Design Doc Template Terminology

*Status: (Draft)*

*Authors: misko@google.org*

*This document is published to the web as part of the public* [*Angular Design Docs*](https://drive.google.com/#folders/0BxgtL8yFJbacUnUxc3l5aTZrbVk) *folder*

# Objective

When discussing how components work in angular the same item can take on a different name/role depending on context. This tries to clear the confusion.

# Background

Many discussions of angular workings have shown that terminology is has often got in way of clear explanation. The issue is that the same component can be called different thing depending on the context. For example an Element can be a LightDOM, ShodowDom and a WebComponent at the same time.

# Prior Art

Angular

WebComponents

Polymer

# Detailed Design

The Actors:

* **Component**: The upper component for the purposes of the discussion. (Keep in mind that the same Component may be a SubComponent from a different part of the system)
* **SubComponent**: The lower component which is used to build up the component.
* Template: The template which the component uses. There are two templates:
  + **LightDOM**: Elements inside the component.
  + **ShadowDOM**: Elements from which the component is built up from and which are hidden from the view due to ShadowDOM encapsulation.

## Case1: Simple Component

|  |  |
| --- | --- |
| <div>  <**border**>  <span>Hello World</span>  </**border**>  </div> | <div class="thick">  <content></content>  </div> |

In its simplest form a component (such as border) can attach a ShadowDOM to itself. Notice that the ShadowDOM lives parallel to the component itself.

## Case2: Component and SubComponent in ShadowDOM

|  |  |
| --- | --- |
| <div>  <**border**>  <span>Hello World</span>  </**border**>  </div> | <div class="thick">  <**zippy**>  <content></content>  </**zippy**>  </div> |

This is a more complex case where the Component is built up from other SubComponents. (Keep in mind that the SubComponents) can have other components inside itself with its own ShadowDOM not shown)

The important thing to realize here is that the author at the time of writing the Border component chose to implement it using a Zippy component. For this reason it is fair for the Border component to make assumptions about Zippy. For example what events it fires, what API it exposes, and whether it is WebComponent or Angular component.

## Case2: Component and SubComponent in LightDOM

|  |  |
| --- | --- |
| <div>  <**border**>  <**zippy**>  <span>Hello World</span>  </**zippy**>  </**border**>  </div> | <div class="thick">  <content></content>  </div> |

In this case the zippy SubComponent was placed inside the LightDOM of the Border component.

At the time of writing of the Border component, the Border Component author did not know that Zippy will be included. For this reason the Border Component can not make any assumptions about any SubComponents. The implication of this is that the Border component can not depend on Zippy component API. For this reason there needs to be a mechanism to bridge the API for SubComponent with that of the Component.

The author of the template on the other hand did know that they were combining Border and Zippy and therefore they need a way to bridge the API miss-match. This means a way to hook up the events of one to method invocation or events for the other. This part is still to be designed. DESIGN DOC NEEDED.

## Summary

|  |  |
| --- | --- |
| **SubComponent inside Component's** | **API / Events** |
| Shadow DOM | At the time of writing Component, SubComponent type and APIs / Events are known, and can therefore be used. |
| Light DOM | At the time of writing Component, it is not clear that SubComponent will be added to the Light DOM, hence no assumptions can be made about the API or Events. |
| Light DOM Collaborator | A unique situation arises when Component/SubComponent are written as a pair and are meant to be used together exclusively. In this case the Component can make assumptions about SubComponent even though the Component does not control the light DOM.  Example:  <tabs>  <pane name="Overview">...</pane>  <pane name="Details">...</pane>  <pane [ng-repeat|p]="panes"  [name]="p.name">...</pane>  </tabs> |

# Caveats

You may need to describe what you did not do or why simpler approaches don't work. Mention other things to watch out for (if any).

# Security Considerations

How you’ll be secure

# Performance Considerations / Test Strategy

How you’ll be fast.

# Work Breakdown

Description of development phases and approximate time estimates.