Visualizing Shadow DOM Concepts

By Eric Bidelman

Engineer @ Google working on web tooling: Headless Chrome, Puppeteer, Lighthouse

Shadow DOM is a difficult topic to wrap your head around. It's just complex. It introduces unfamiliar concepts that we're not used to on the web. Shadow boundaries, styling scoping, event retargeting, insertion points, shadow insertion points, host nodes, distributed nodes,...the lingo goes on and on.
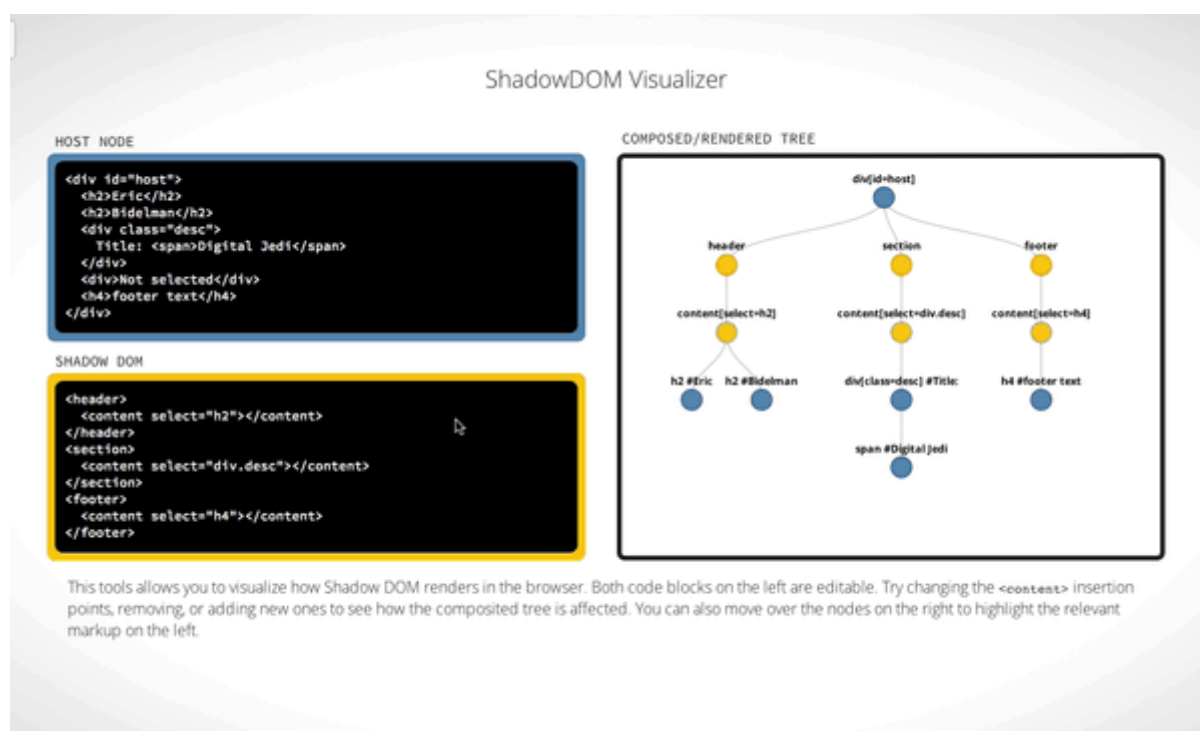
DEMO

One thing that's conceptually taxing about Shadow DOM is the way your final product (DOM) is rendered by the browser. Nodes from the host node are magically swizzled into a ShadowRoot's insertion points, yet logically, still remain in the host node. Weird! So at render time, they appear as part of the shadow tree and not the original host. How this rendering takes place is one of the most confusing pieces of Shadow DOM.

A few days ago, I released a tool I've been working on called Shadow DOM Visualizer to help lessen the learning curve.

It allows you to visually see how Shadow DOM renders in the browser, something DevTools lacks today. Both black code blocks on the left are editable. Try changing the `<content>`

insertion points, removing, or adding new ones to see how the composited (rendered) tree is affected on the right.

Mouse over the nodes in the graph to highlight the relevant markup on the left. Yay for d3.js ☐! Blue nodes are coming from the host node. Yellow nodes come from the Shadow DOM. `<content>` insertion points are the bridge between the two worlds. Because they're logically in the Shadow DOM, they're colored yellow. Their blue border indicates that they invite blue host nodes into the rendering party.

Shadow DOM is available in Chrome 25 and the `<template>` element is available in Chrome 26 (although you only need the first to try the demo).