# @sheet - CSSWG f2f

Kurt Catti-Schmidt
Andy Luhrs

Microsoft Edge

## Background

Multiple stylesheets per file · Issue #5629 · w3c/csswg-drafts - Original CSSWG Issue (2020)

[CSSWG] Minutes Telecon 2023-04-05 [css-highlight-api] [css-pseudo] [css-cascade] [css-values] from Dael Jackson on 2023-04-06 (www-style@w3.org from April 2023) - CSSWG discussion / resolution (2023)

MSEdgeExplainers/AtSheet/explainer.md at main · MicrosoftEdge/MSEdgeExplainers - Explainer (2025)

## @sheet Goals

- Combine multiple stylesheets into one file
  - Fewer network requests
  - Increased compression ratios
  - More methods for organizing styles
  - Solution for sharing inline styles with Declarative Shadow DOM

## @sheet Goals

- Combine multiple stylesheets into one file
    - **Fewer network requests -** Clients only download one .css file instead of many - bundling!
    - Increased compression ratios
    - More methods for organizing styles
    - Solution for sharing inline styles with Declarative Shadow DOM

## @sheet Goals

- Combine multiple stylesheets into one file
    - Fewer network requests
    - **Increased compression ratios** - With dictionary-based compression algorithms, combining many CSS files into one file will allow all of the CSS-specific tokens to be dictionary hits, improving compression ratios
        - In a real-world site (cnn.com) I made a basic test case that improved compression by **0.4%** simply by combining stylesheets via `@sheet`.
    - More methods for organizing styles
    - Solution for sharing inline styles with Declarative Shadow DOM

## @sheet Goals

- Combine multiple stylesheets into one file
  - Fewer network requests
  - Increased compression ratios
  - **More methods for organizing styles** - `@sheet` gives developers another tool for structuring large CSS files
  - Solution for sharing inline styles with Declarative Shadow DOM

## @sheet Goals

- Combine multiple stylesheets into one file
  - Fewer network requests
  - Increased compression ratios
  - More methods for organizing styles
  - **Solution for sharing inline styles with Declarative Shadow DOM**

## Core concept already resolved

**RESOLVED: Accept @sheet with URL fragment referencing rule. Exact details to be in the Cascade spec**

## Initial Proposal - Multiple Stylesheets per File (CSSWG #5629)

**sheet.css**

```css
@sheet sheet1 {
  :host {
    display: block;
    background: red;
  }
}

@sheet sheet2 {
  p {
    color: blue;
  }
}
div { color: orange; }
```

**JavaScript**

```js
import {sheet1, sheet2} from './sheet.css' assert {type: 'css'};
```

**or**

```js
import styles, {sheet1, sheet2} from './sheet.css' assert {type: 'css'};
styles.sheet1 === sheet1;
styles.sheet2 === sheet2;
```

**RESOLVED: Accept @sheet with URL fragment referencing rule. Exact details to be in the Cascade spec**

## CSSWG 2023 Discussion (CSSWG #5629)

**sheet.css**

```
@sheet sheet1 {
  :host {
    display: block;
    background: red;
  }
}

@sheet sheet2 {
  p {
    color: blue;
  }
}
div { color: orange; }
```

**HTML**

```
<link rel=”stylesheet” href=”sheet.css#sheet1” />
```

**CSS**

```
@import(“sheet.css#sheet1”);
```

**RESOLVED: Accept @sheet with URL fragment referencing rule. Exact details to be in the Cascade spec**

## CSSWG 2023 Discussion (CSSWG #5629)

**sheet.css**

```
@sheet sheet1 {
  :host {
    display: block;
    background: red;
  }
}

@sheet sheet2 {
  p {
    color: blue;
  }
}

div { color: orange; }
```

**HTML**

```
<link rel=”stylesheet” href=”sheet.css#sheet1” />
```

**CSS**

```
@import(“sheet.css#sheet1”);
```

**RESOLVED: Accept @sheet with URL fragment referencing rule. Exact details to be in the Cascade spec**

## What About Inline Styles?

**HTML**

```
<style>
@sheet sheet1 {
  :host {
    display: block;
    background: red;
  }
}

@sheet sheet2 {
  p {
    color: blue;
  }
}

div { color: orange; }

</style>


<link rel="stylesheet" href="#sheet1" />
```

**OR**

```
<style>
@import("#sheet1");
</style>
```

**NOT DISCUSSED AT CSSWG! Should this work?**

## What does this have to do with Declarative Shadow DOM?

- Shadow DOM has an `adoptedStyleSheets` DOM attribute
- Declarative Shadow DOM (DSD) can use the `adoptedStyleSheets` DOM API
    - Using a DOM API to set initial styles sort of defeats the point of DSD though
- The *only* current mechanisms for sharing styles with Declarative Shadow DOM are:
    - Duplicated `<link rel>` stylesheets
        - `<link rel>` must be an external file or a dataURI!
    - Duplicated inline styles
    - Script-based `adoptedStyleSheets`

## Declarative Shadow DOM continued

- If we support same-document fragment identifiers for `@sheet`, this could be an elegant solution for sharing inline styles with Declarative Shadow DOM!

```
<style>
@sheet sheet1 {
  :host {
    display: block;
    background: red;
  }
}
</style>
<template shadowrootmode="open">
  <link rel="stylesheet" href="#sheet1" />
  <span>I'm in the shadow DOM</span>
</template>
```

## Still lots of open issues with @sheet

- MSEdgeExplainers/AtSheet/explainer.md at main · MicrosoftEdge/MSEdgeExplainers - direct link to open issues

## Declarative CSS Modules

- Modules (CSS, HTML, JS, WASM) have lots of use cases and developer interest
- Declarative CSS can prevent FOUC (Flash of unstyled content)
- Another potential solution for sharing declarative CSS with Declarative Shadow DOM
- MSEdgeExplainers/ShadowDOM/explainer.md at main · MicrosoftEdge/MSEdgeExplainers

## Declarative CSS Module Goals

- Modules (CSS, HTML, JS, WASM) have lots of use cases and developer interest
- Declarative CSS can prevent FOUC (Flash of unstyled content)
- **Another potential solution for sharing declarative CSS with Declarative Shadow DOM**
- MSEdgeExplainers/ShadowDOM/explainer.md at main · MicrosoftEdge/MSEdgeExplainers

## Declarative CSS Modules Proposal

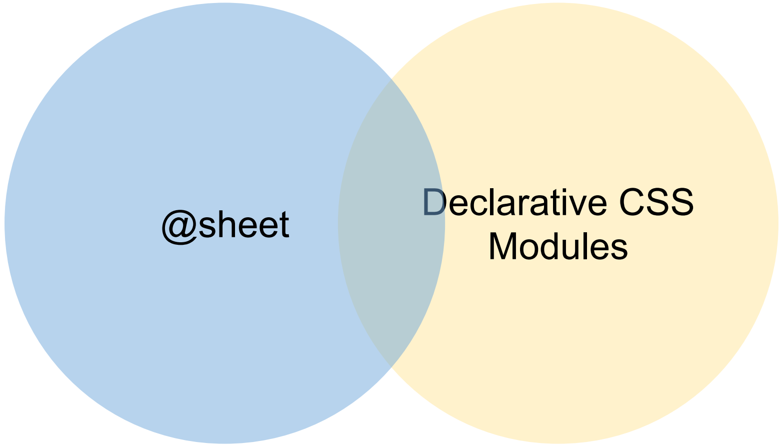
```
<script type="css-module" specifier="/foo.css">
  #content {
    color: red;
  }
</script>

<my-element>
  <template shadowrootmode="open" adoptedstylesheets="/foo.css">
    <!-- ... -->
  </template>
</my-element>
```

## Declarative CSS Modules (WHATWG #10673 & TAG #1000)

**“To allow these styles to be defined inline, `@sheet` seems like a better solution.”** - TAG feedback

# Shared Goals

Improve Bundling



Improve Shadow DOM

Style Re-use (including inline style)
Minimize Network Requests
Minimize JavaScript

# Expanded @sheet Proposal (CSSWG #11509)

## Normal Usage

```
<style>
  @sheet sheet1 {
    ...
  }
</style>
<link rel="stylesheet" href="#sheet1">
```

## Shadow DOM Usage

```
<style>
  @sheet sheet1 {
    ...
  }
</style>
<template shadowrootmode="open">
  <link rel="stylesheet" href="#foo" />
  <span>I'm in the shadow DOM</span>
</template>
```

# Open Issues (link)

1. Do we want to be able to access sheets declared in shadow DOM from light DOM?
2. Are fragment-only identifiers (without a URL) the right approach?

---

3. Are rules from @sheets applied automatically or do they need to be imported?
4. Should @import be possible within @sheet? Should it be allowed if it's the first/only statement? or should it be blocked?
5. What happens with multiple @sheet definitions with the same identifier? First-definition wins, or do they get merged like @layer?
6. If a stylesheet contains named @sheet references and rules outside of the @sheet references, what happens when a fragment identifier is not specified?

## Do we want to be able to access sheets declared in shadow DOM from light DOM? (#938)

- Streaming SSR?

```
<template shadowrootmode="open">
  <style>
    @sheet foo {
      div {
        color: red;
      }
    }
  </style>
  <link rel="stylesheet" href="#foo" />
  <span>I'm in the shadow DOM</span>
</template>

<link rel="stylesheet" href="#foo" />
<span>I'm in the light DOM</span>
```

## Are fragment-only identifiers (without a URL) the right approach? (#935)

- The CSSWG already resolved to use fragment identifiers with external .css files `<link rel=”stylesheet” href=”foo.css#bar”>`) - is there a good reason why this shouldn’t work for local references? (`<link rel=”stylesheet” href=”#bar”>`)
- Would need to modify fragment syntax and/or how the `id` attribute works
  - URL fragments are currently used for scrolling the viewport
    - This is a very different scenario than stylesheet sharing
- New attribute `adoptedStyleSheets`?
  - This sidesteps the open issues with standalone fragments

## Other Open Issues

(if time allows)

## Are rules from @sheets applied automatically or do they need to be imported? (#934)

## Should @import be possible within @sheet? Should it be allowed if it's the first/only statement? or should it be blocked? (#936)

https://lists.w3.org/Archives/Public/www-style/2023Apr/0004.html

## What happens with multiple @sheet definitions with the same identifier? First-definition wins, or do they get merged like @layer? (#937)

- https://github.com/w3c/csswg-drafts/issues/5629#issuecomment-1498299448
- it's possible to have a "Flash of other-styled content" if it's last-defintion-wins, as the first definition may apply, then a later definition from an external CSS file may override it.

## If a stylesheet contains named @sheet references and rules outside of the @sheet references, what happens when a fragment identifier is not specified?

**sheet.css**

```
@sheet foo {
  div{
    color: red;
  }
}
div {
  color: blue;
}
```

```
<style>
  /* Does the @sheet "foo" get dropped? */
  @import "sheet.css"
</style>
<!-- Does the @sheet "foo" get dropped? -->
<link rel="stylesheet" href="sheet.css">
```

## Compression Benefits of @sheet vs separate .css files

On cnn.com, I was able to improve the compression of their .css files by **0.4%!**

- Instead of 3 separate .css files being compressed individually, combining them via `@sheet` and then compressing increases the compression dictionary’s hit rate
- The total bytes went from 32279 to 32144, a 0.4% improvement
- The real-world percentage will vary based on the stylesheet contents