

View Flattening

⚠ CAUTION

This document refers to the architecture of the new renderer, [Fabric](#), that is in active roll-out.

View Flattening is an optimization by the React Native renderer to avoid deep layout trees.

The React API is designed to be declarative and reusable through composition. This provides a great model for intuitive development. However, in implementation, these qualities of the API lead to the creation of deep React Element Trees, where a large majority of React Element Nodes only affect the layout of a View and don't render anything on the screen. We call these types of nodes **"Layout-Only" Nodes**.

Conceptually, each of the Nodes of the React Element Tree have a 1:1 relationship with a view on the screen, therefore rendering a deep React Element Tree that is composed by a large amount of "Layout-Only" Node leads to poor performance during rendering.

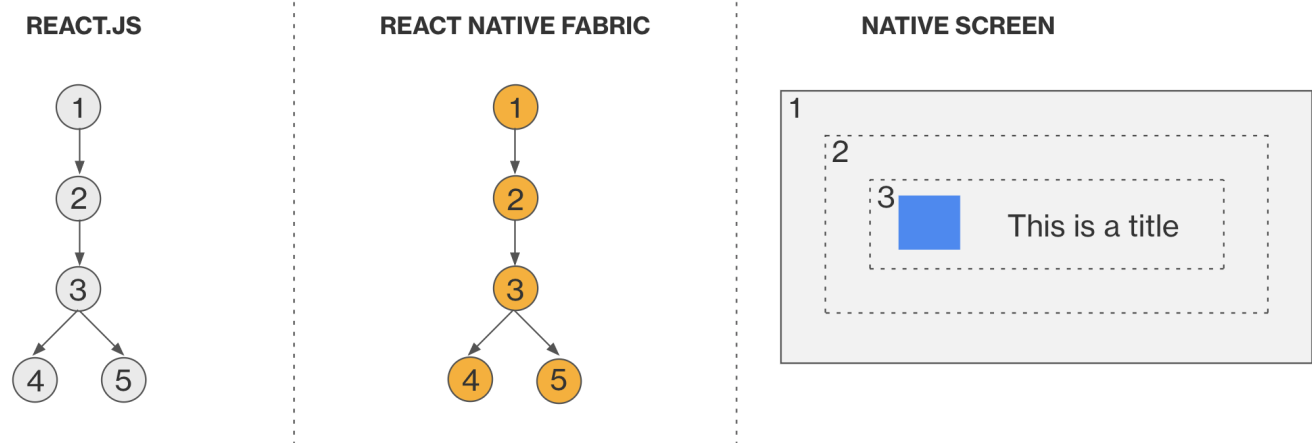
Here is a common use case that is affected by the cost of "Layout Only" views.

Imagine you want to render an image and a title that is handled by the `TitleComponent`, and you include this component as a child of the `ContainerComponent` that has some margin styles. After decomposing the components, the React code would look like this:

```
function MyComponent() {
  return (
    <View>                                // ReactAppComponent
      <View style={{margin: 10}} /> // ContainerComponent
      <View style={{margin: 10}}> // TitleComponent
        <Image {...} />
        <Text {...}>This is a title</Text>
      </View>
    </View>
  )
}
```

```
);  
}
```

As part of the render process, React Native will produce the following trees:



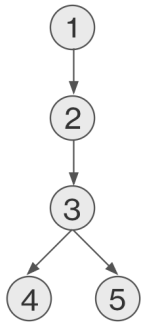
Note that the Views (2) and (3) are “Layout Only” views, because they are rendered on the screen but they only render a `margin` of 10 px on top of their children.

To improve the performance of these types of React Element Trees, the renderer implements a View Flattening mechanism that merges or flattens these types of Nodes, reducing the depth of the host view hierarchy that is rendered on the screen. This algorithm takes into consideration props like: `margin`, `padding`, `backgroundColor`, `opacity`, etc.

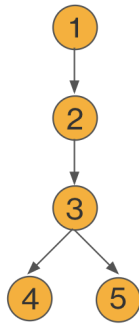
The View Flattening algorithm is integrated by design as part of the diffing stage of the renderer, which means that we don’t use extra CPU cycles to optimize the React Element Tree flattening these types of views. As the rest of the core, the View flattening algorithm is implemented in C++ and its benefits are shared by default on all supported platforms.

In the case of the previous example, the Views (2) and (3) would be flattened as part of the “diffing algorithm” and as a result their styles will be merged into the View (1):

REACT.JS



REACT NATIVE FABRIC




NATIVE SCREEN



It is important to note that this optimization allows the renderer to avoid the creation and render of two host views. From the user's perspective there are no visible changes on the screen.

Is this page useful?  

 Edit this page

Last updated on **Mar 10, 2022**