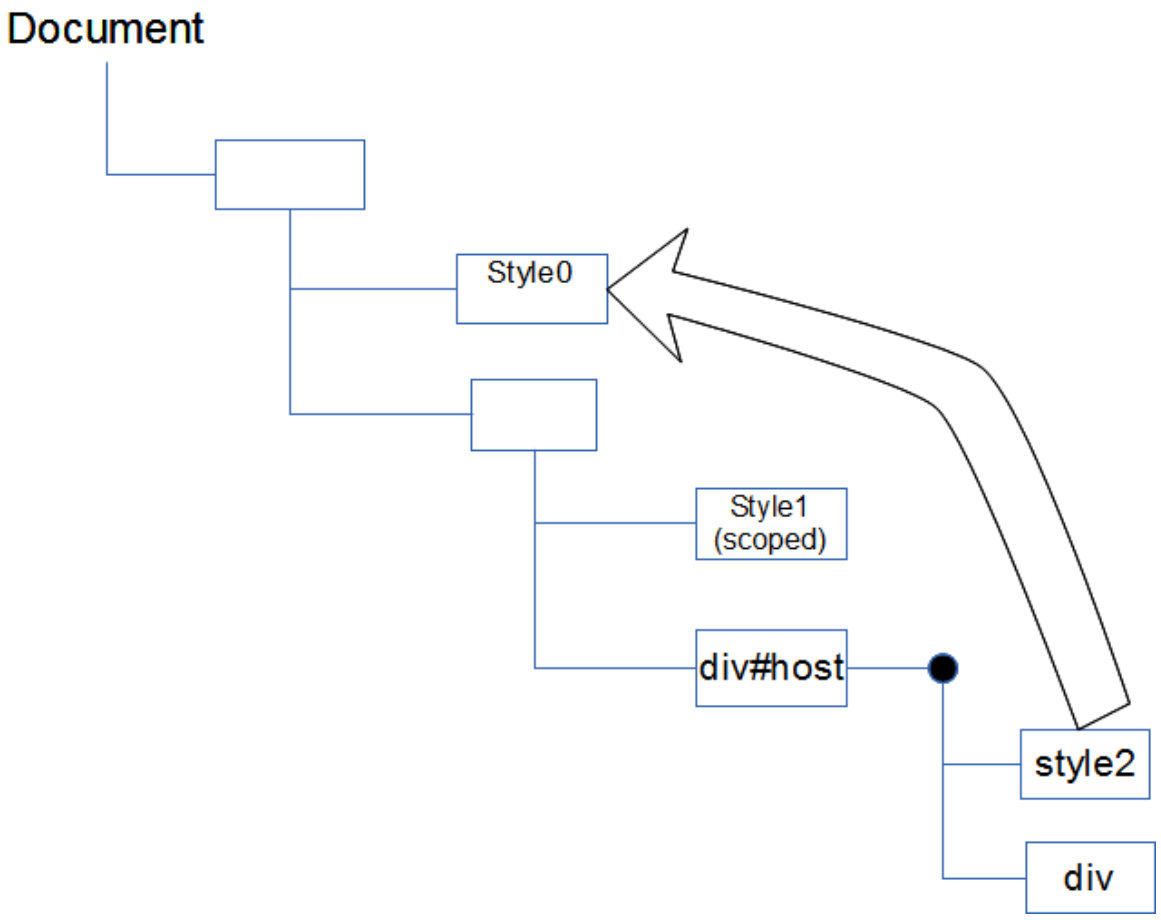
Idea 1: use a containing treescope

Idea 1: stylesheets in shadow trees are treated as if their scoping node is a containing treescope, style scoped always wins.

For example,



Since style0's scoping node is a root node of a treescope containing div#host, style0 and style2 are treated as if they have the same scoping node.

style0	div { border-color: blue; }
style1	div { border-color: lime; }
style2	div#host { border-color: red; }

	TreePosition	TreeInnerPosition

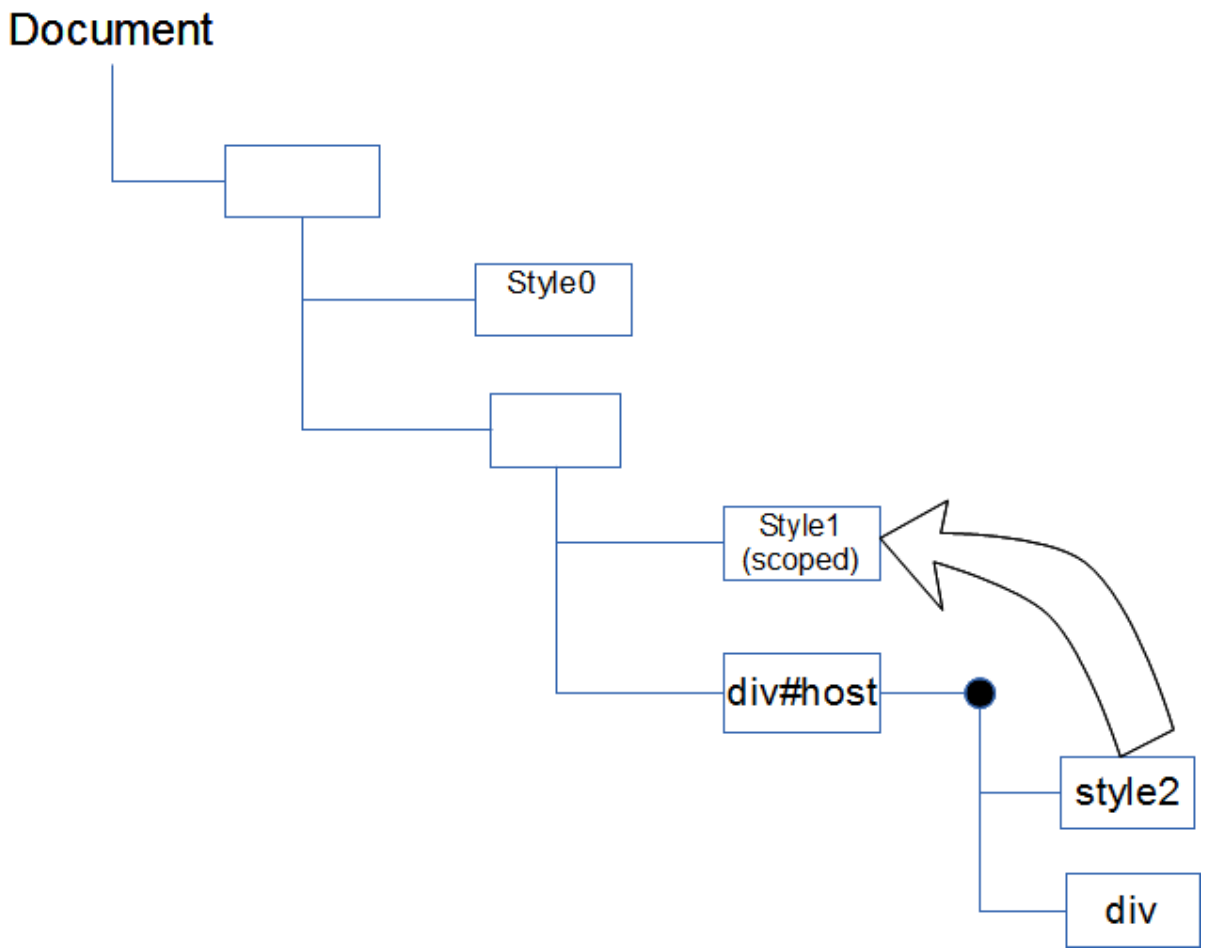
div { border-color: blue; }	0	1
div { border-color: lime; }	1	ignore
div#host { border-color: red; }	0	0

getComputedStyle('div#host').borderColor should be lime.

Idea 2: use a scoping node of style scoped if found

Idea 2: stylesheets in shadow trees are treated as if their scoping node is the same as the style scoped's scoping node. The scoping node is the first one when walking up from the shadow host.

For example,



When walking up from div#host, we find a scoping node of style1 before a scoping node of style0.
So we treat style2 as if its scoping node is the same as style1's.

style0	div { border-color: blue; }
style1	div { border-color: green; }
style2	div#host { border-color: red; }

	TreePosition	TreeInnerPosition
div { border-color: blue; }	0	ignore
div { border-color: lime; }	1	1
div#host { border-color: red; }	1	0

Since CSS cascade order is (1) TreePosition, (2) specificity and (3) TreeInnerPosition + order of appearance,

getComputedStyle('div#host').borderColor should be red.