1  
    }  
}
```

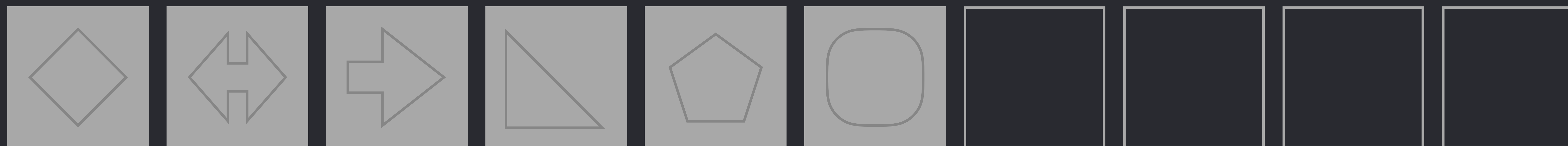


```
if shapes[i].isSelected {  
    let x = shapes.remove(at: i)  
    buffer.append(x)  
}  
else {  
    i += 1  
}  
}
```

```
shapes.insert(contentsOf: buffer, at: insertionPoint)
```

```
}
```

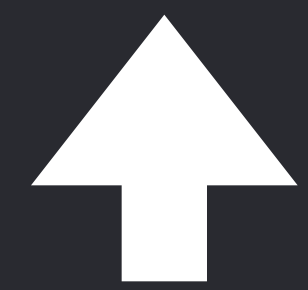
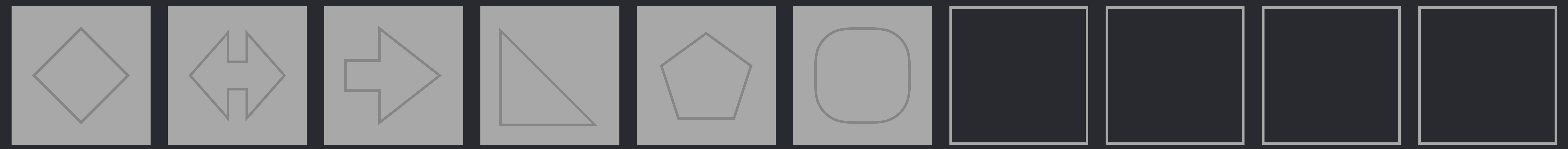
```
}
```



```
if shapes[i].isSelected {
    let x = shapes.remove(at: i)
    buffer.append(x)
}
else {
    i += 1
}
}
```

```
shapes.insert(contentsOf: buffer, at: insertionPoint)
```

```
}
```



```
if shapes[i].isSelected {  
    let x = shapes.remove(at: i)  
    buffer.append(x)  
}  
else {  
    i += 1  
}  
}
```

```
shapes.insert(contentsOf: buffer, at: insertionPoint)
```

```
}
```

```
}
```





```
if shapes[i].isSelected {
  let x = shapes.remove(at: i)
  buffer.append(x)
}
else {
  i += 1
}
}

shapes.insert(contentsOf: buffer, at: insertionPoint)
}
}
```



```
if shapes[i].isSelected {
    let x = shapes.remove(at: i)
    buffer.append(x)
}
else {
    i += 1
}
}

shapes.insert(contentsOf: buffer, at: insertionPoint)
}
}
```



```
    if shapes[i].isSelected {
      let x = shapes.remove(at: i)
      buffer.append(x)
    }
    else {
      i += 1
    }
  }
}

shapes.insert(contentsOf: buffer, at: insertionPoint)
}
}
```



```
    if shapes[i].isSelected {
      let x = shapes.remove(at: i)
      buffer.append(x)
    }
    else {
      i += 1
    }
  }
}

shapes.insert(contentsOf: buffer, at: insertionPoint)
}
}
```



```
    if shapes[i].isSelected {
      let x = shapes.remove(at: i)
      buffer.append(x)
    }
    else {
      i += 1
    }
  }
}

shapes.insert(contentsOf: buffer, at: insertionPoint)
}
}
```



```
    if shapes[i].isSelected {
      let x = shapes.remove(at: i)
      buffer.append(x)
    }
    else {
      i += 1
    }
  }
}

shapes.insert(contentsOf: buffer, at: insertionPoint)
}
}
```



```
if shapes[i].isSelected {
  let x = shapes.remove(at: i)
  buffer.append(x)
}
else {
  i += 1
}
}

shapes.insert(contentsOf: buffer, at: insertionPoint)
}
}
```



```
if shapes[i].isSelected {
    let x = shapes.remove(at: i)
    buffer.append(x)
}
else {
    i += 1
}
}

shapes.insert(contentsOf: buffer, at: insertionPoint)
}
}
```





```
    if shapes[i].isSelected {
      let x = shapes.remove(at: i)
      buffer.append(x)
    }
    else {
      i += 1
    }
  }
}

shapes.insert(contentsOf: buffer, at: insertionPoint)
}
}
```



```
extension MutableCollection {  
  /// Gathers elements satisfying `predicate` at `target`, preserving their relative order.  
  ///  
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.  
  mutating func gather(at target: Index, allSatisfying predicate: (Element) -> Bool) {  
    let start = self[..<target].stablePartition(isSuffixElement: predicate)  
    let end = self[target...].stablePartition(isSuffixElement: { !predicate($0) })  
  }  
}
```



```
extension MutableCollection {  
  /// Gathers elements satisfying `predicate` at `target`, preserving their relative order.  
  ///  
  /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.  
  mutating func gather(at target: Index, allSatisfying predicate: (Element) -> Bool) {  
    let start = self[..<target].stablePartition(isSuffixElement: predicate)  
    let end = self[target...].stablePartition(isSuffixElement: { !predicate($0) })  
  }  
}
```

```
extension Canvas {  
  mutating func gatherSelected(at target: Int) {  
    shapes.gather(at: target) { $0.isSelected }  
  }  
}
```



```
extension MutableCollection {
    /// Gathers elements satisfying `predicate` at `target`, preserving their relative order.
    ///
    /// - Complexity:  $O(n \log n)$  where  $n$  is the number of elements.
    mutating func gather(at target: Index, allSatisfying predicate: (Element)->Bool) {
        let start = self[..<target].stablePartition(isSuffixElement: predicate)
        let end = self[target...].stablePartition(isSuffixElement: { !predicate($0) })
    }
}

extension Canvas {
    mutating func gatherSelected(at target: Int) {
        shapes.gather(at: target) { $0.isSelected }
    }
}
```

```

extension Canvas {
  mutating func gatherSelected(at target: Int) {
    var buffer: [Shape] = []
    var insertionPoint = target
    var i = 0

    while i < insertionPoint {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        buffer.append(x)
        insertionPoint -= 1
      }
      else {
        i += 1
      }
    }

    while i < shapes.count {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        buffer.append(x)
      }
      else {
        i += 1
      }
    }

    shapes.insert(contentsOf: buffer, at: insertionPoint)
  }
}

```

```

extension MutableCollection {
  /// Gathers elements satisfying `predicate` at `target`, preserving their relative order.
  ///
  /// - Complexity: O(n log n) where n is the number of elements.
  mutating func gather(at target: Index, allSatisfying predicate: (Element)->Bool) {
    let start = self[..<target].stablePartition(isSuffixElement: predicate)
    let end = self[target...].stablePartition(isSuffixElement: { !predicate($0) })
  }
}

extension Canvas {
  mutating func gatherSelected(at target: Int) {
    shapes.gather(at: target) { $0.isSelected }
  }
}

```

```

extension Canvas {
  mutating func gatherSelected(at target: Int) {
    var buffer: [Shape] = []
    var insertionPoint = target
    var i = 0

    while i < insertionPoint {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        buffer.append(x)
        insertionPoint -= 1
      }
      else {
        i += 1
      }
    }

    while i < shapes.count {
      if shapes[i].isSelected {
        let x = shapes.remove(at: i)
        buffer.append(x)
      }
      else {
        i += 1
      }
    }

    shapes.insert(contentsOf: buffer, at: insertionPoint)
  }
}

```

```

extension MutableCollection {
  /// Gathers elements satisfying `predicate` at `target`, preserving their relative order.
  ///
  /// - Complexity: O(n log n) where n is the number of elements.
  mutating func gather(at target: Index, allSatisfying predicate: (Element)->Bool) {
    let start = self[..<target].stablePartition(isSuffixElement: predicate)
    let end = self[target...].stablePartition(isSuffixElement: { !predicate($0) })
  }
}

extension Canvas {
  mutating func gatherSelected(at target: Int) {
    shapes.gather(at: target) { $0.isSelected }
  }
}

```

**Discover Generic Algorithms**

“If you want to improve the code quality in your organization, replace all of your coding guidelines with one goal:

**No Raw Loops”**

Sean Parent, *C++ Seasoning*



# More Information

Sean Parent, C++ Seasoning

<https://channel9.msdn.com/Events/GoingNative/2013/Cpp-Seasoning>

Alexander Stepanov and Paul McJones, Elements of Programming

<https://www.youtube.com/watch?v=Ih9gpJga4Vc>

Swift Algorithms Prototype

<https://github.com/apple/swift/blob/master/test/Prototypes/Algorithms.swift>

Swift Standard Library Documentation

[https://developer.apple.com/documentation/swift/swift\\_standard\\_library](https://developer.apple.com/documentation/swift/swift_standard_library)

# More Information

<https://developer.apple.com/wwdc18/223>

---

Swift Generics

Hall 2

Wednesday 3:00PM

---

 **WWDC18**