

Macoun

Auto-Layout

The Hall of Frame

Clemens Wagner
macmoonshine@gmx.de

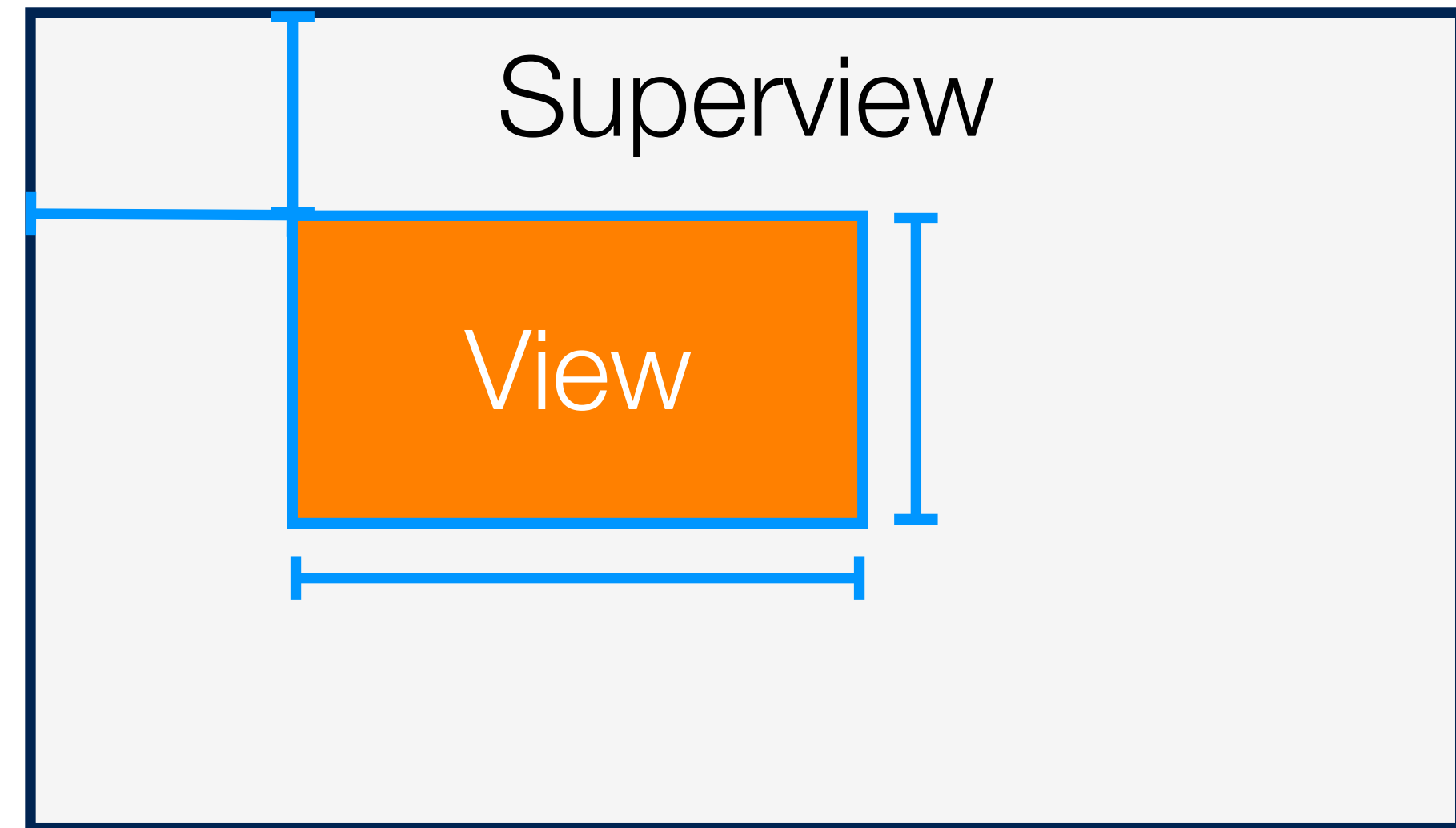
Adaptive Layouts

Adaptives Layout

bestmögliche Anpassung an
Bildschirmgröße

Auto-Sizing

- Ausrichtung über Frame
- Abstand zum Rand und Größe
- Verankerungen (Autosizing-Maske)



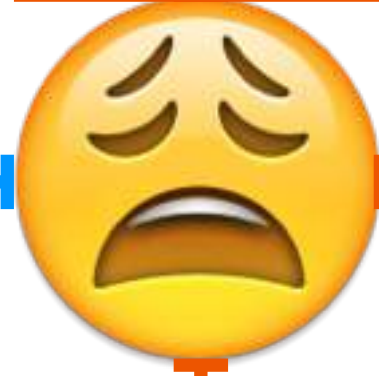
Adaptives Auto-Sizing?

Adaptives Layout

=

Auto-Sizing + Programmieren

Auto-Sizing

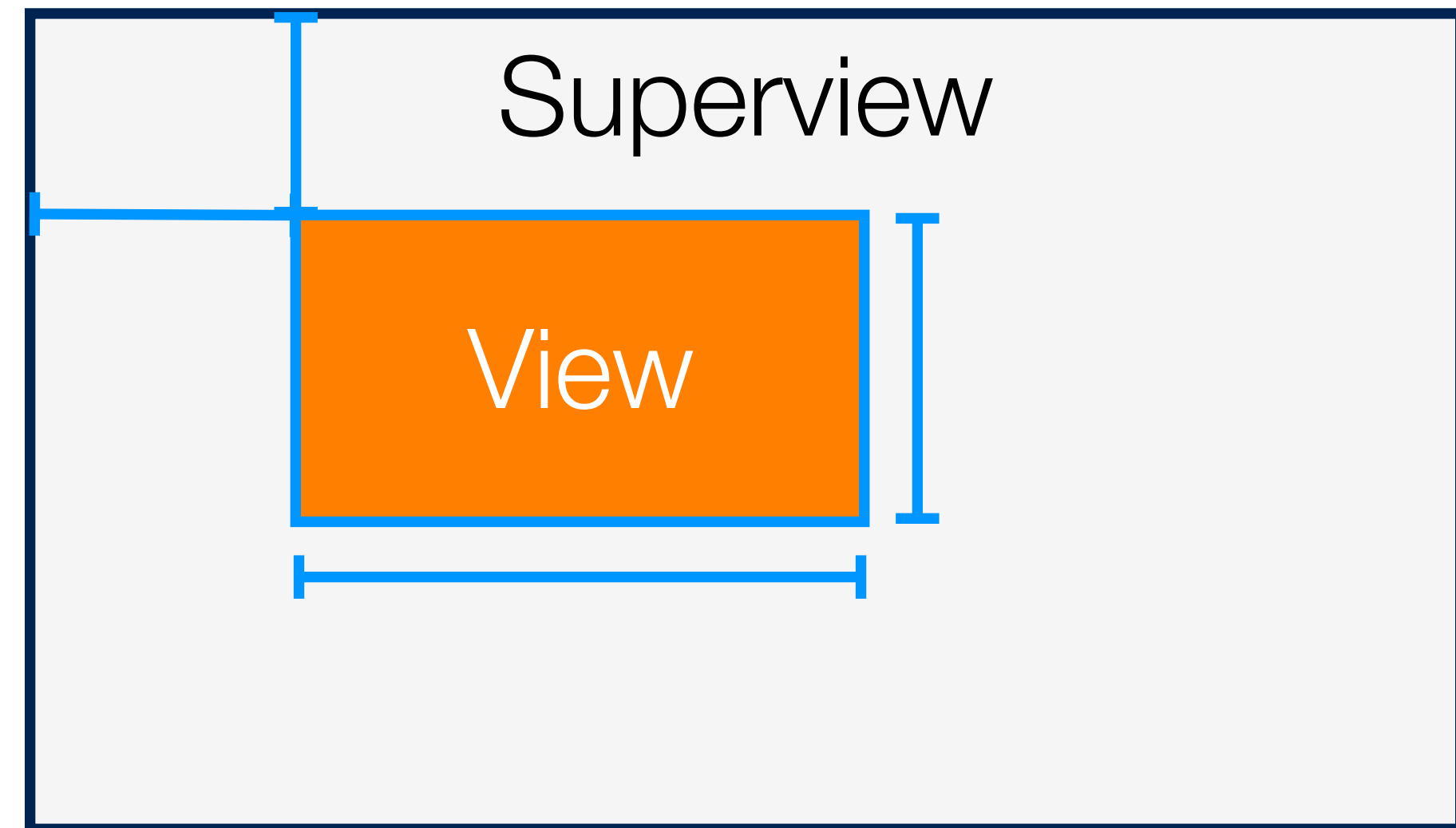


Mit Auto-Sizing lassen sich nur einfache adaptive Layouts umsetzen.

Da Auto-Sizing es nur erlaubt, die Abstände zum Superview zu setzen, lassen sich beispielsweise Layouts mit mehreren Views in einem Superview nur schlecht durch Frames und Auto-Sizing-Masken beschreiben. Der Programmierer muss für solche Layouts Code erstellen, der die Frames berechnet.

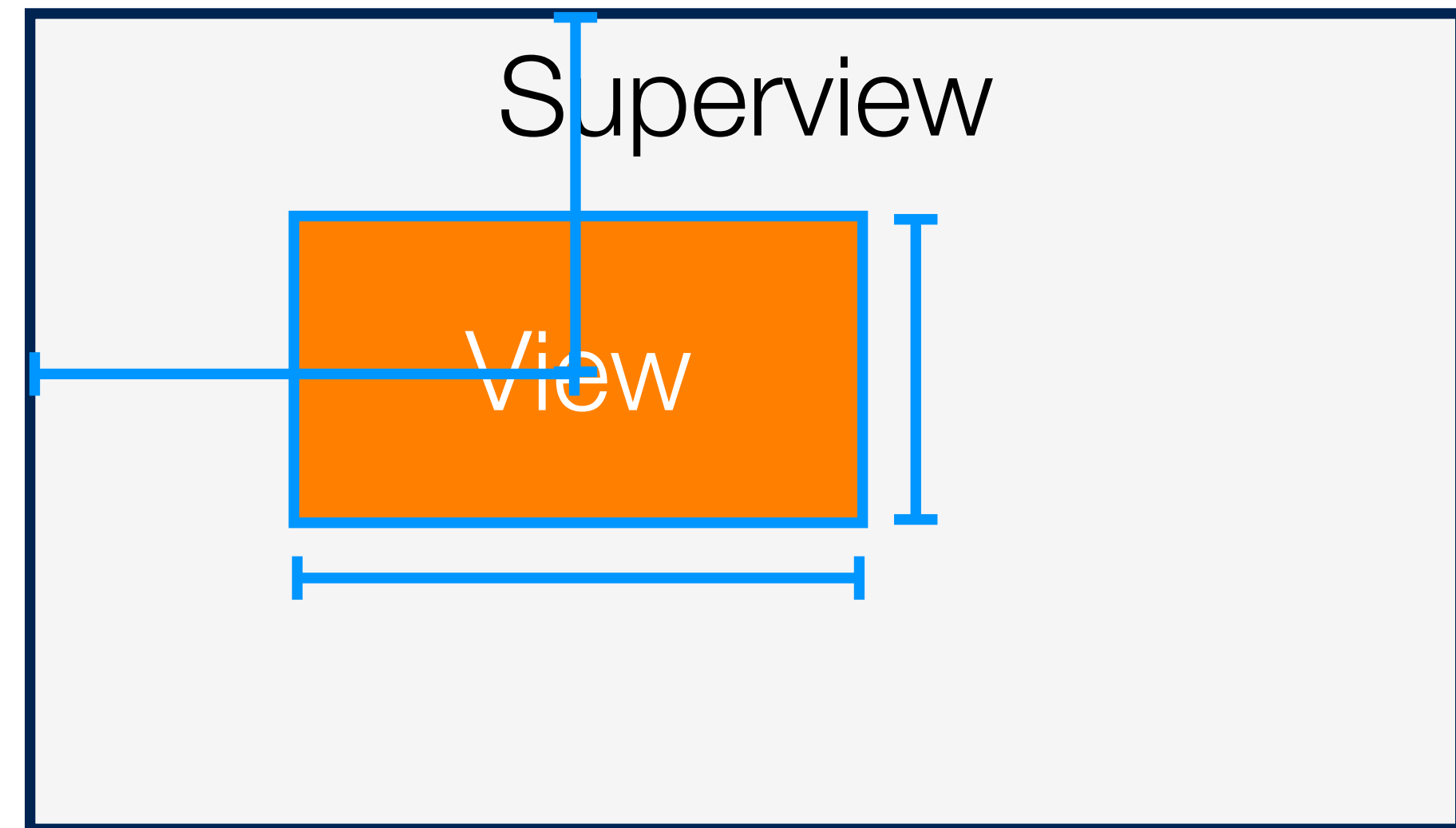
Auto-Layout

- Ausrichtung über 2 Constraints pro Dimension
- Berechnung des Frames



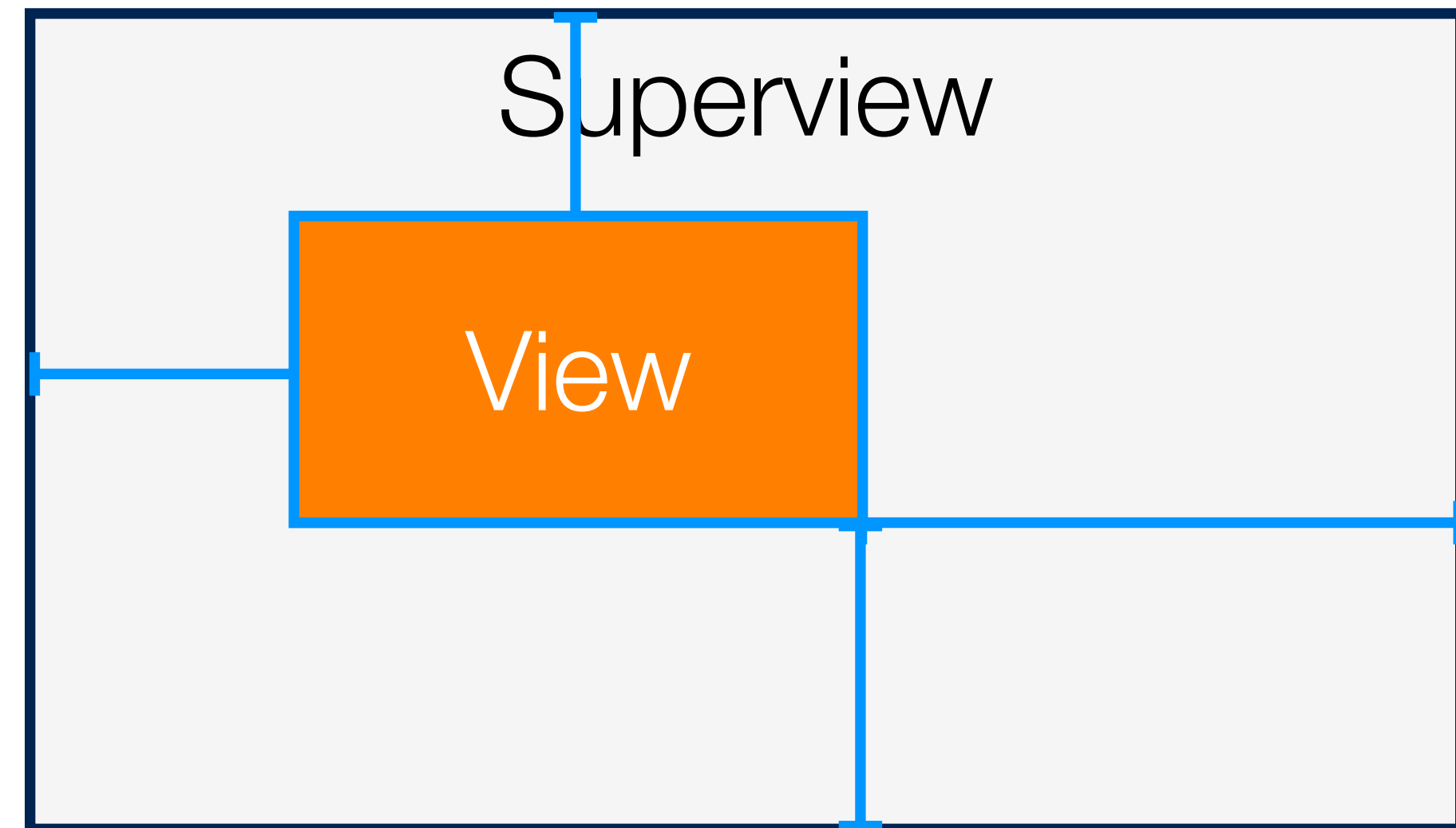
Auto-Layout

Mittelpunkt und Größe



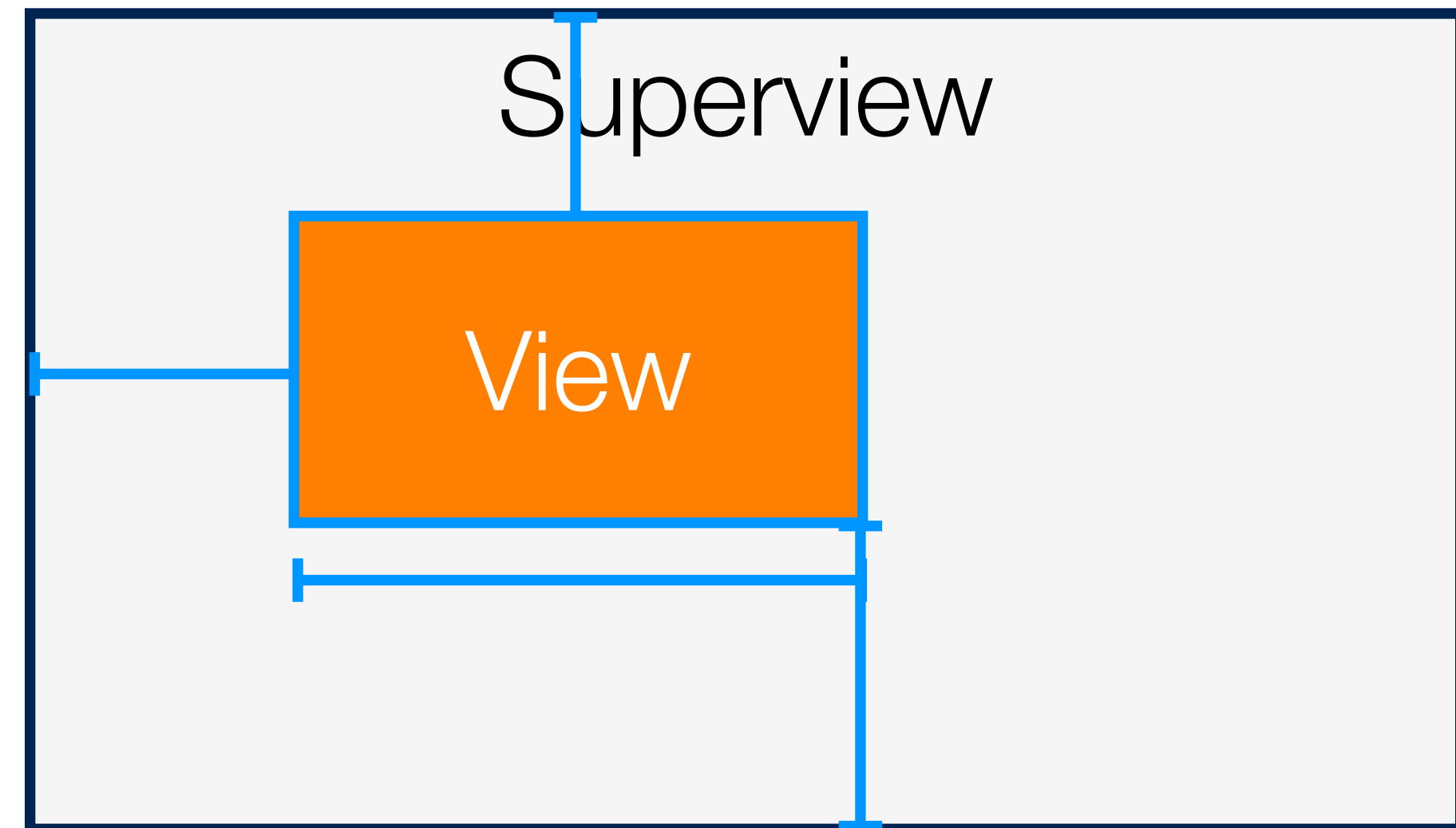
Auto-Layout

Zwei Punkte



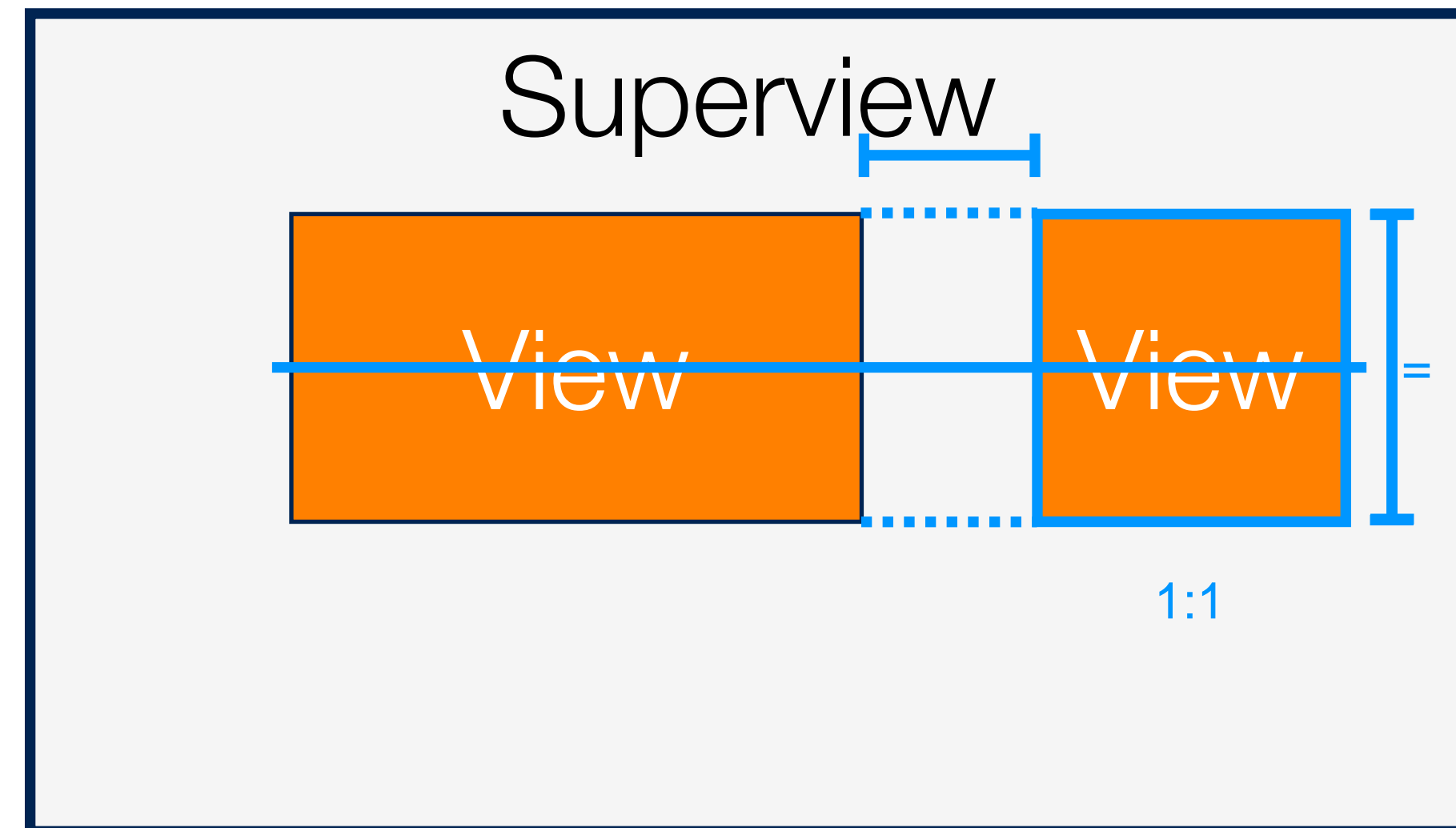
Auto-Layout

oder gemischt



Auto-Layout

relativ zu anderen Views



Adaptives Auto-Layout?

Adaptives Layout

=

Auto-Layout

Auto-Layout



Mit Auto-Layout lassen sich auch komplexe adaptive Layouts umsetzen.

Da Auto-Layout nahezu beliebige Abstände zu anderen Views beschreiben kann, lassen sich damit auch sehr gut komplexere Layouts beschreiben. Der Programmierer muss in der Regel sehr wenig Code erstellen, um Frames der Views zu berechnen.

Layout-Constraint

item1.attribute1 = multiplier × item2.attribute2 + constant

Demo

Layout-Constraints anlegen

Layout-Constraints anlegen

- Interface-Builder
- Viewcontroller
- View



Layout-Constraints erzeugen

- Convenience Konstruktor
- Visual Format Language 🙄
- Layout-Anker 

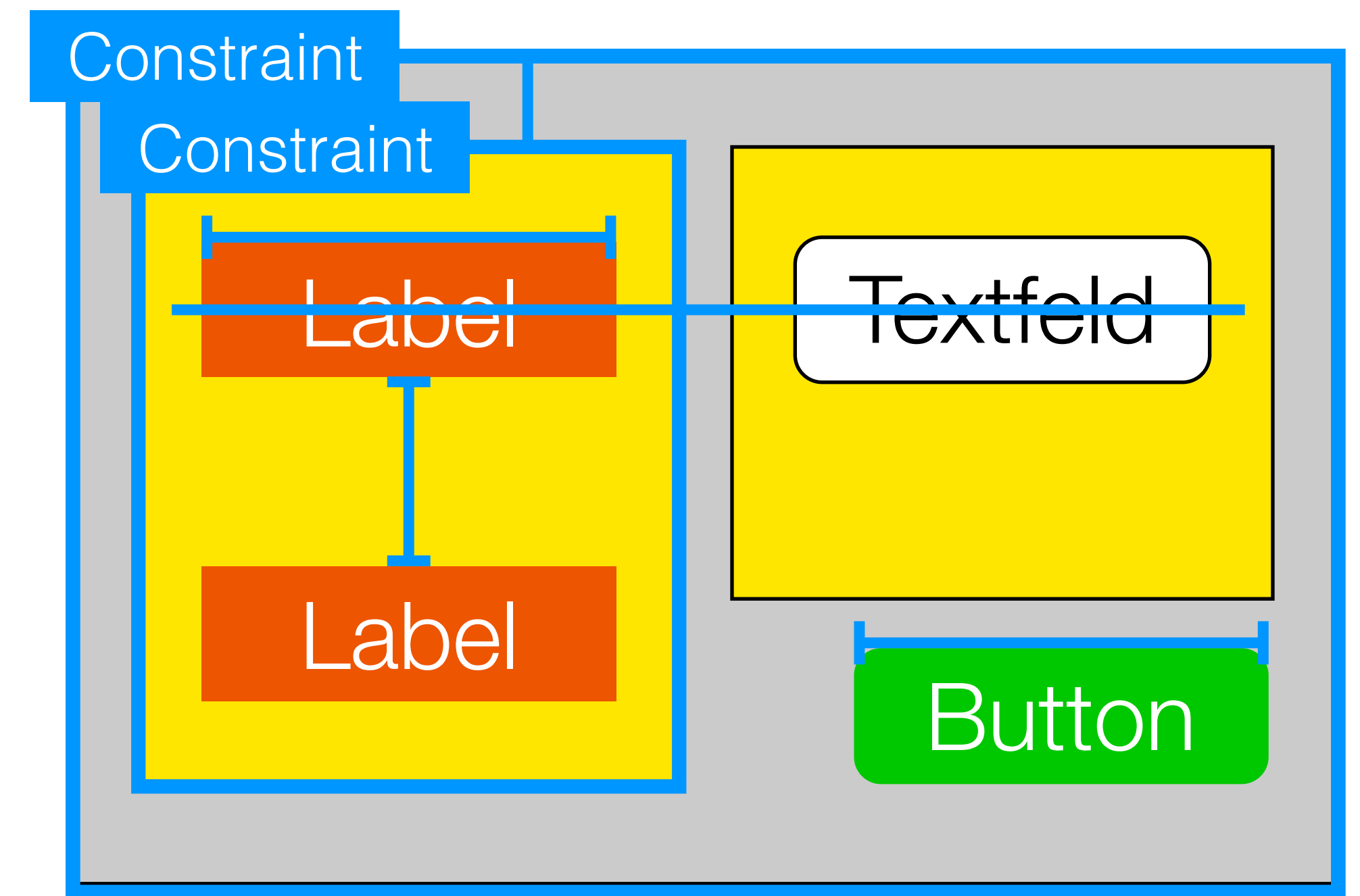
Layout-Anker

```
theConstraint = [NSLayoutConstraint constraintWithItem:theSuperView  
    attribute:NSLayoutAttributeLeading  
    relatedBy:NSLayoutRelationEqual  
    toItem:theView attribute:NSLayoutAttributeLeading  
    multiplier:1.0 constant:0.0];
```

```
theConstraint = [theSuperView.leadingAnchor  
    constraintEqualToAnchor:theView.leadingAnchor constant:0.0];
```

Aktivierung

- durch Hinzufügen zu View
- (kleinster) gemeinsamer Superview



Richtig aktivieren

```
NSLayoutConstraint *theConstraint = [NSLayoutConstraint ...];  
UIView *theView = ... // passenden View suchen  
  
[theView addConstraint:theConstraint];
```



```
NSLayoutConstraint *theConstraint = [NSLayoutConstraint ...];  
  
theConstraint.active = YES;
```



```
NSArray *theConstraints = @[ [NSLayoutConstraint ... ] ...];  
  
[NSLayoutConstraint activateConstraints:theConstraints];
```



Richtig deaktivieren

```
NSLayoutConstraint *theConstraint = ...;  
theConstraint.active = NO;
```



```
NSArray *theConstraints = ...;  
[NSLayoutConstraint deactivateConstraints:theConstraints];
```



Effizient aktualisieren

- über View
- Aktualisieren:
`updateConstraints`
- triggern:
`setNeedsUpdateConstraints`

```
-(void)updateConstraints {  
    // Constraints aktualisieren  
    [super updateConstraints];  
}
```

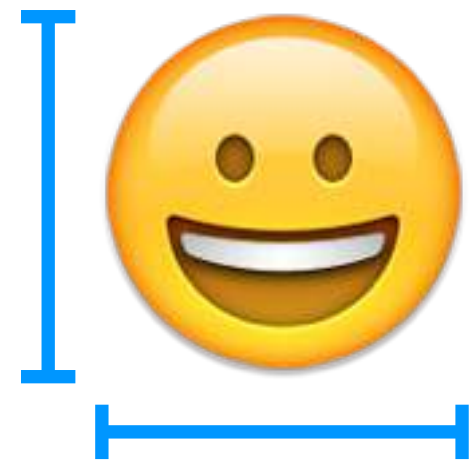
update ≠ create



letzter Aufruf

Intrinsische Größe

UIImageView



UIToolbar

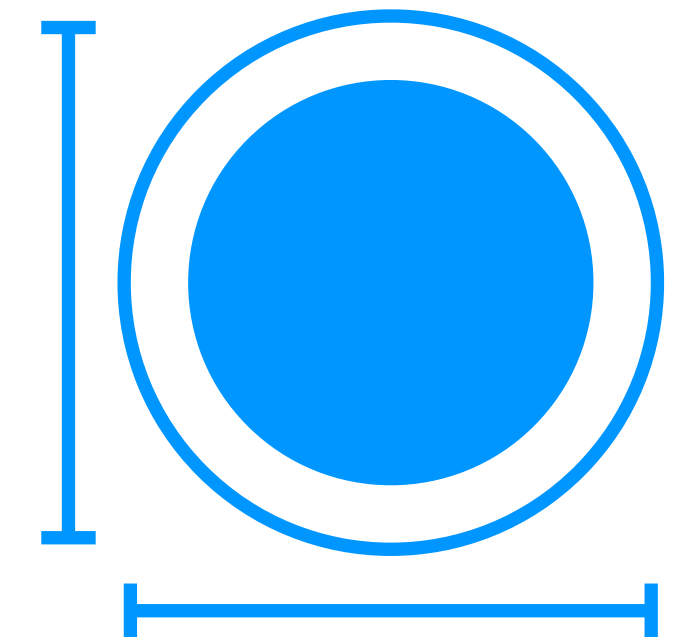


UILabel



Bei der intrinsischen Größe legt der View seine eigene Größe passend zu seinem Inhalt fest.

MyView



Views mit intrinsischer Größe

- Überschreiben: `intrinsicContentSize`
- `UIViewNoIntrinsicMetric`
- bei Änderungen: `invalidateIntrinsicContentSize`

Intrinsische Größe implementieren

```
@implementation BannerView

-(CGSize)intrinsicContentSize {
    CGFloat theHeight = [self heightForFont:self.font];

    return CGSizeMake(UIViewNoIntrinsicMetric, theHeight);
}
```

Intrinsische Größe implementieren

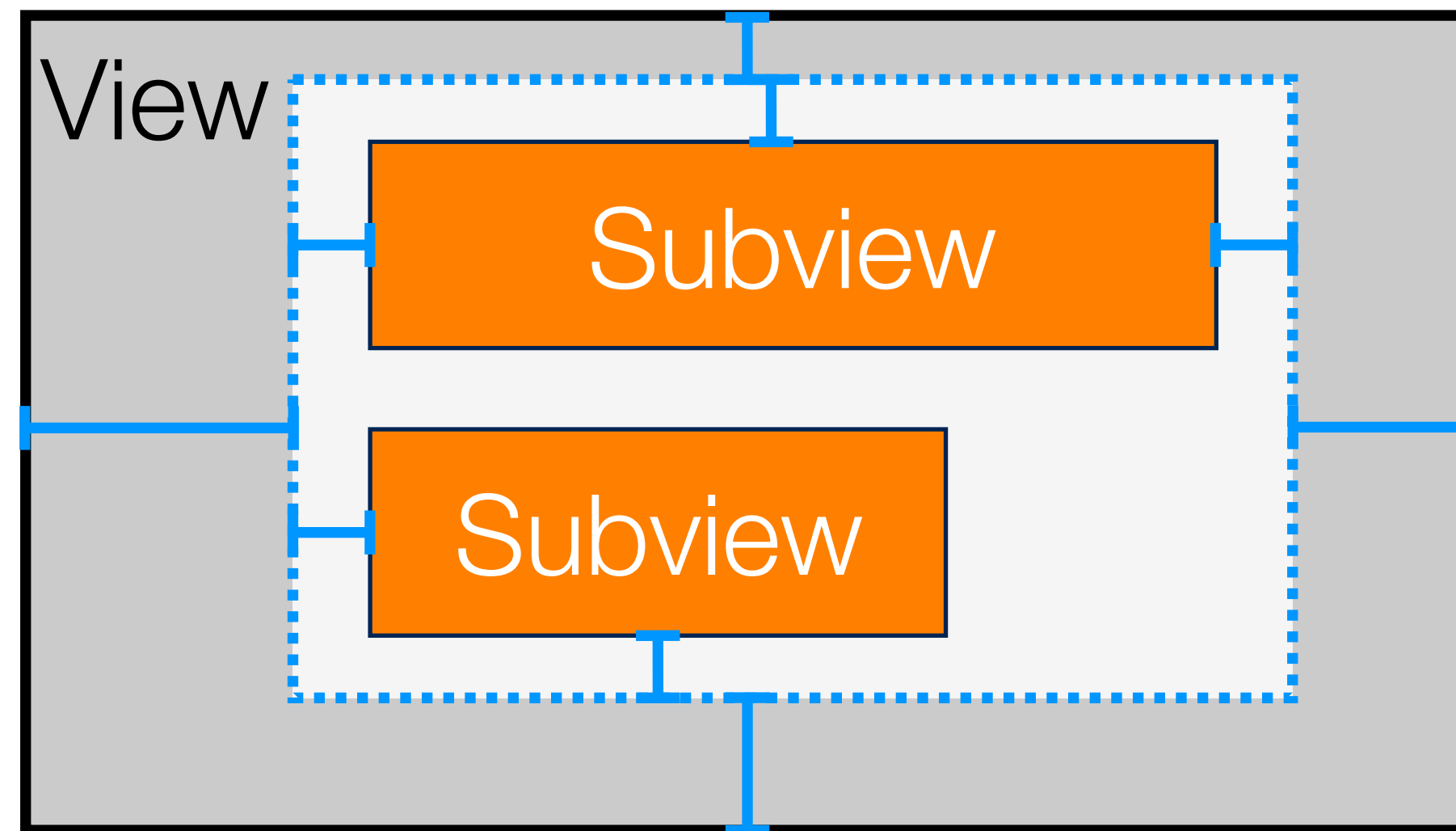
```
@implementation BannerView

-(CGSize)intrinsicContentSize {
    CGFloat theHeight = [self heightForFont:self.font];

    return CGSizeMake(UIViewNoIntrinsicMetric, theHeight);
}

-(void)setFont:(UIFont *)inFont {
    _font = inFont;
    [self invalidateIntrinsicContentSize];
}
```

Feature: Layout-Margins



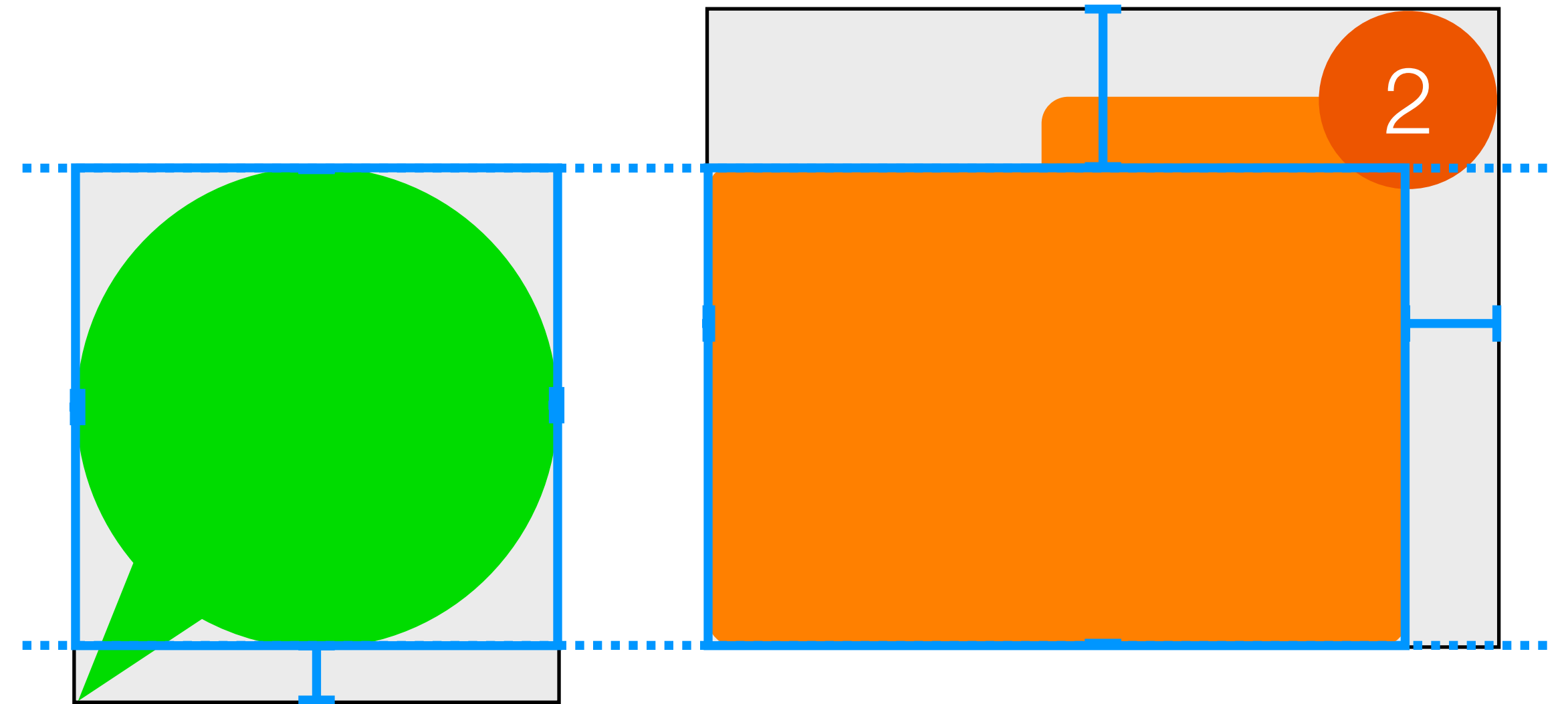
Layout-Margins

=

Padding

Feature: Alignment-Rects

- Ausrichtung am Alignment-Rect
- Vergrößerung der Fläche
 - zum Zeichnen
 - für Events



Alignment-Rect-Insets

```
-(UIEdgeInsets)alignmentRectInsets {  
    //                top    left    bottom    right  
    return UIEdgeInsetsMake(12.0, 16.0, 12.0, 0.0);  
}
```

Alignment-Rect-Insets

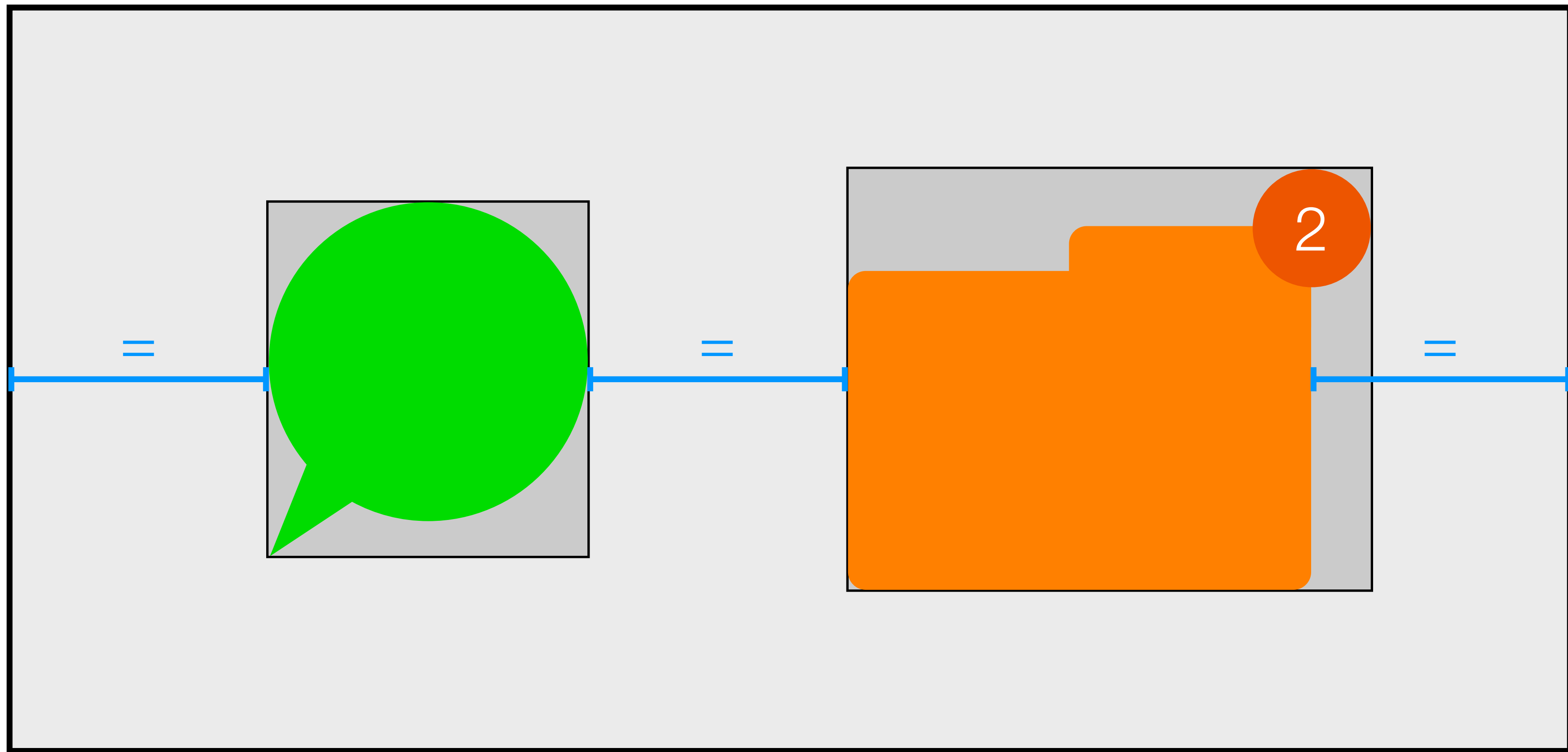
```
-(UIEdgeInsets)alignmentRectInsets {  
    //                top    left    bottom    right  
    return UIEdgeInsetsMake(12.0, 16.0, 12.0, 0.0);  
}  
  
-(void)drawRect:(CGRect)inRect {  
    CGRect theBounds = self.bounds;  
    UIEdgeInsets theInsets = [self alignmentRectInsets];  
    CGRect theInnerRect = UIEdgeInsetsInsetRect(theBounds, theInsets);  
  
    ...  
}
```

Layout-Guides

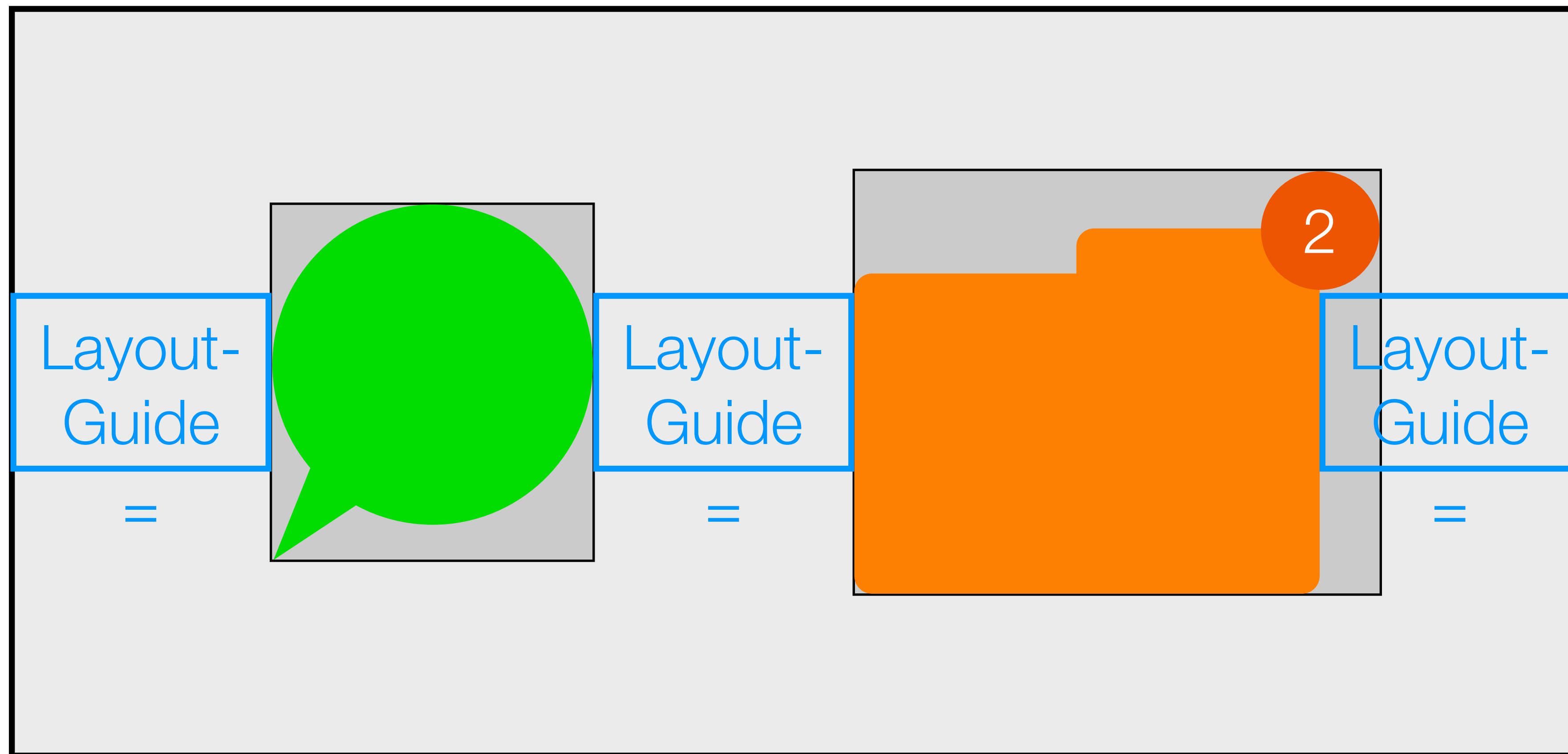
- rechteckiger Bereich
- ersetzen Dummy-Views
- nicht im Interface Builder 😞



Layout-Guides



Layout-Guides

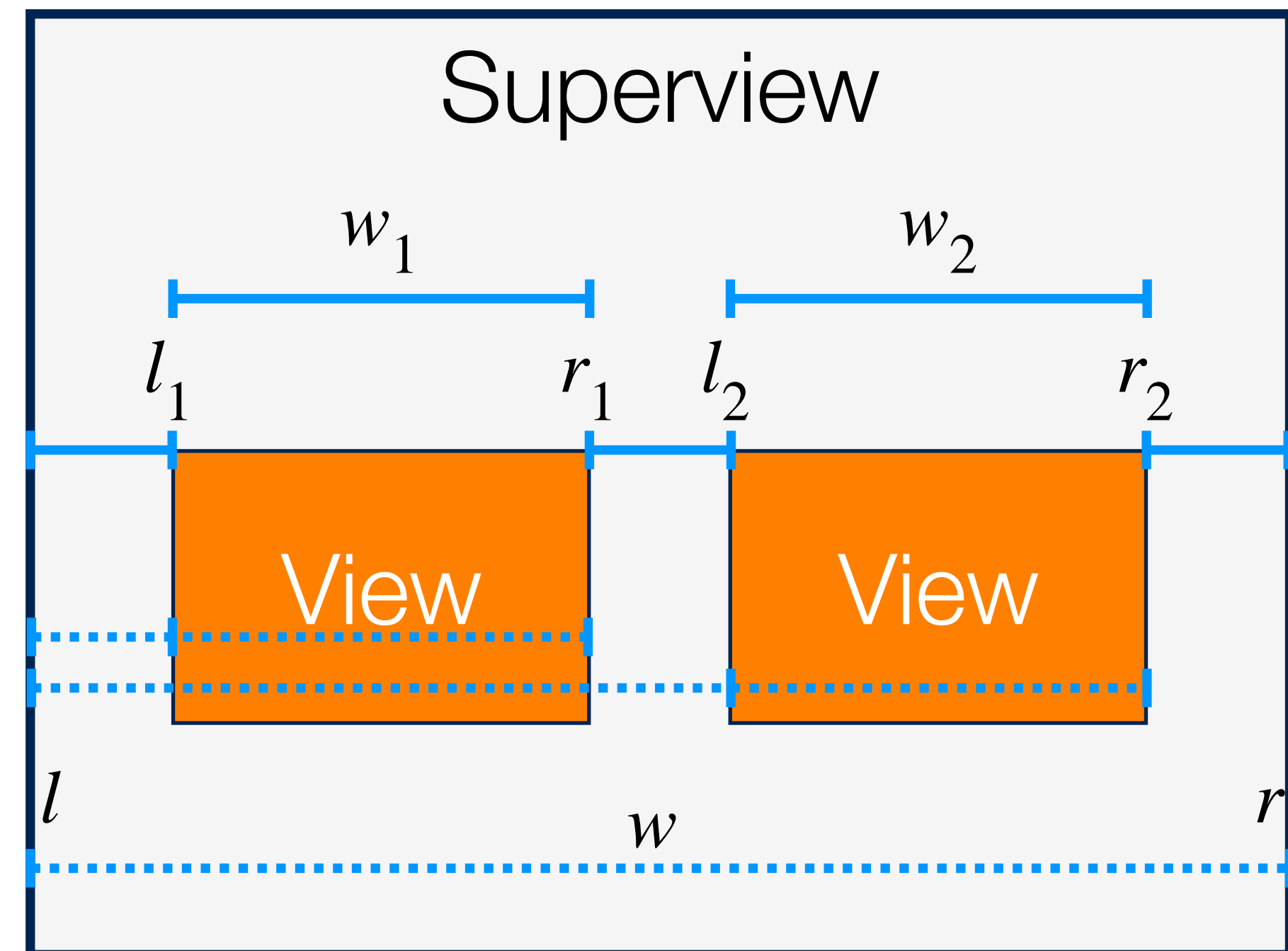


Demo

Wie funktioniert Auto-Layout?

Layout-Regeln auswerten

1. $l_1 = 1 \times l + 20$
2. $l_2 = 1 \times r_1 + 20$
3. $r_2 = 1 \times r - 20$
4. $w_2 = 1 \times w_1$
5. $r_1 = 1 \times l_1 + w_1$
6. $r_2 = 1 \times l_2 + w_2$
7. $r = 1 \times l + w$
8. $l = 0$



Gleichungssystem lösen

- 9 Unbekannte
- 8 Gleichungen
- Breite w ist gegeben



CSP-Solver

- CSP = Constraint Satisfaction Problem
- Cassowary:
<http://constraints.cs.washington.edu/cassowary/>
- CSP-Solver können mehr

Mehrdeutigkeit und Prioritäten

CSP-Solver können mehr

- Ungleichungen
- Mehrdeutigkeiten mit Prioritäten
- Minimierung

CSP-Löser können mehr

Auto-Layout



Wenn der Text neben dem Bild kurz ist, reichen die Constraints aus.

.....
Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed diam
voluptua. At vero eos et accusam et
justo duo dolores et ea rebum. Stet clita
kasd gubergren, no sea takimata
sanctus est Lorem ipsum dolor sit amet.
Lorem ipsum dolor sit amet, consetetur

Auto-Layout



Wenn der Text neben dem Bild zu lang ist, reichen die Constraints aber nicht aus. Dann überlappt der Teaser den Haupttext.

.....
Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed diam
voluptua. At vero eos et accusam et
justo duo dolores et ea rebum. Stet clita
kasd gubergren, no sea takimata
sanctus est Lorem ipsum dolor sit amet.
Lorem ipsum dolor sit amet, consetetur

Auto-Layout

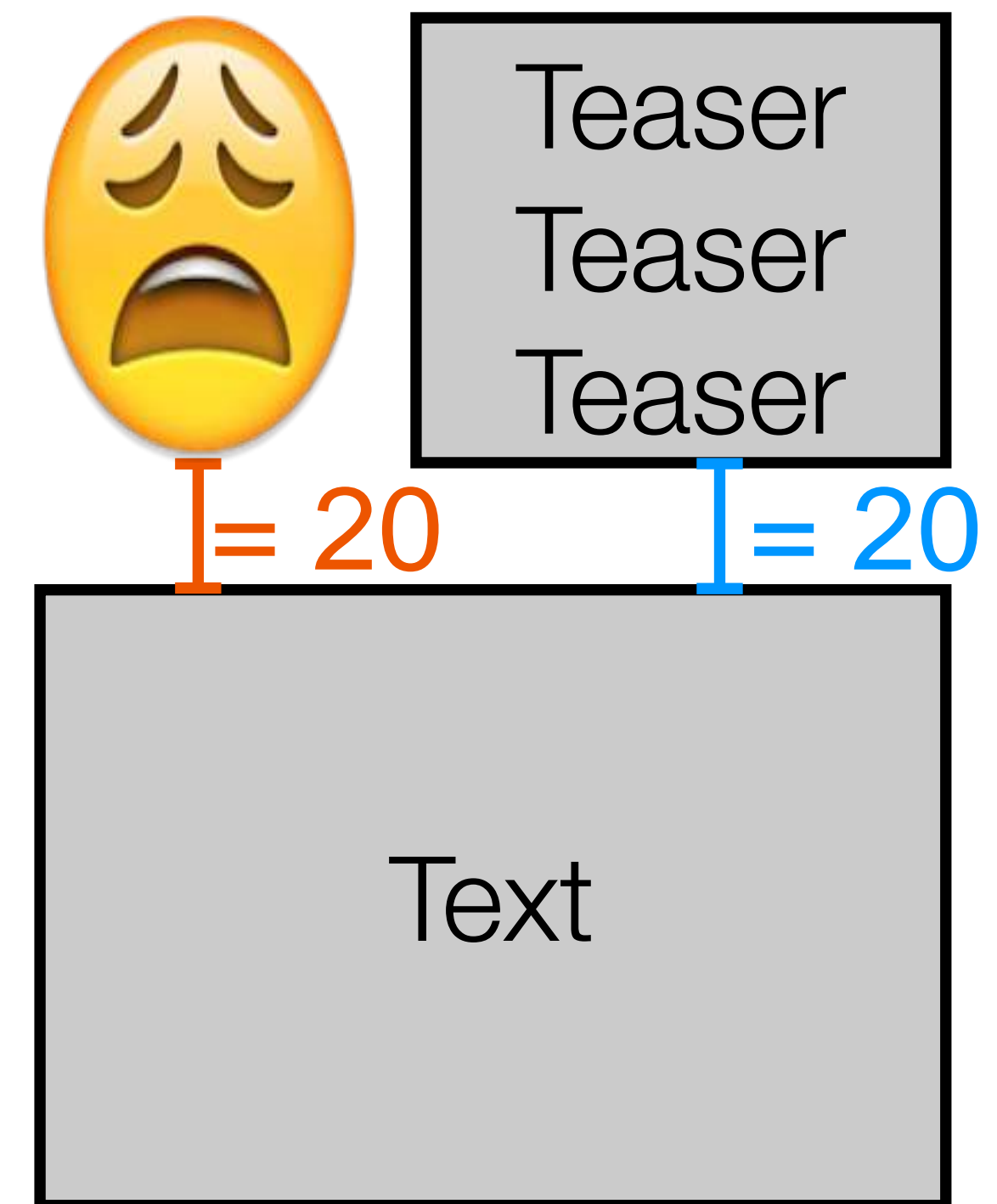
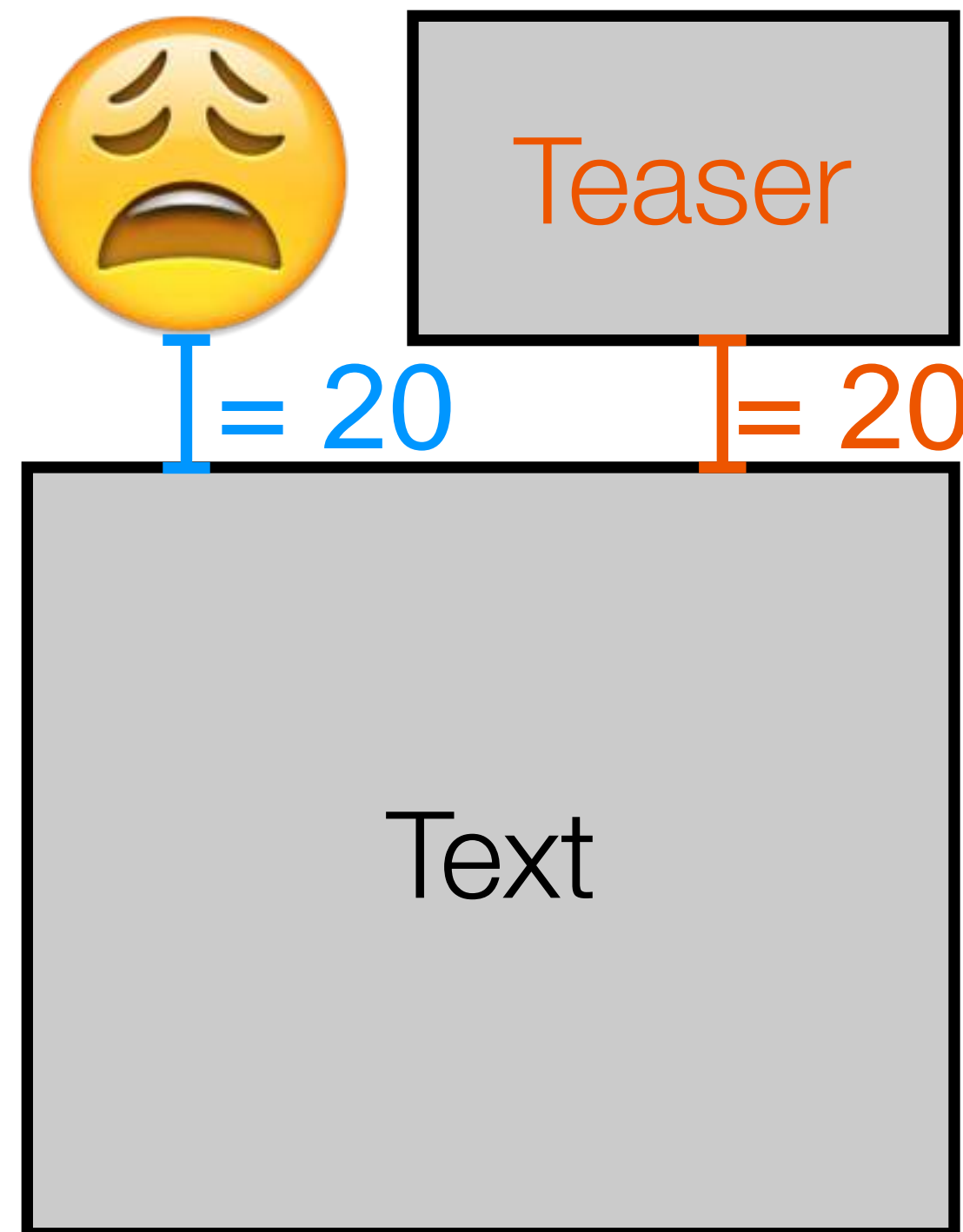


Über Ungleichungen und Prioritäten lassen sich Varianten beschreiben, damit sich das Layout bestmöglich an unterschiedliche Größen der Views anpassen kann.

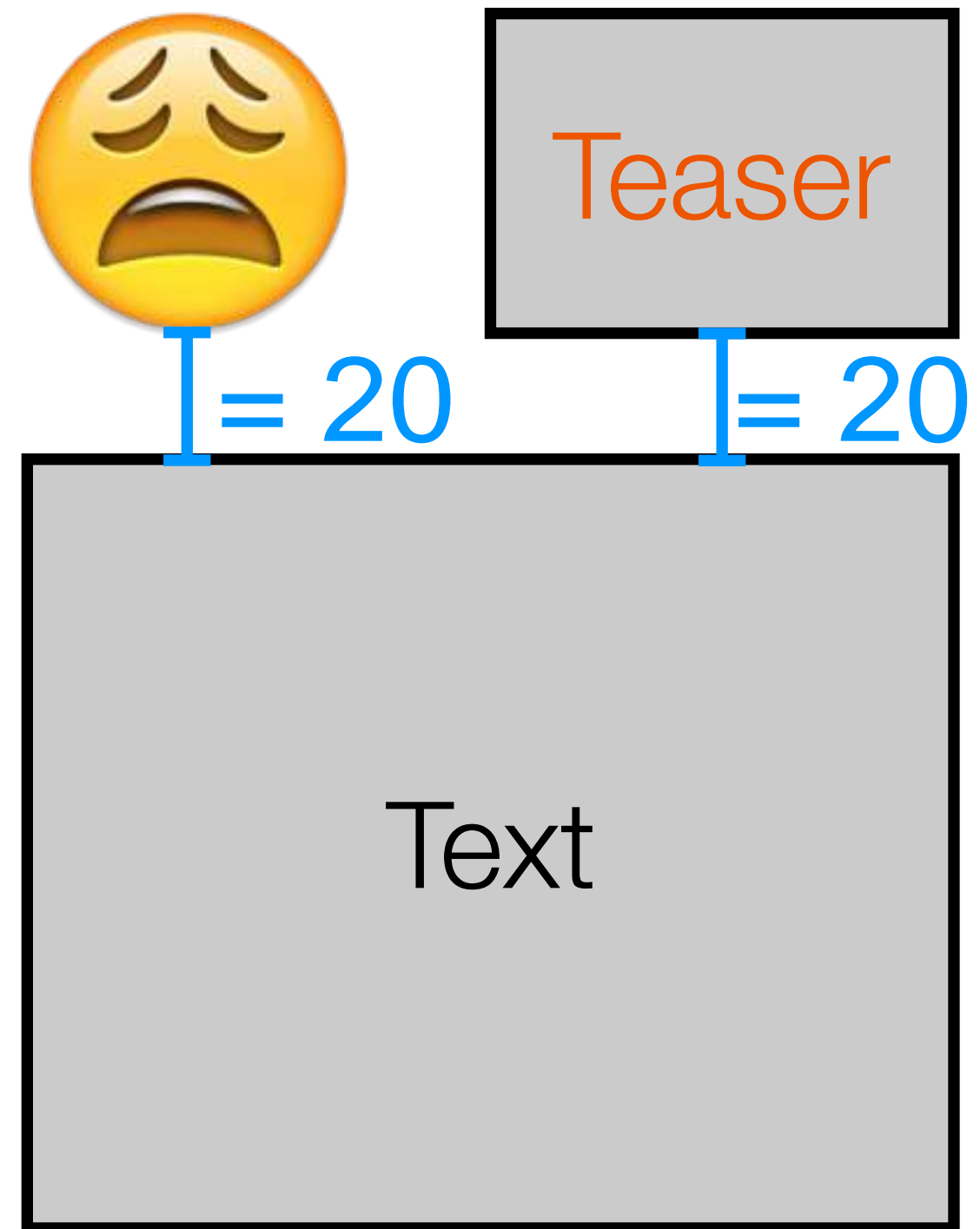
.....
Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed diam
voluptua. At vero eos et accusam et

Gleichungen

nicht flexibel genug

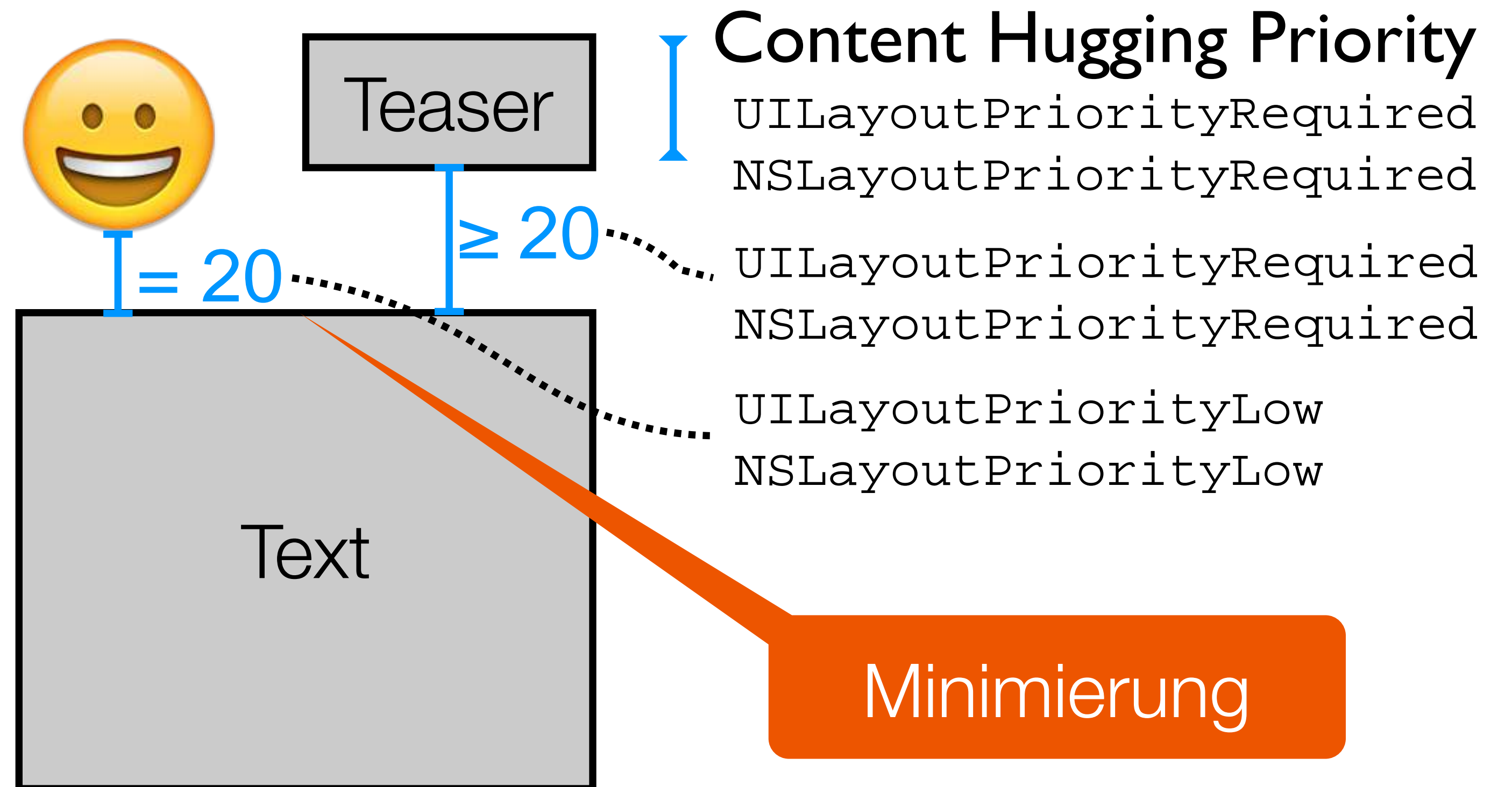


Ungleichungen & Prioritäten



Ungleichungen & Prioritäten

- als Mindestabstand festlegen
- Priorität für intrinsische Höhe
- und Abstände



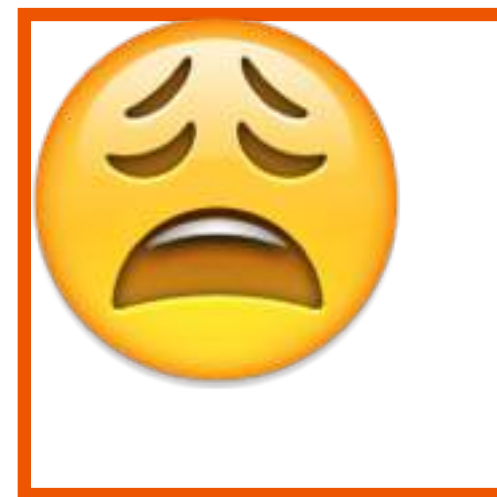
Prioritäten für intrinsische Größe

Rahmen passend



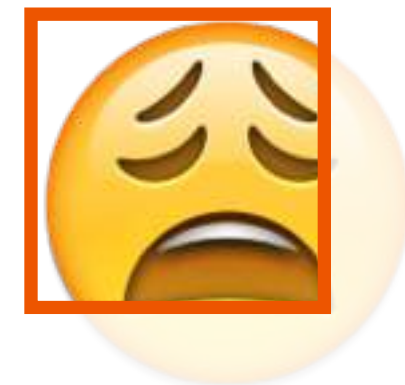
`UILayoutPriorityRequired`
`NSLayoutPriorityRequired`

Rahmen größer?



Content Hugging Priority anpassen

Rahmen kleiner?

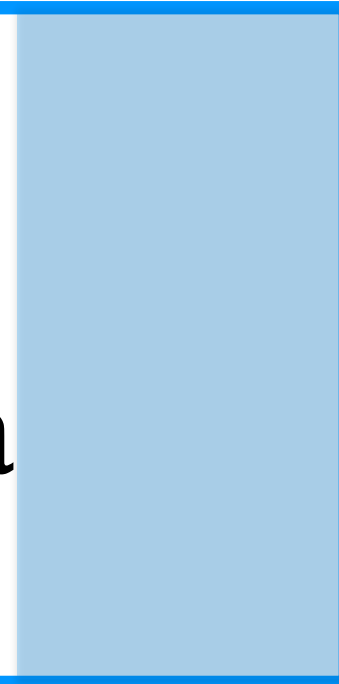


Content Compression Resistance Priority
anpassen

Content Hugging



Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod
tempor invidunt ut labore et dolore magna
aliqua.



UILayoutPriorityRequired
NSLayoutPriorityRequired



UILayoutPriorityLow
NSLayoutPriorityLow

Demo

Auto-Layout und Auto-Sizing

Freie Positionierung

- Möglich: `constant` änderbar
- Constraints merken
- Umständlich gegenüber Auto-Sizing

Auto-Sizing einschalten

- Explizit:

`translatesAutoresizingMaskIntoConstraints = YES;`

- Implizit:



Fehlerquelle

- Rootviews aus Storyboards & NIBs
- Views aus Programmcode

Auto-Sizing

```
UIView *theView = [UIView new]; // Auto-Sizing implizit an  
  
// Positionierung über Rahmen  
theView.frame = ... // Größe & Ursprung  
  
// oder  
theView.bounds = ... // Größe  
theView.center = ... // Mittelpunkt
```


Auto-Sizing & -Layout mischen

- Auto-Layout: mehrere Durchläufe
- Auto-Sizing in Zyklus integrieren
- Einstieg: `layoutSubviews`

Demo

Layout-Änderungen und Transformationen

Layout-Änderungen

- Constraints anlegen und löschen 
- Aktivieren und Deaktivieren
- Konstanten ändern

Animierte Änderungen

- Konstanten ändern
- Aktivieren und Deaktivieren
- Layout-Änderung animieren

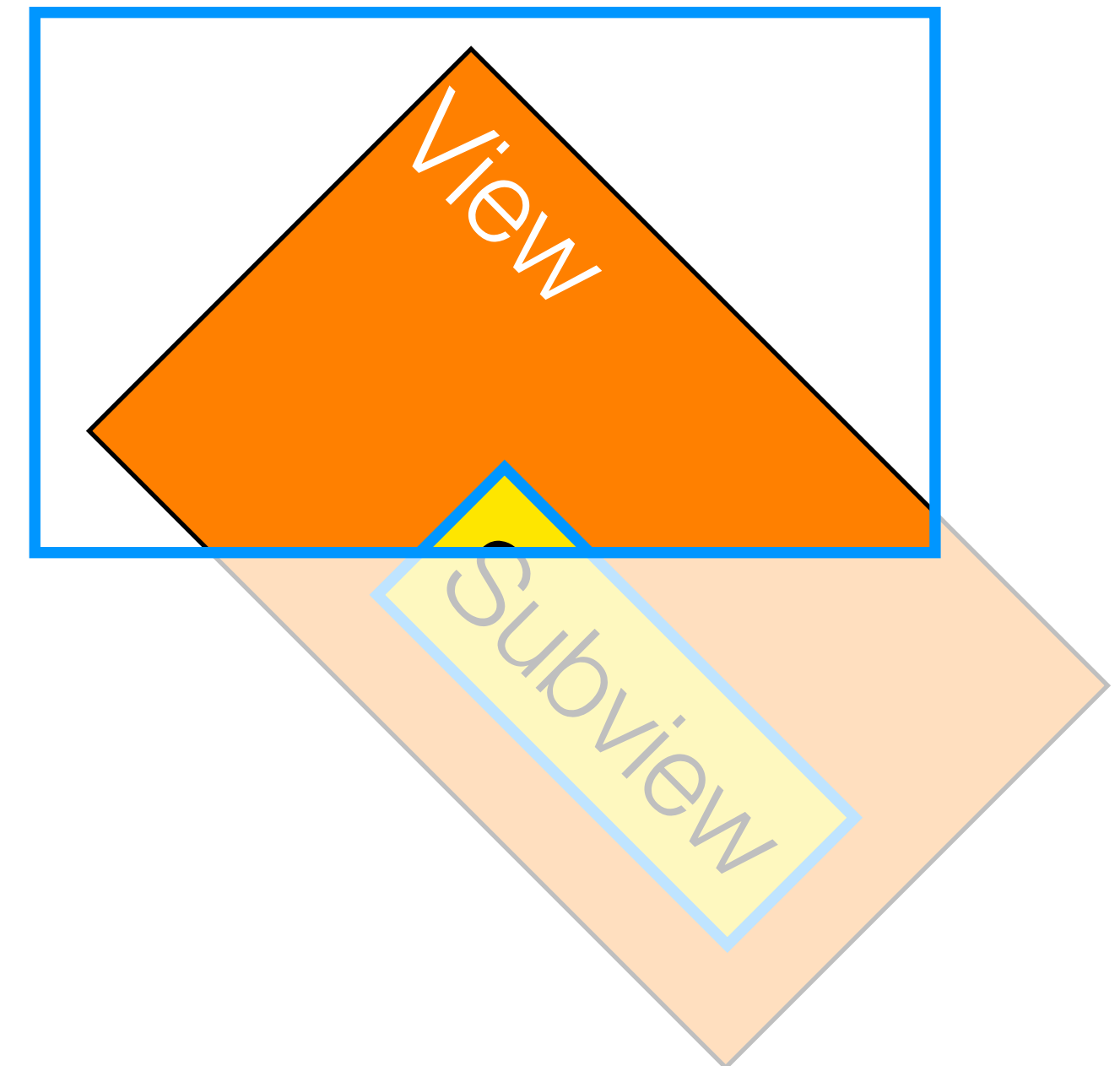
```
NSLayoutConstraint *theConstraint = ...;

theConstraint.constant = 100.0;
self.leftConstraint.active = NO;
[UIView animateWithDuration:0.5
    animations:^(
        UIView *theView = ...;
        [theView layoutIfNeeded];
    )];
```

gemeinsamer
Superview

Transformationen

- beeinflussen nicht Auto-Layout
- Animationen
 - verhalten sich manchmal unerwartet 😞
 - Core Animation hilft 😊

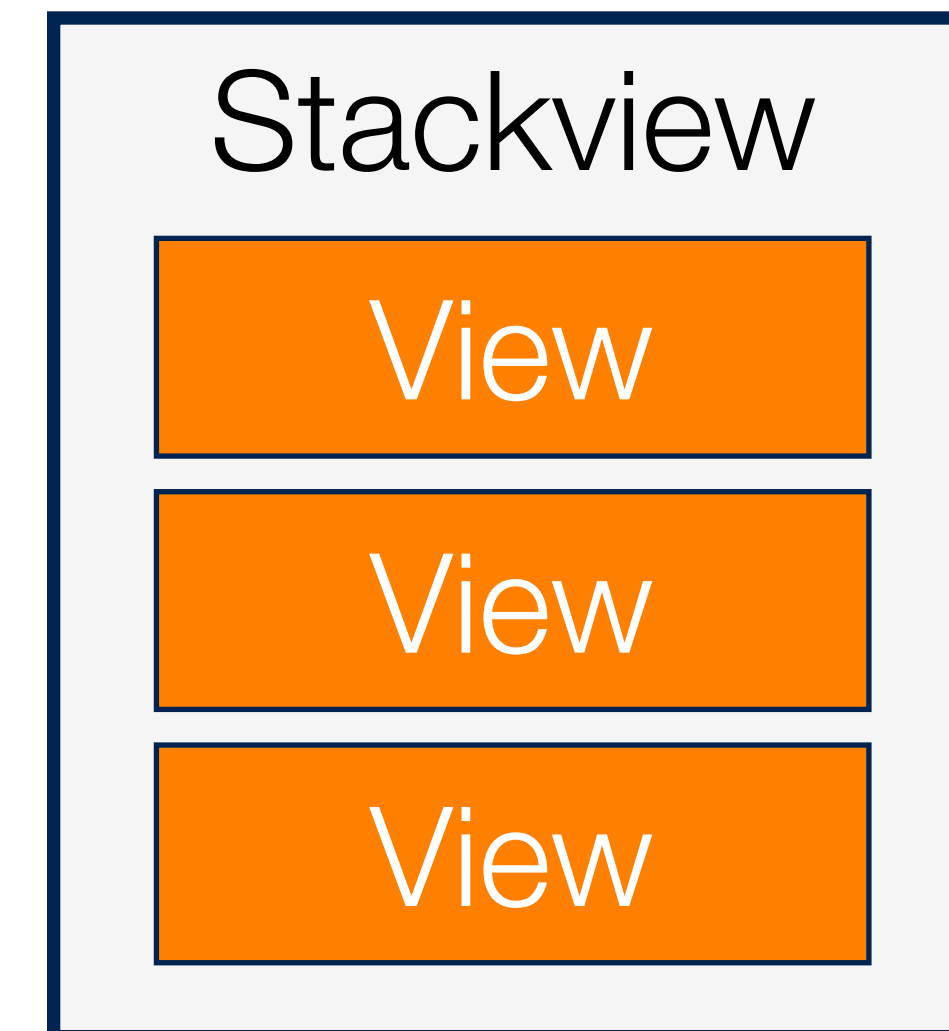


Demo

Stackviews

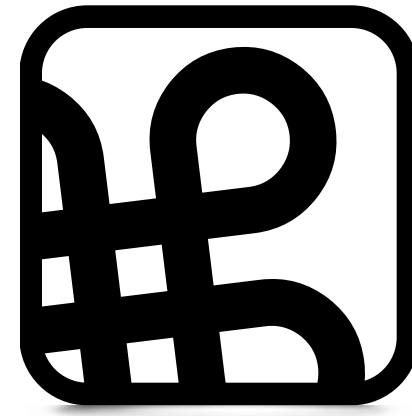
Stackviews

- automatische Anordnung
- horizontal oder vertikal
- Views verstecken



Demo

Fragen?



Macoun