DISKUSIKAN MATERI
LAPORKAN MATERI

Daftar Modul

Masukkan kata kunci

Persetujuan Hak Cipta
Modul 1: Introduction Course (Gratis)
Prasyarat Kelas (Gratis)
Apa yang Akan Kita Pelajari? (Gratis)
Tools Requirement
Modul 2: ECMAScript 6 (Gratis)

Shadow DOM in Web Components

Potongan kode untuk materi ini: https://repl.it/@dicodingacademy/163-03-shadow-dom-in-web-component?lite=true

Untuk membantu menerapkan enkapsulasi pada custom element, Shadow DOM berperan sebagai salah satu API standar yang digunakan dalam membuat Web Component (Hal ini distandarisasi oleh W3C). Kita sudah belajar bagaimana menerapkan Shadow DOM pada elemen yang berada pada Document Tree, namun bagaimana caranya bila itu diterapkan pada custom element?

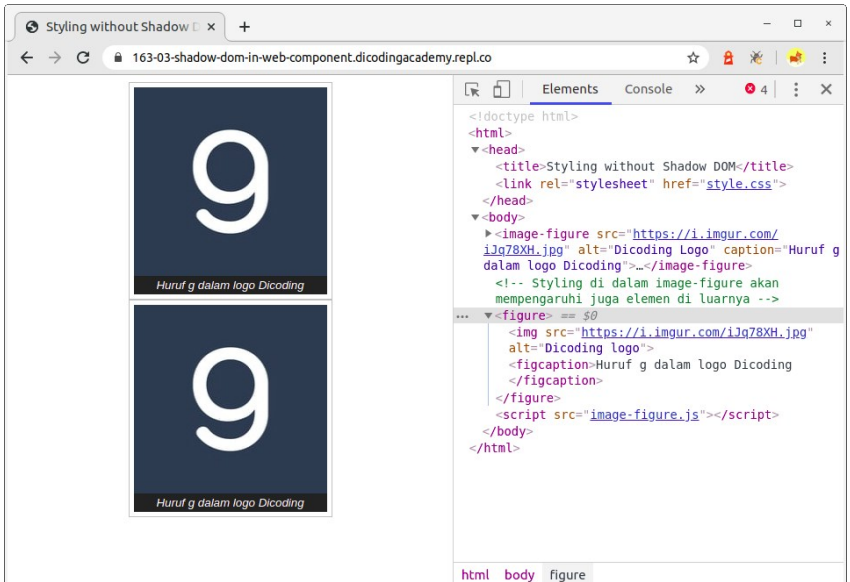
Jawabannya cukup mudah! Mari kita lihat kembali contoh custom element yang pernah kita buat pada materi Styling Custom Element without Shadow DOM

index.html image-figure.js

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width">
   <title>Styling without Shadow DOM</title>
   <link rel="stylesheet" href="style.css"/>
 </head>
<body>
   <image-figure
     src="https://i.imgur.com/iJq78XH.jpg"
     alt="Dicoding Logo"
     caption="Huruf g dalam logo Dicoding">
     </image-figure>


     <!-- Styling di dalam image-figure akan mempengaruhi juga elemen di luarnya -->
     <figure>
       <img src="https://i.imgur.com/iJq78XH.jpg"
       alt="Dicoding logo"/>
       <figcaption>Huruf g dalam logo Dicoding</figcaption>
     </figure>
   <script src="image-figure.js"></script>
 </body>
</html>
```

Ketika kode tersebut dijalankan pada browser, kita bisa melihat terdapat dua komponen <figure> yang ditampilkan, salah satunya adalah custom element.



Kita bisa melihat juga bahwa keduanya memiliki styling yang sama, padahal kita hanya menetapkan styling di dalam komponen `ImageFigure` saja. Yup, hal tersebut wajar terjadi karena pada custom element kita tidak menetapkan Shadow DOM sehingga styling pada custom element akan berdampak juga terhadap komponen di luarnya.

Dalam melampirkan Shadow DOM pada custom element sama seperti pada elemen biasanya, yaitu menggunakan `attachShadow`. Namun dalam custom element, kita lakukan pada constructor class-nya seperti ini:

```
class ImageFigure extends HTMLElement {

 constructor() {
   super();
   this._shadowRoot = this.attachShadow({mode: "open"});
 }

  .....
}
```

Agar nilai `shadowRoot` dapat diakses pada fungsi mana saja di class, maka kita perlu memasukkan nilai `shadowRoot` pada properti class menggunakan `this`. Kita bebas menentukan nama properti sesuai keinginan, namun untuk memudahkan kita gunakan nama `_shadowRoot`. Lalu mengapa penamaannya menggunakan tanda underscore (`_`) di depannya? Jawabannya, this pada konteks class ini merupakan HTMLElement dan ia sudah memiliki properti dengan nama `shadowRoot`. Untuk membedakan properti `_shadowRoot` asli dengan properti baru yang kita buat, kita bisa tambahkan underscore di awal penamaannya. Hal ini dibutuhkan karena jika kita menerapkan mode `closed` pada Shadow DOM, nilai properti `shadowRoot` akan mengembalikan `null`, sehingga tidak ada cara lain untuk kita mengakses *Shadow Tree*.

Setelah menerapkan Shadow DOM pada `constructor`, ketika ingin mengakses apapun yang merupakan properti dari DOM kita harus melalui `_shadowRoot`. Contohnya ketika ingin menerapkan template HTML, kita tidak bisa menggunakan langsung `this.innerHTML`, namun perlu melalui `this._shadowRoot.innerHTML`.

Sehingga kita perlu menyesuaikan kembali beberapa kode yang terdapat pada fungsi render menjadi seperti ini:

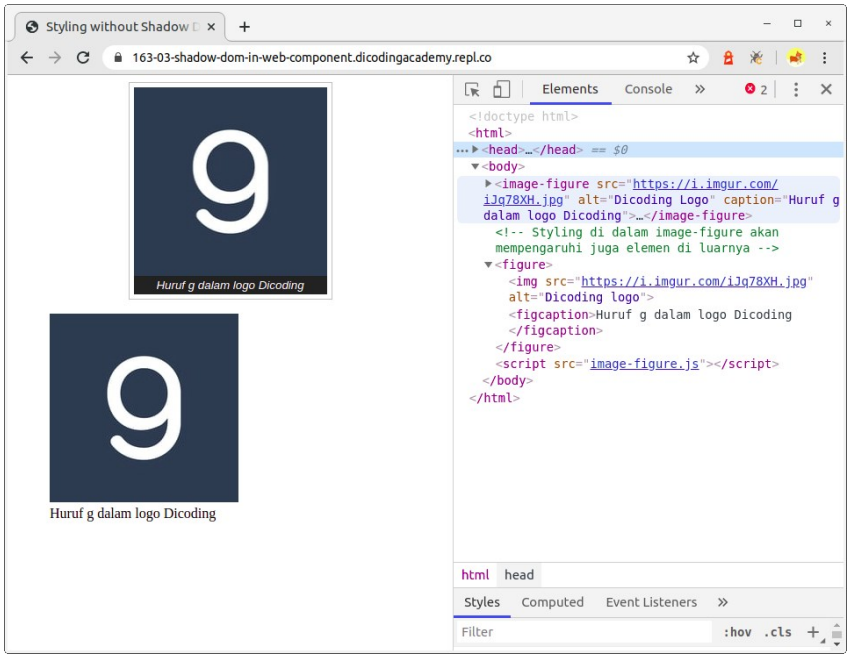
```
render() {
   this._shadowRoot.innerHTML = `
     <style>
       figure {
         border: thin #c0c0c0 solid;
         display: flex;
         flex-flow: column;
         padding: 5px;
         max-width: 220px;
         margin: auto;
       }

       figure > img {
         max-width: 220px;
       }

       figure > figcaption {
         background-color: #222;
         color: #fff;
         font: italic smaller sans-serif;
         padding: 3px;
         text-align: center;
       }
     </style>

     <figure>
       <img src="${this.src}"
            alt="${this.alt}">
```

Dengan begitu sekarang *styling* pada komponen hanya berlaku pada komponen itu sendiri. Begitu juga sebaliknya, *styling* yang dituliskan di luar dari komponen tidak akan berdampak pada elemen di dalam komponen.



← KEMBALI KE MATERI SEBELUMNYA

LANJUTKAN KE MATERI BERIKUTNYA →