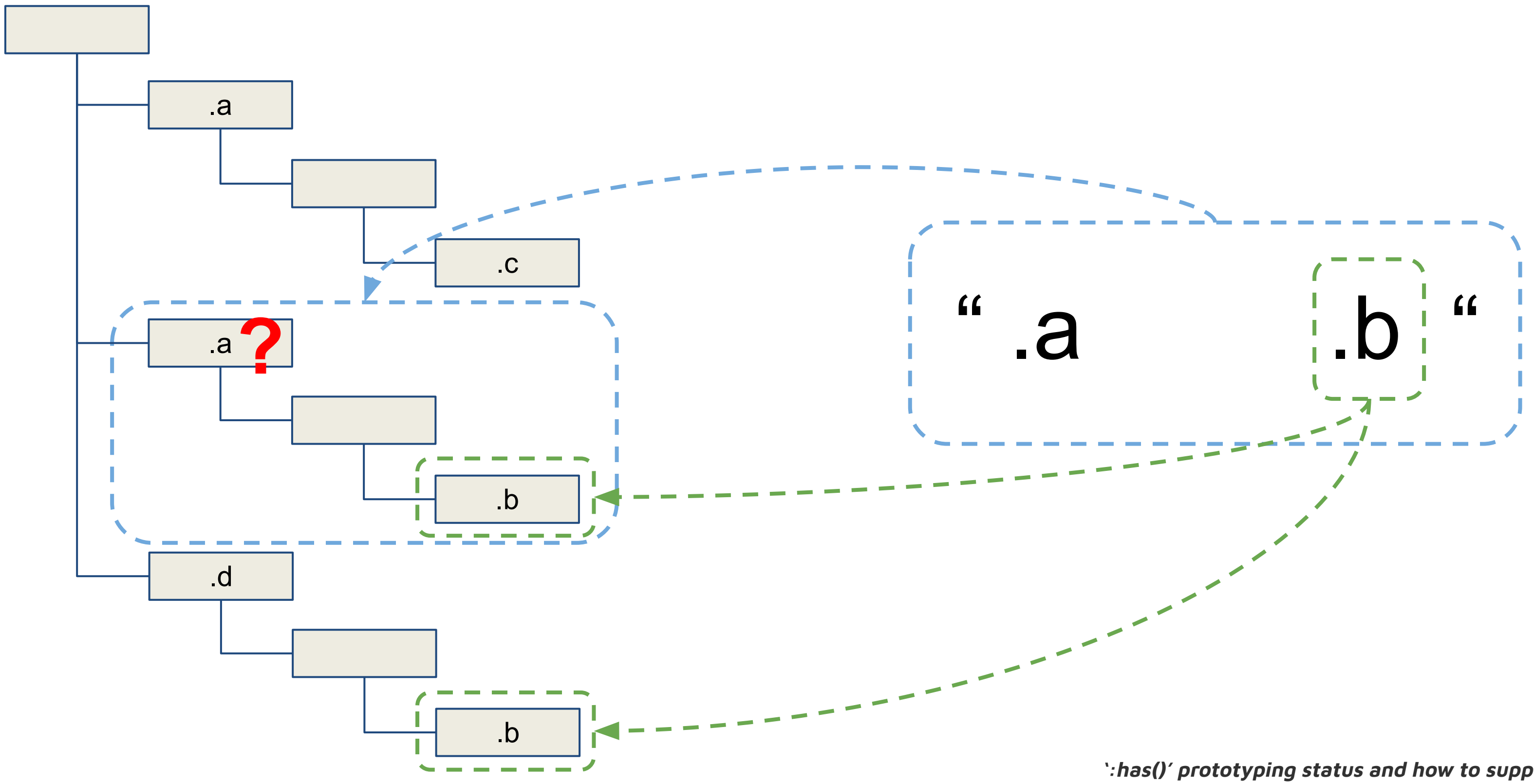
`:has()´ prototyping status
and how to support invalidation

Byungwoo Lee

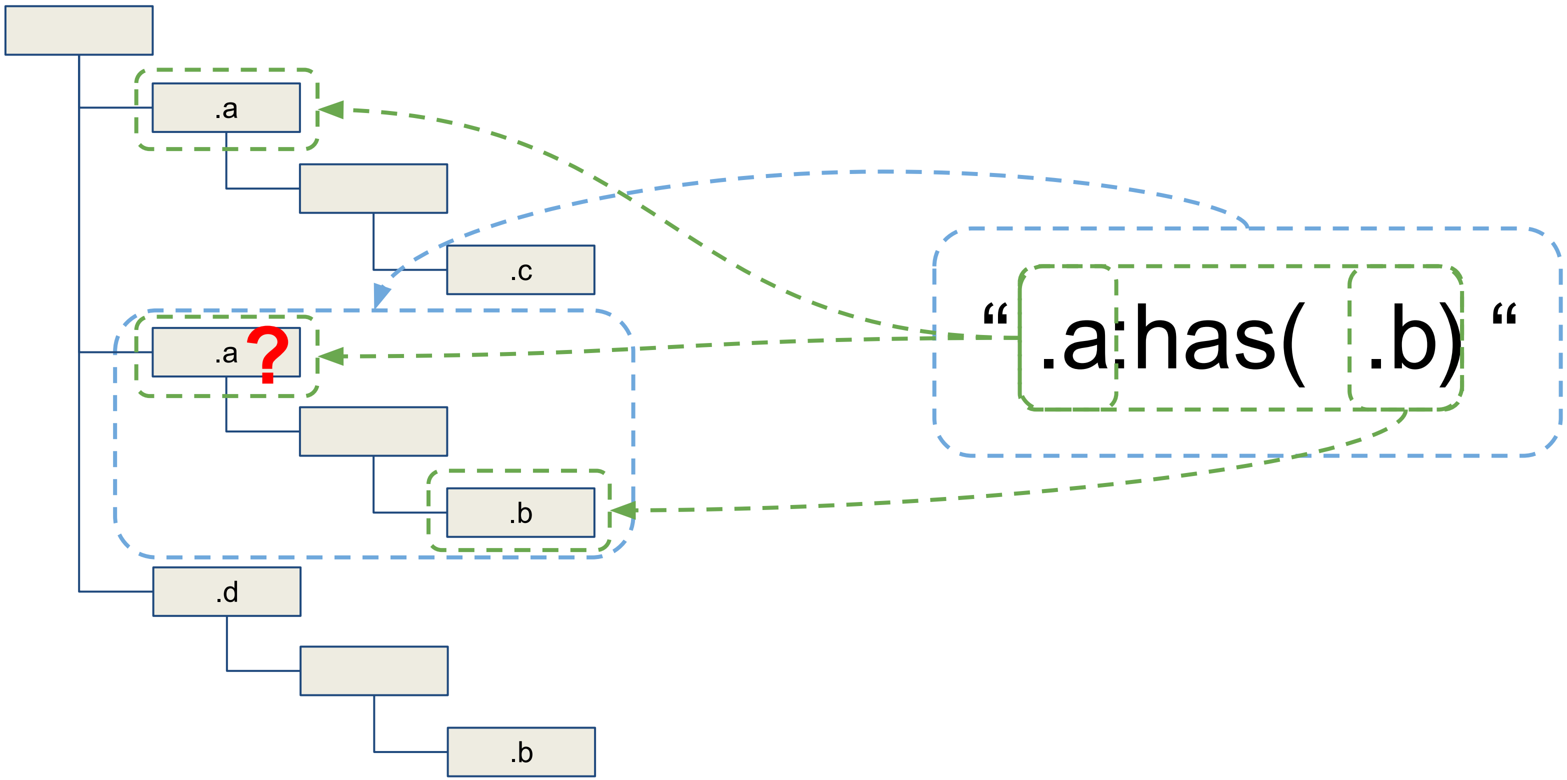
BlinkOn 15
2021.11.16

What can we do with `:has()’?

CSS Selector represents subject and subtree condition



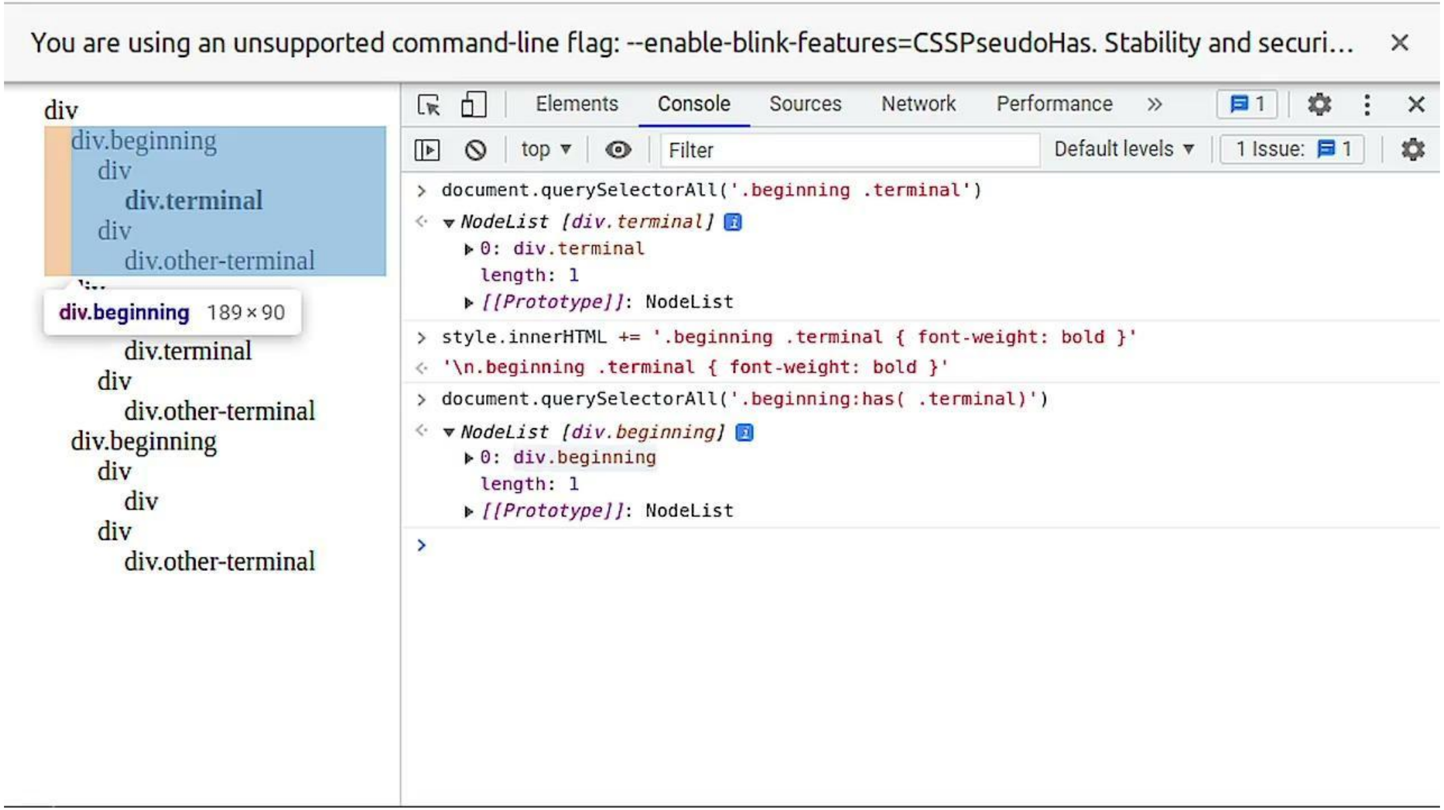
We can change the subject with ':has()'



':has()' prototyping status and how to support invalidation
Byungwoo Lee, BlinkOn 15 2021.11.16

What is the progress so far?

What we can do now



You are using an unsupported command-line flag: --enable-blink-features=CSSPseudoHas. Stability and securi… ✕

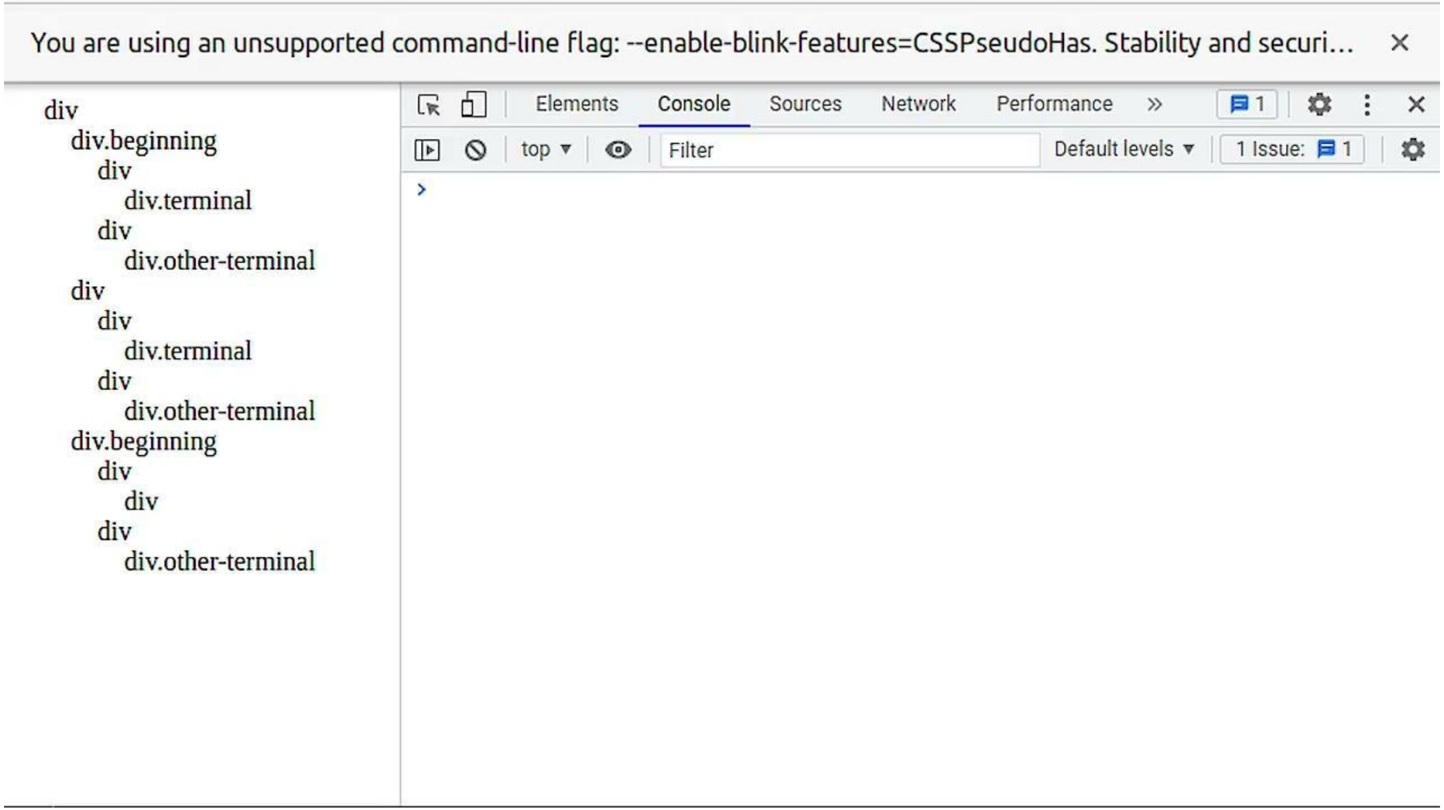
```
div
  div.beginning
    div
      div.terminal
    div
      div.other-terminal
```

div.beginning 189 × 90

```
      div.terminal
    div
      div.other-terminal
  div.beginning
    div
      div
    div
      div.other-terminal
```

```
> document.querySelectorAll('.beginning .terminal')
<· ▾NodeList [div.terminal]
    ▸0: div.terminal
     length: 1
    ▸[[Prototype]]: NodeList
> style.innerHTML += '.beginning .terminal { font-weight: bold }'
<· '\n.beginning .terminal { font-weight: bold }'
> document.querySelectorAll('.beginning:has( .terminal)')
<· ▾NodeList [div.beginning]
    ▸0: div.beginning
     length: 1
    ▸[[Prototype]]: NodeList
>
```

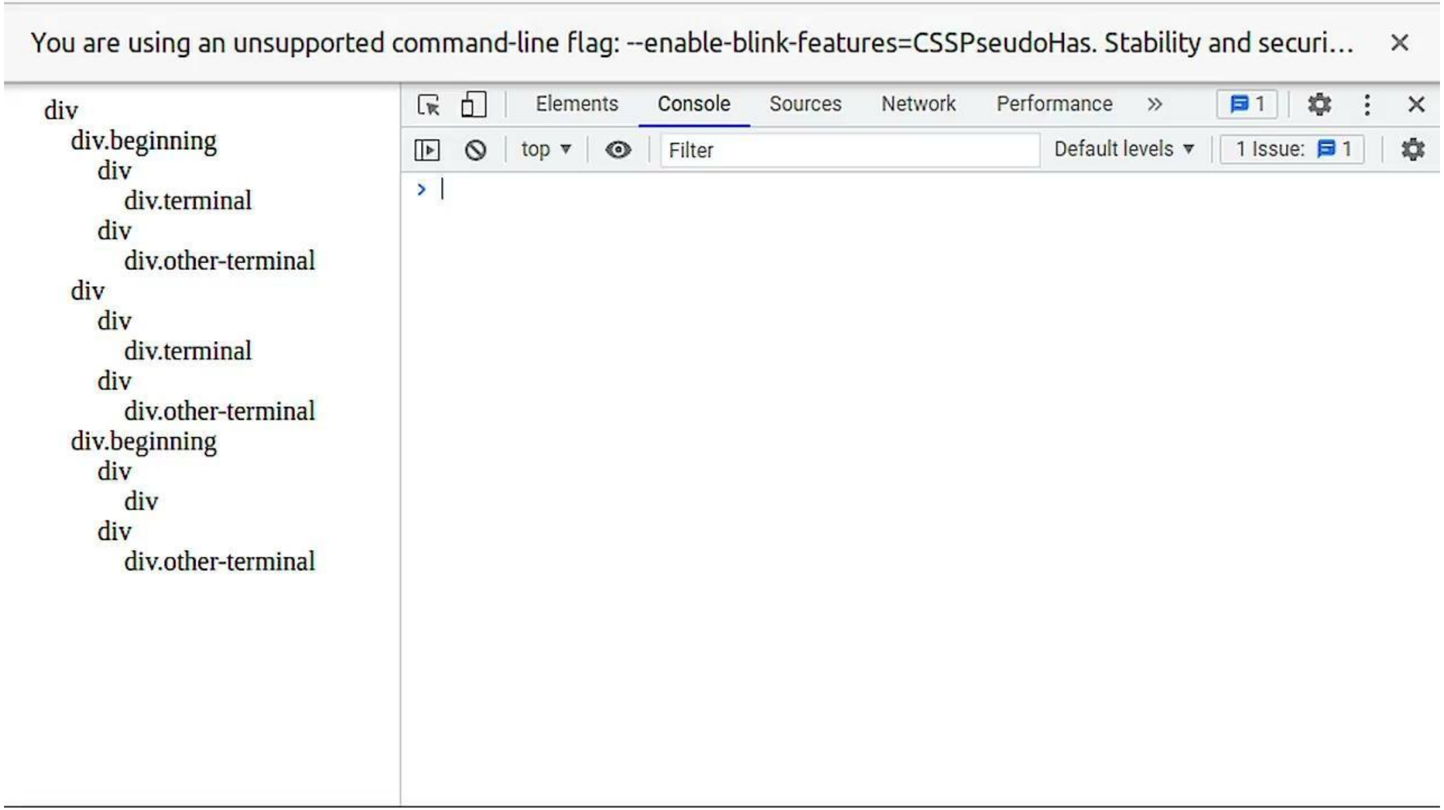
What we can’t do now



You are using an unsupported command-line flag: --enable-blink-features=CSSPseudoHas. Stability and securi… ×

```
div
  div.beginning
    div
      div.terminal
    div
      div.other-terminal
  div
    div
      div.terminal
    div
      div.other-terminal
  div.beginning
    div
      div
    div
      div.other-terminal
```

What we can do next



You are using an unsupported command-line flag: --enable-blink-features=CSSPseudoHas. Stability and securi… ×

```
div
  div.beginning
    div
      div.terminal
    div
      div.other-terminal
  div
    div
      div.terminal
    div
      div.other-terminal
  div.beginning
    div
      div
    div
      div.other-terminal
```

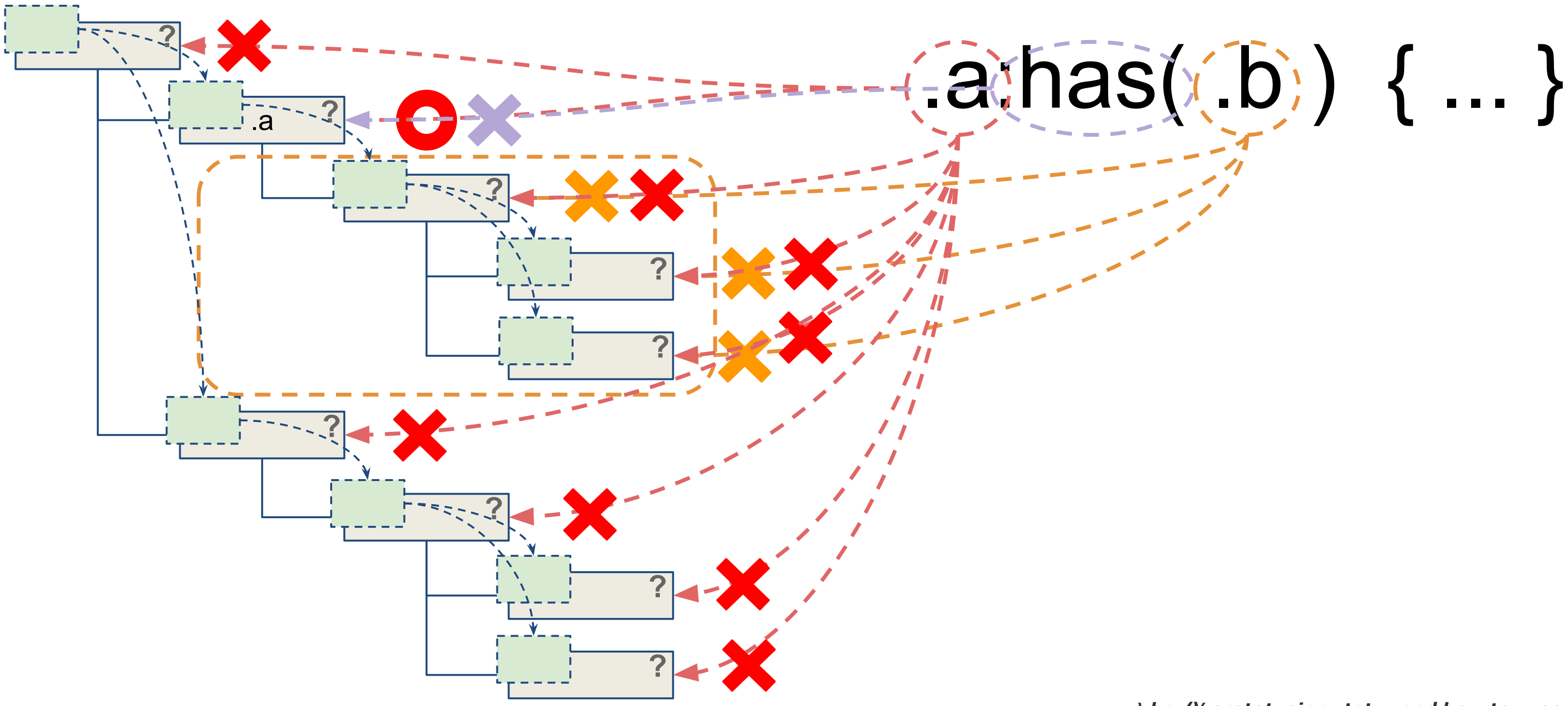
Next step?
Style invalidation with `:has()’

Let’s focus on a simple case

```
.a:has(.b) { … }
```

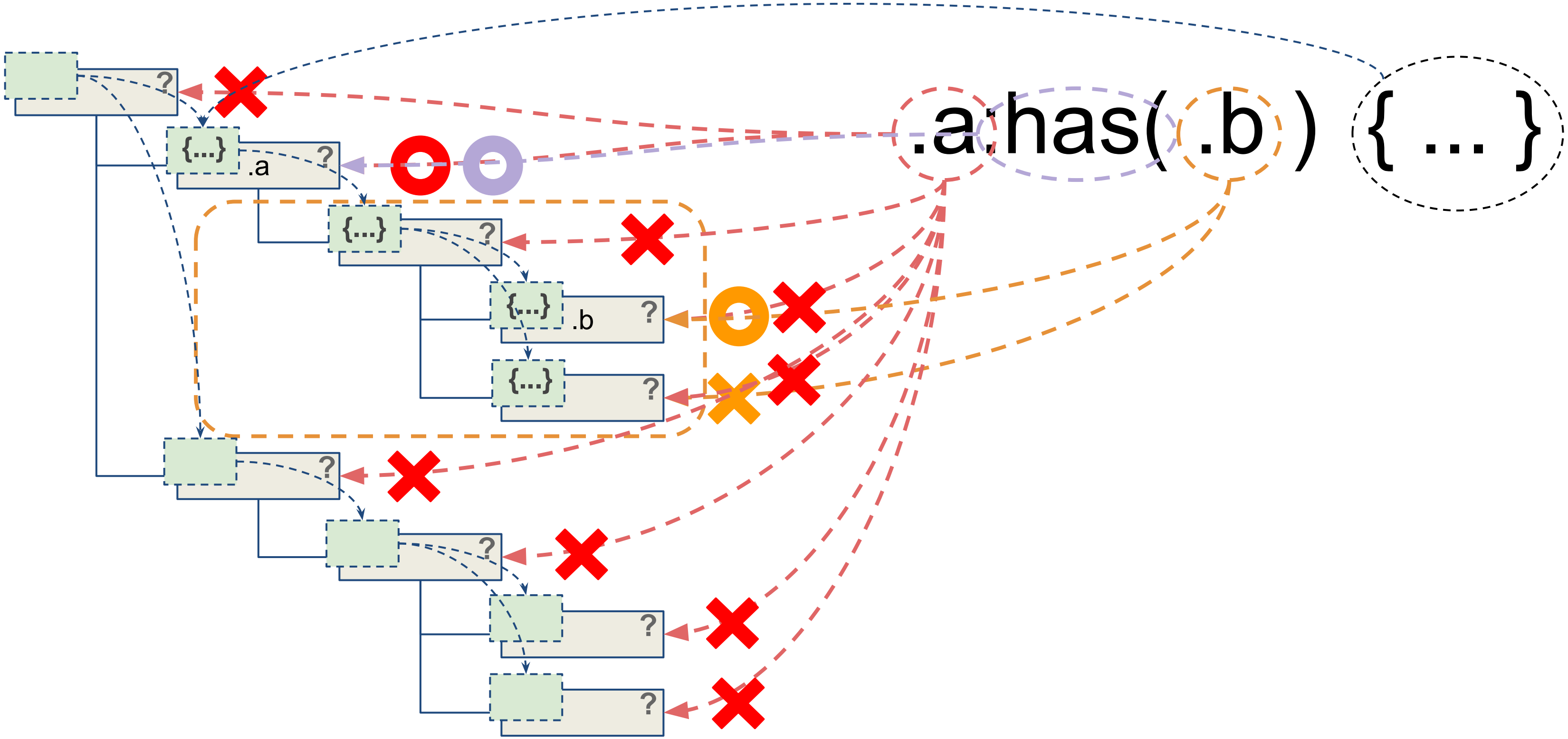
Recalculation and Invalidation with ‘:has()’

Style recalculation with `.a:has(.b)'



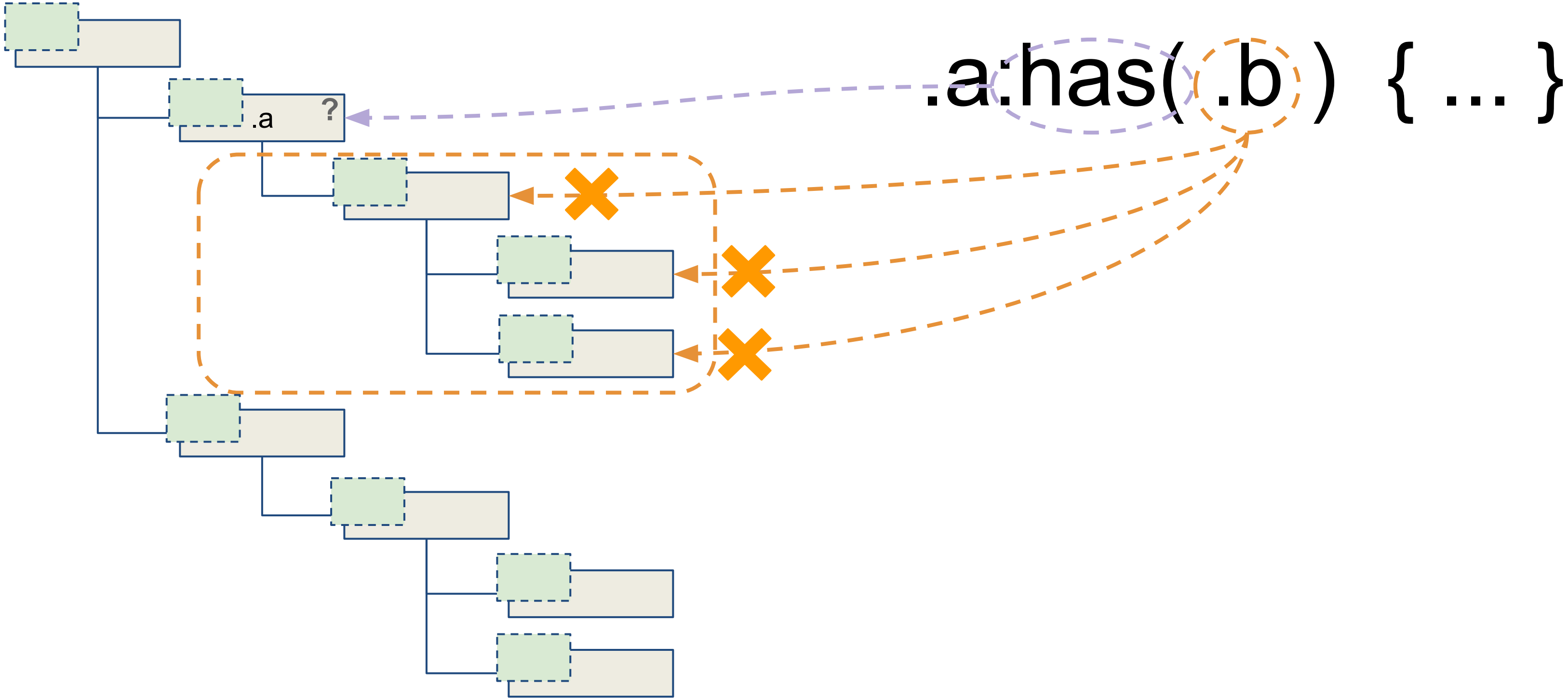
`:has()' prototyping status and how to support invalidation
Byungwoo Lee, BlinkOn 15 2021.11.16

Style recalculation with `.a:has(.b)'

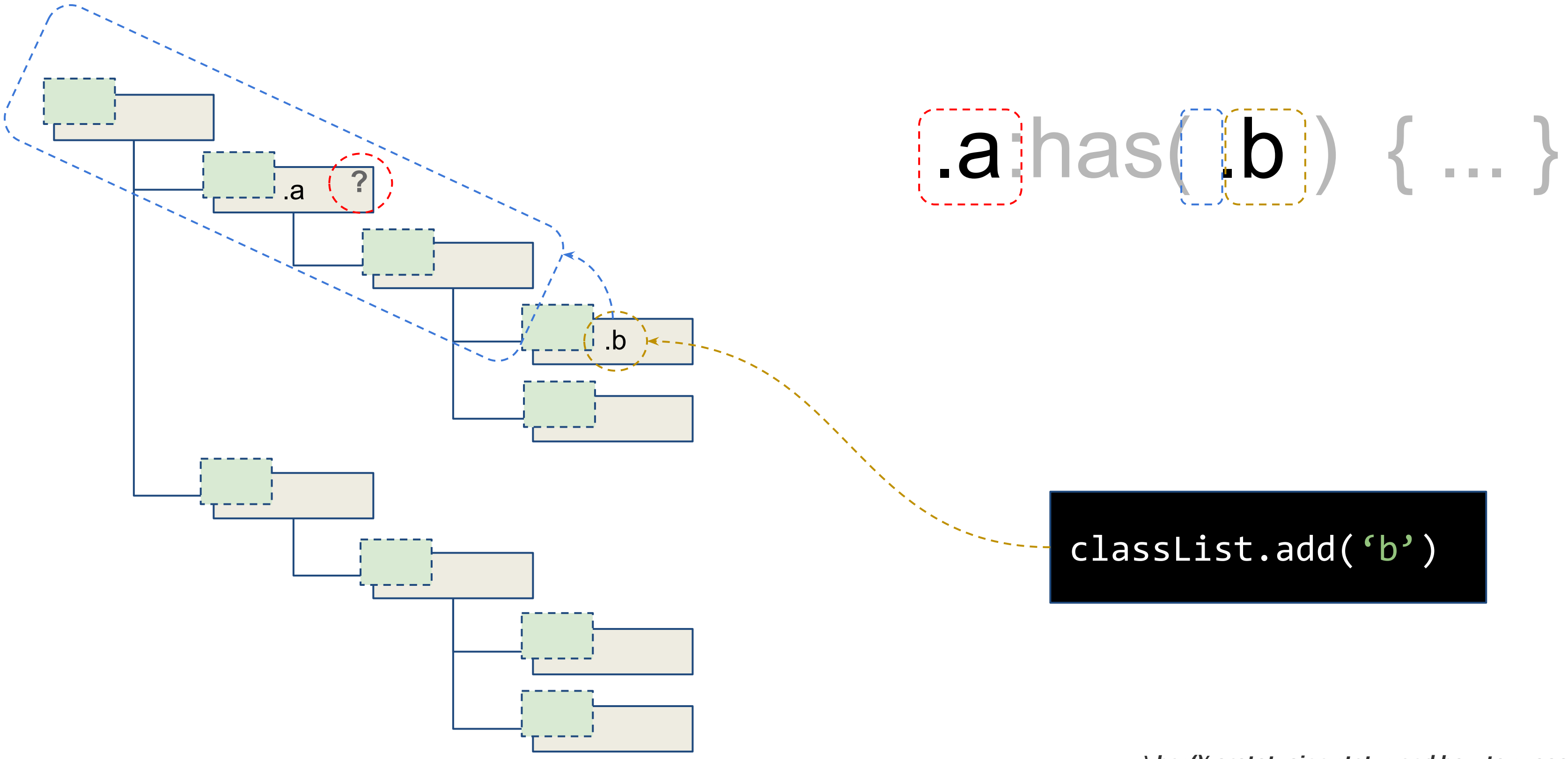


`:has()' prototyping status and how to support invalidation
Byungwoo Lee, BlinkOn 15 2021.11.16

O(n) `:has()′ matching overhead in style recalculation



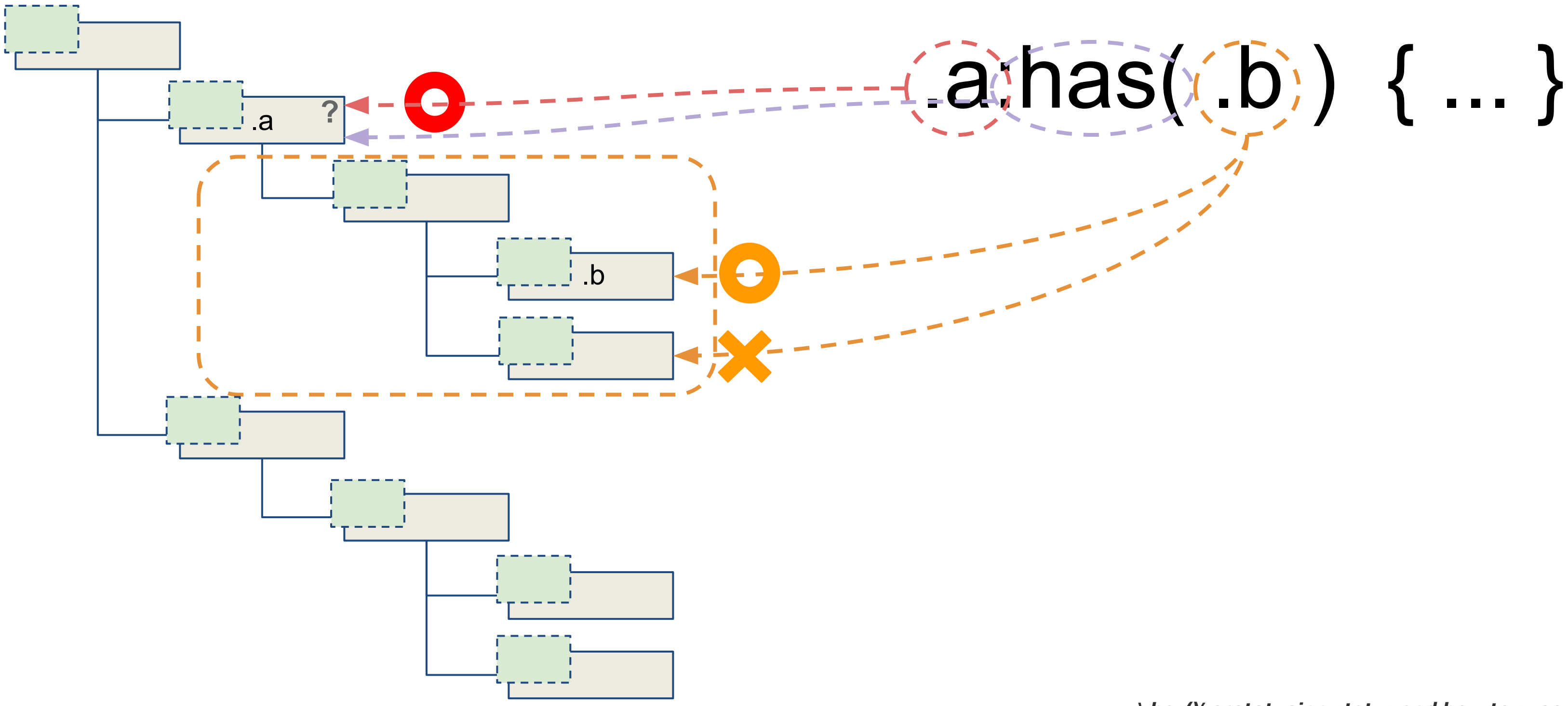
Style invalidation with `.a:has(.b)'



.a:has(.b) { … }

```
classList.add('b')
```

Recalculation will be triggered by the invalidation



':has()' prototyping status and how to support invalidation
Byungwoo Lee, BlinkOn 15 2021.11.16

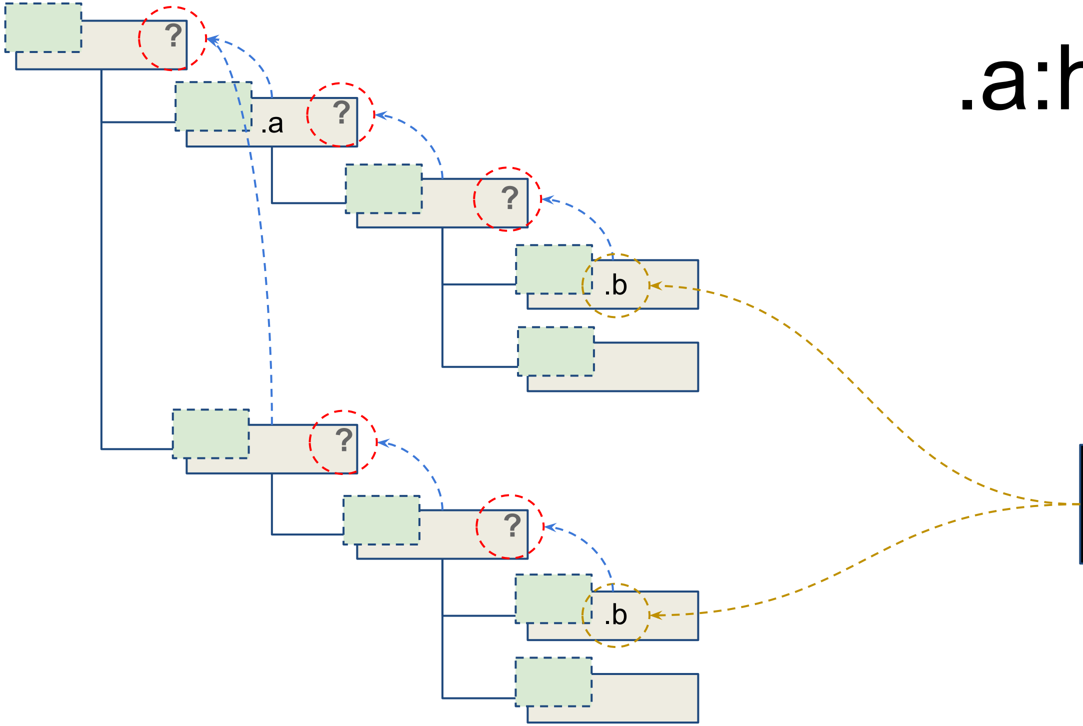
Style invalidation with `:has()′

For a mutation on an element, walk up ancestors and invalidate the style.

Every mutation?
Every element?
Every ancestors?

Optimize style invalidation with `:has()’

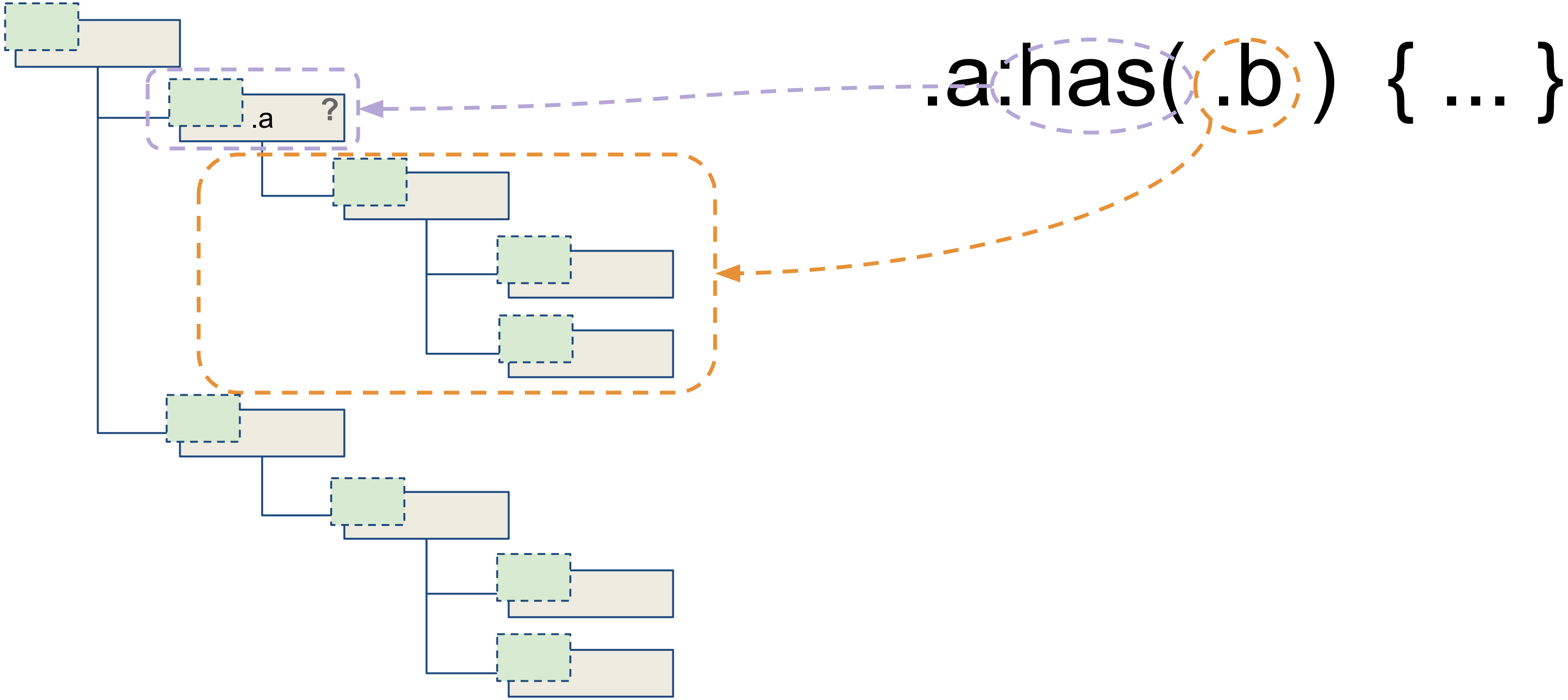
Invalidate every ancestor of every element



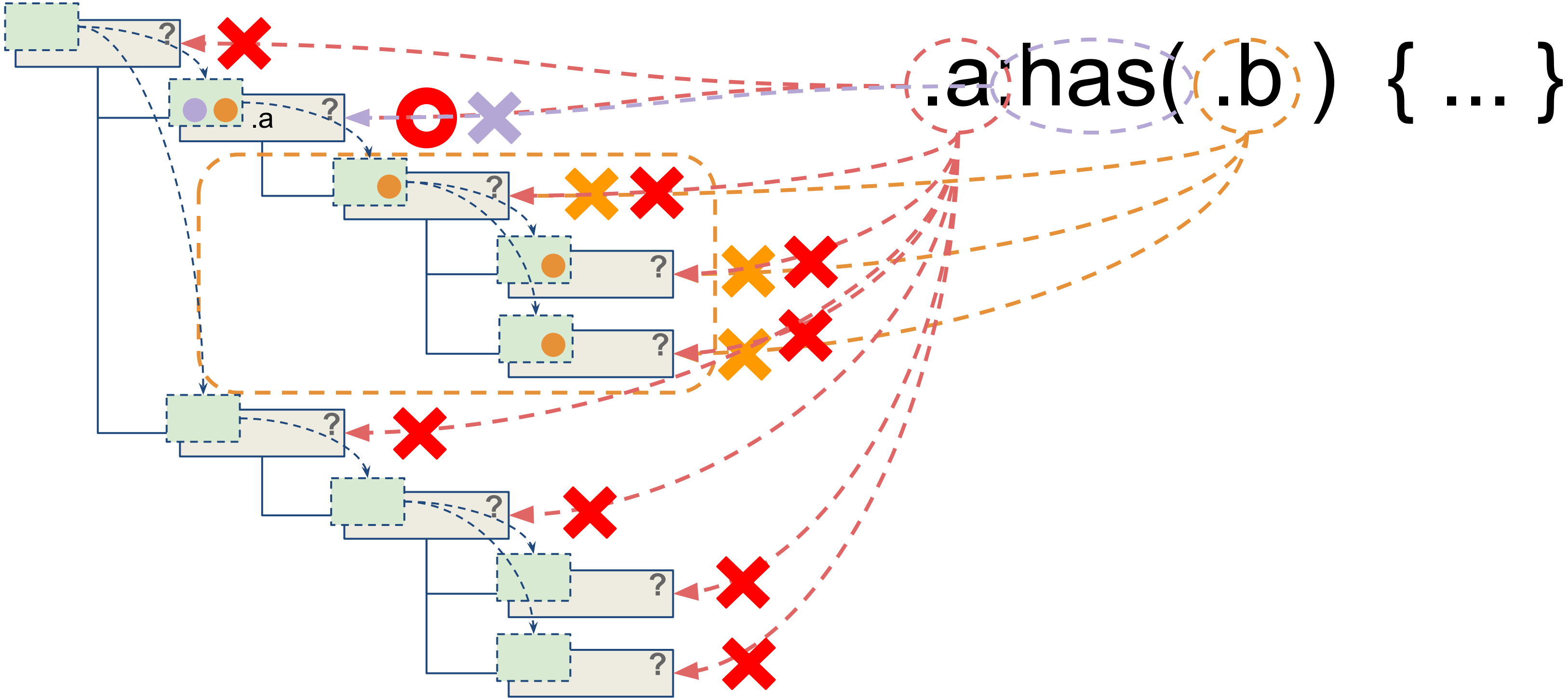
.a:has(.b) { … }

```
classList.add('b')
```

Affected by `:has()' / Affecting `:has()'



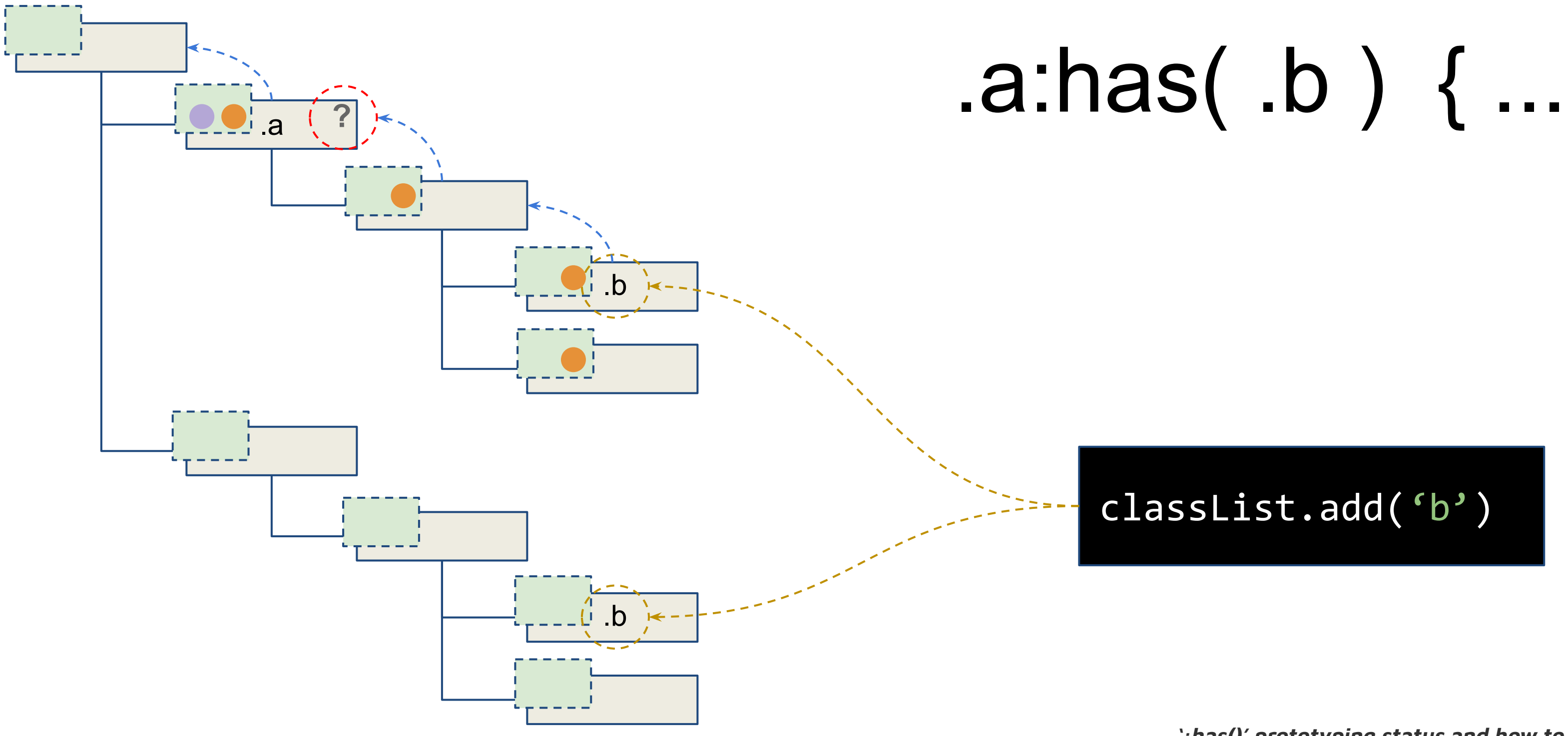
Marking ‘affected by’/’affecting’ flag during recalculation



‘:has()’ prototyping status and how to support invalidation
Byungwoo Lee, BlinkOn 15 2021.11.16

Prevent walk up and invalidation on irrelevant element

.a:has(.b) { … }



```
classList.add(‘b’)
```

Style invalidation with `:has()’

For a mutation on an element, walk up ancestors and invalidate the style.

~~Every~~ element?

marked as affecting `:has()’

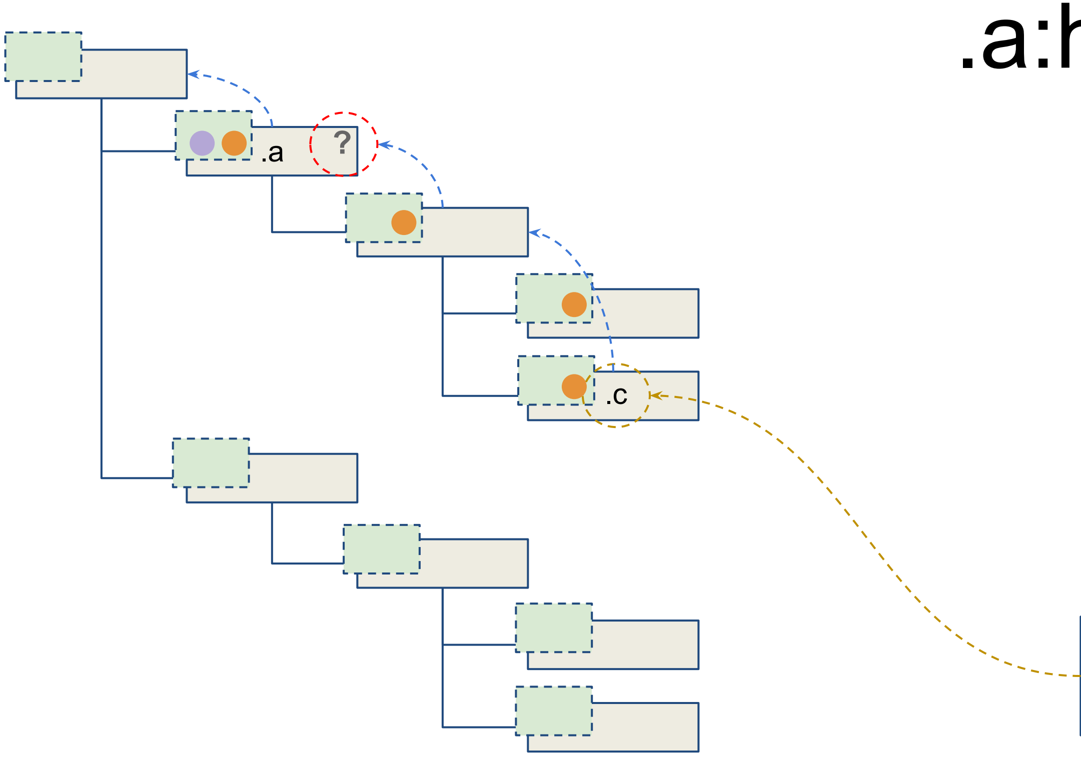
Every mutation?

~~Every~~ ancestors?

marked as affected by `:has()’

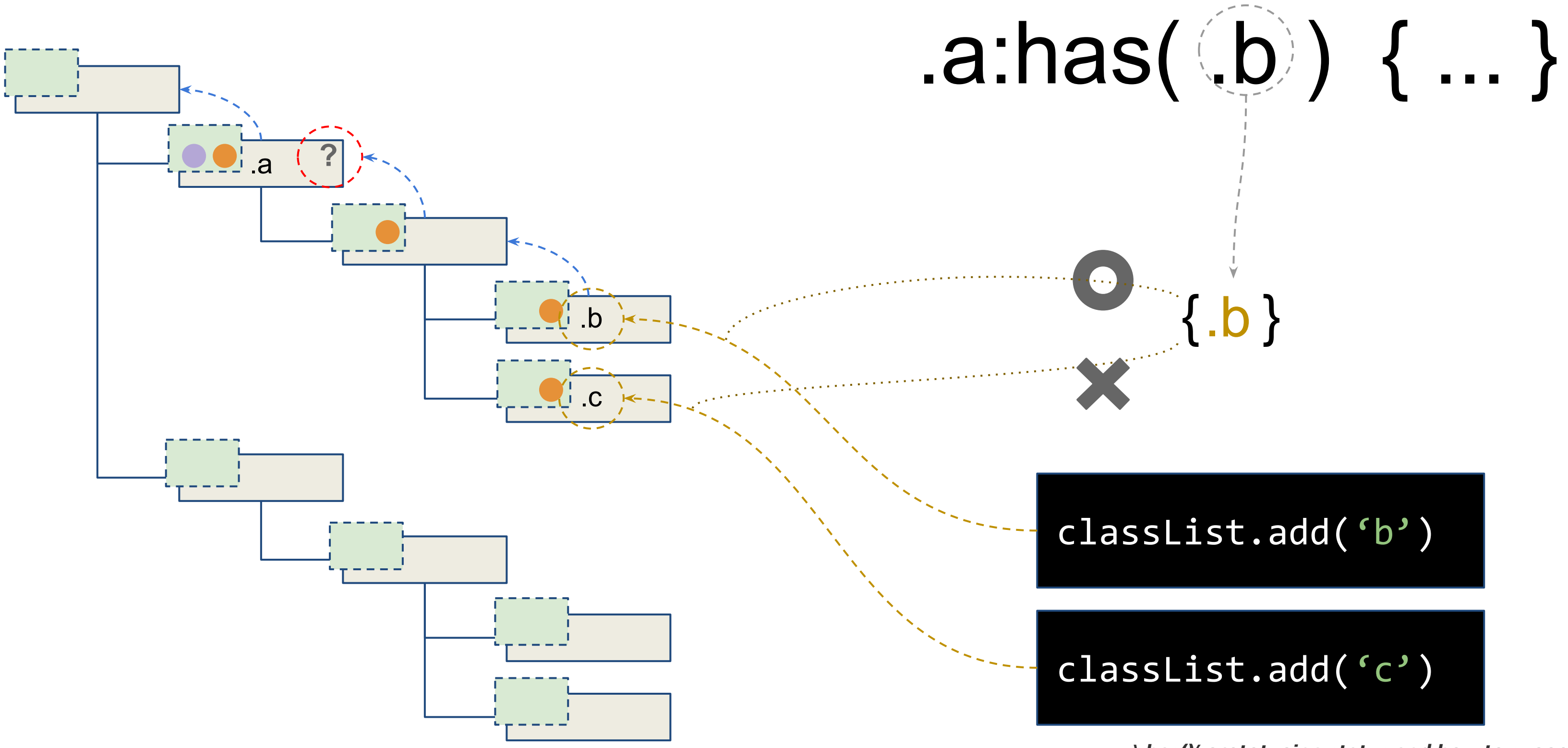
Walk up ancestors for irrelevant mutations

.a:has(.b) { … }



```
classList.add(‘c’)
```

Prevent walk up ancestors for irrelevant mutations



```
classList.add('b')
```

```
classList.add('c')
```

Style invalidation with `:has()’

For a mutation on an element, walk up ancestors and invalidate the style.

~~Every~~ element?

marked as affecting `:has()’

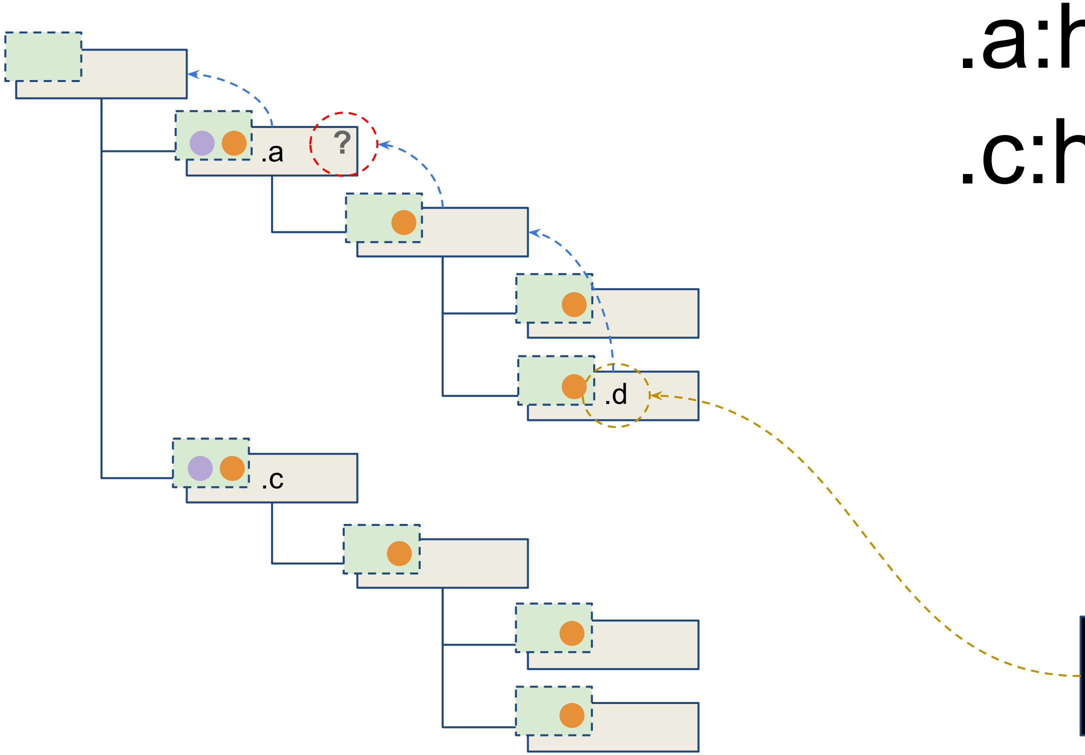
~~Every~~ mutation?

matches one of features from `:has()’ arguments

~~Every~~ ancestors?

marked as affected by `:has()’

Invalidation of ancestors irrelevant to a mutation



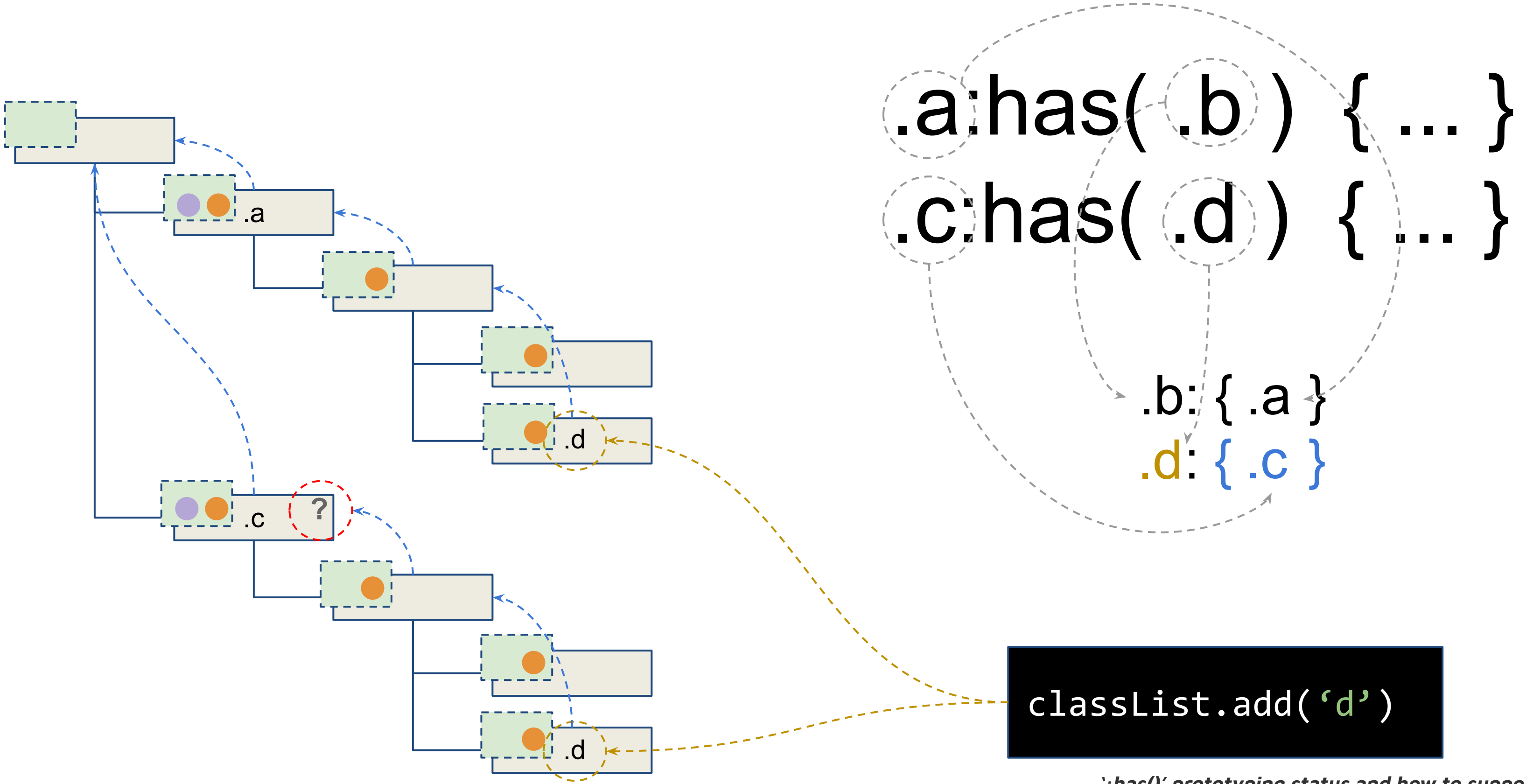
.a:has(.b) { … }

.c:has(.d) { … }

{ .b , .d }

```
classList.add(‘d’)
```

Prevent invalidation of ancestors irrelevant to a mutation



.a:has(.b) { ... }

.c:has(.d) { ... }

.b: { .a }

.d: { .c }

```
classList.add(‘d’)
```

Style invalidation with `:has()'

For a mutation on an element, walk up ancestors and invalidate the style.

~~Every~~ element?

    marked as affecting `:has()'

~~Every~~ mutation?

    matches one of features from `:has()' arguments

~~Every~~ ancestors?

    marked as affected by `:has()'

    matches one of features from the compound containing the `:has()'

For Other variations of `:has()’ invalidation

We will go through the other variations step by step

complex selector as argument	`.a:has(.b ~ .c)`
none-terminal ‘:has()’	`.a:has(.b) .c`
‘:has()’ in logical combination	`.a:is(:has(.b), :has(.c))`
pseudos in argument	
user action pseudo classes in argument	`.a:has(.b:hover)`
input pseudo classes in argument	`.a:has(.b:checked)`
logical combinations in argument	`.a:has(.b:not(.c))`
…	
argument starts with ‘~’ or ‘+’	`.a:has(~ .b)`

Thank you!

blee@igalia.com

igalia