

Introduction to Flex Components

There are many ways...

- **creating a composite MXML component**
 - **extending an existing component**
 - **extending UIComponent**
- ...and this is only for the visual components**

When creating MIXML components

- you can use either absolute positioning
- or relative positioning
- and even enhanced constraints

To accommodate your children

- **there are different types of containers**
- **layout containers (Canvas, VBox, HBox, Tile, etc)**
- **list containers (DataGrid, List)**
- **and a repeater (which is not a container)**

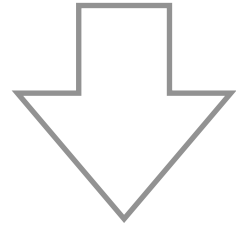
Children loved unequally

- all components that extend *mx.core.Container* make distinctions between two types of children
- *children* (a.k.a. content children) - the children defined inside the MXML of the container
- *rawChildren* - children like borders, background, highlight halo, ..., and possibly the content children too
- the *rawChildren* is an *mx.core.IChildList*
- but what happens when the content children are added to a *content pane*?

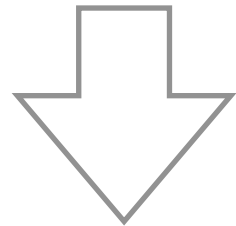
To extend the
UIComponent we need to
know the...

UIComponent Life-cycle

Initialization Phase

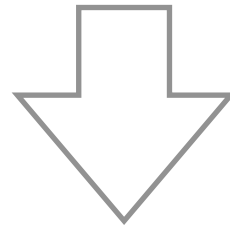


Update Phase

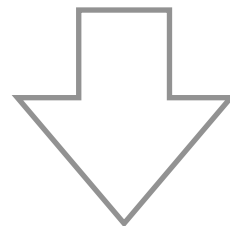


Destruction Phase

Initialization Phase



Update Phase

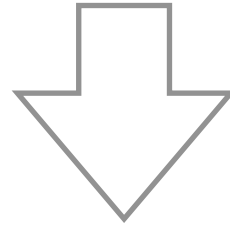


Destruction Phase

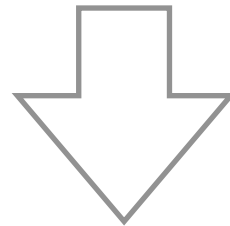
Construction Stage

- **the constructor is called**
- **don't create display objects in the constructor**
- **set initial values of component properties**
- **add event listeners**
- **initialize other objects**

Initialization Phase



Update Phase



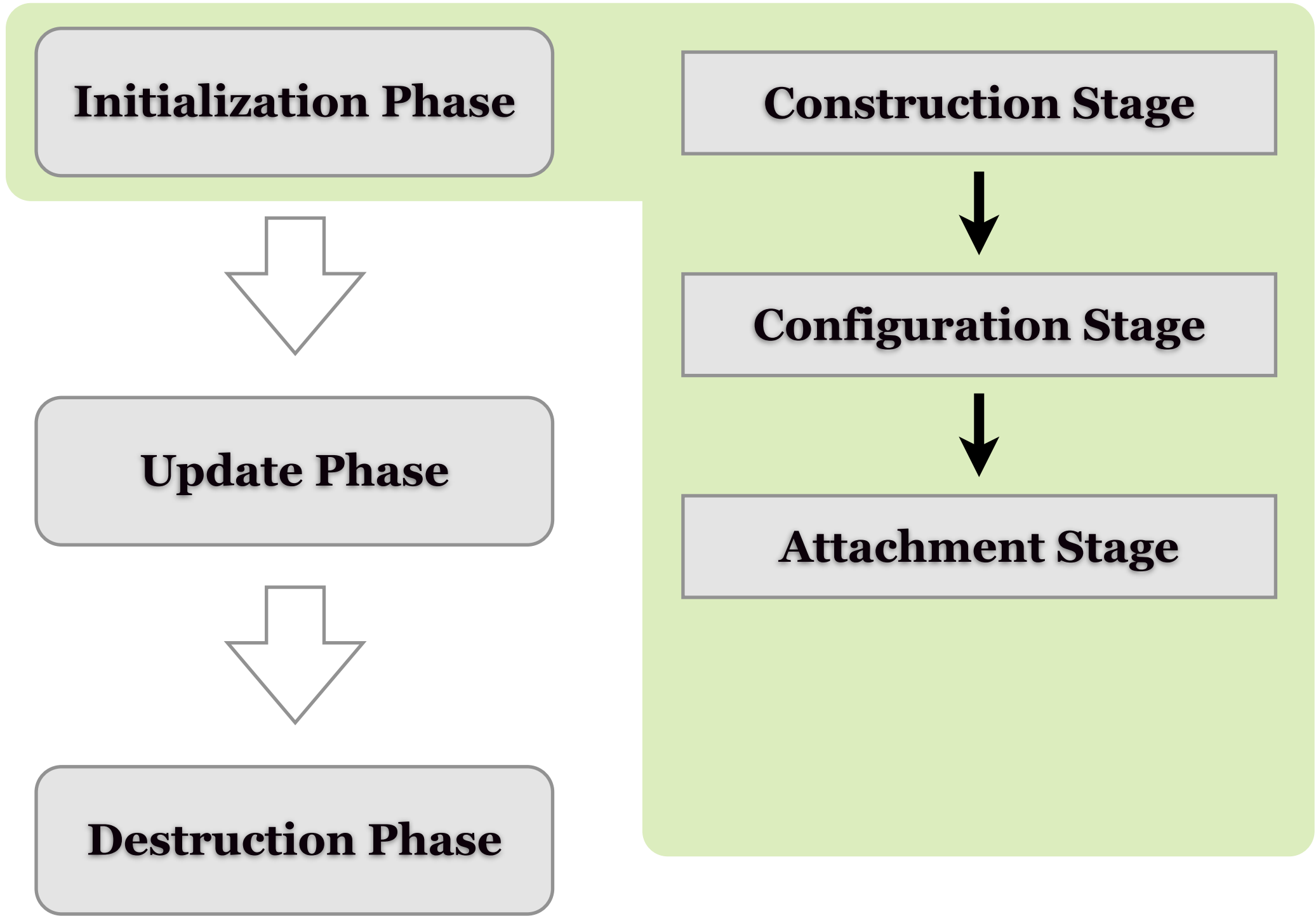
Destruction Phase

Construction Stage

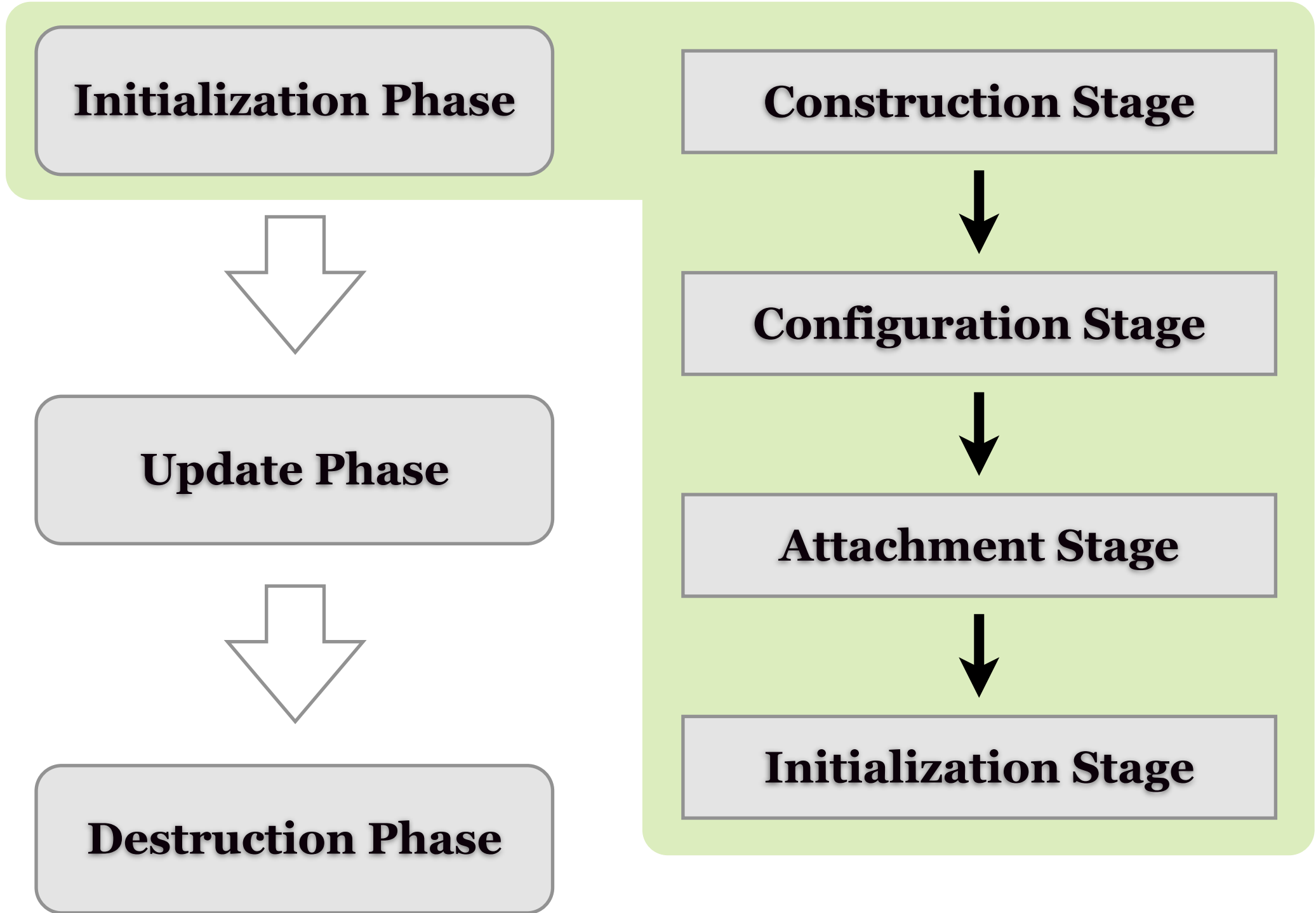


Configuration Stage

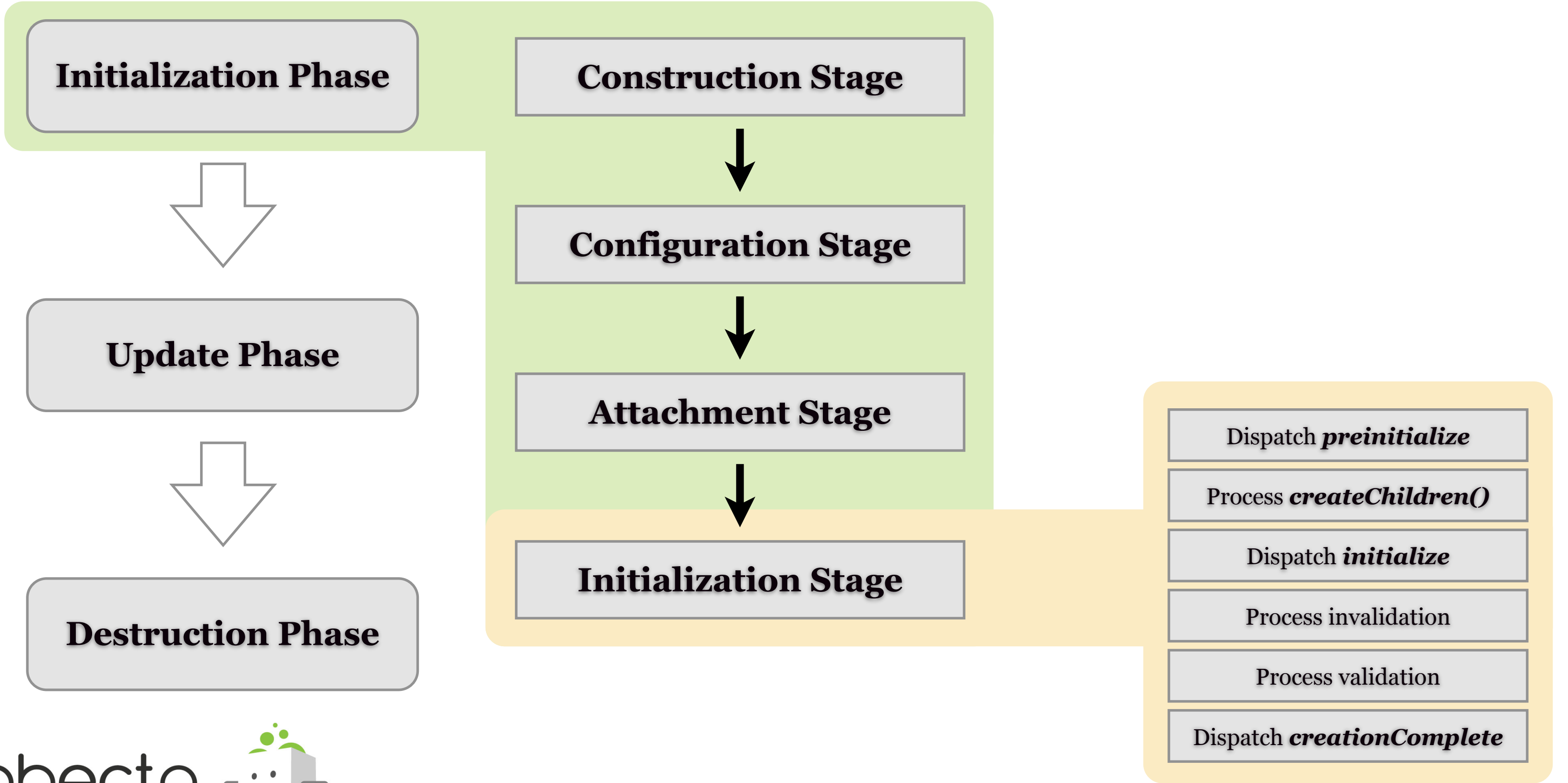
- **configuring the newly-constructed instance**
- **properties**
- **styles**
- **event handlers**



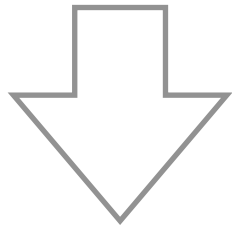
- the component is added to the display list
- now the component has a parent
- calls *initialize()* automatically to move to the initialization stage



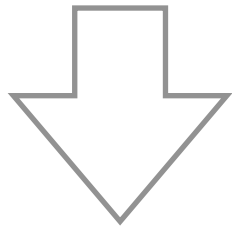
- create children objects
- sizing and positioning
- applying the configured properties and styles



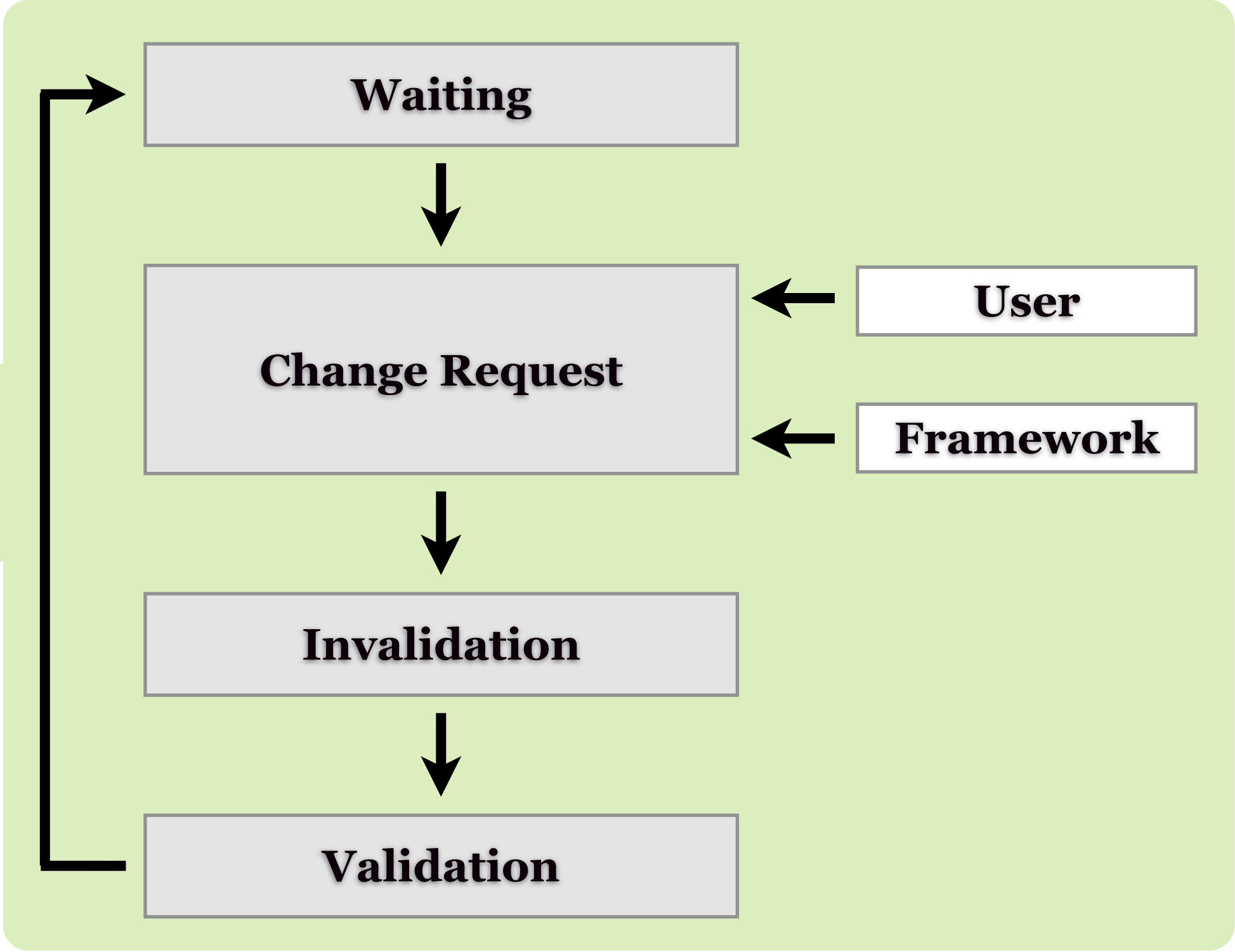
Initialization Phase



Update Phase



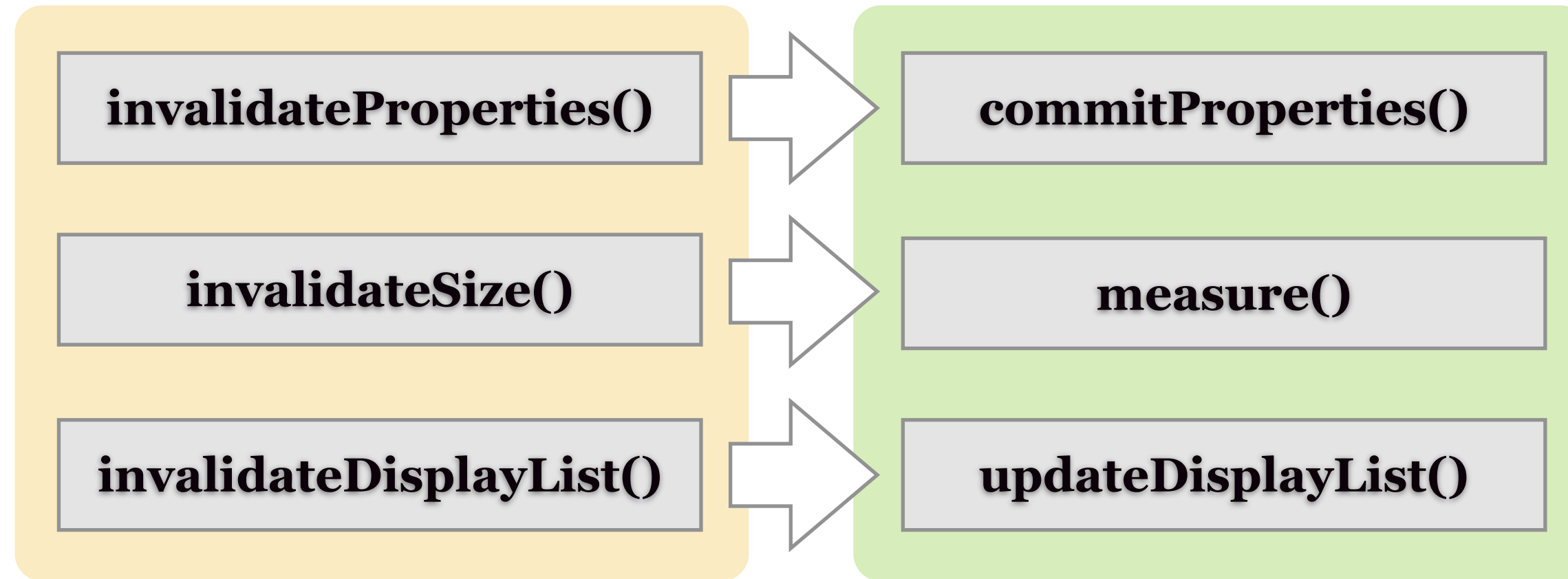
Destruction Phase



The Invalidation Mechanism

Invalidation

Validation



- avoids sequential coupling, or at least such coupling can be located only in the *commitProperties()* method instead of forcing a protocol to the component's clients
- some kind of optimization - prevents unnecessary work in case of setting the property multiple times

States & Transitions

Flex states

```
<mx:states>
  <mx:State name="collapsed"/>
  <mx:State name="expanded"
    basedOn="null"
    enterState="handler"
    exitState="handler">
    <!-- State Overrides -->
  </mx:State>
</mx:states>
```

**you can also use ActionScript
to define states**

- **states can be based on other states, thus defining a hierarchy of (visual) inheritance**
- **states are based on the root state by default**

Visual transitions

```
<mx:transitions>
  <mx:Transition
    fromState="*"
    toState="*">
    <!-- Some Effect -->
  </mx:Transitions>
</mx:transitions>
```

**you can also use ActionScript
to define transitions**

- **provide mechanism for smooth visual change between states**
- **this smooth visual change is a Flex Effect**
- **the begin and end values of the effect are determined from the current state, the properties you've specified in the effect and the destination state**

Summary

- **there are many ways to create and extend Flex Components**
- **absolute and relative positioning**
- **the UIComponent life-cycle and the LayoutManager phases**
- **overview of the invalidating mechanism**
- **states and transitions**
- **...we won't focus on effects, validators, formatters, data descriptors**