The Shadow DOM: encapsulate styles or scripts

ENCAPSULATE CSS AND JS CODE IN YOUR TEMPLATES, AND HIDE WITH THE SHADOW DOM

By mixing templates and the shadow DOM, it's possible to embed a template's content in the shadow root. In this scenario, it's easy to encapsulate CSS styles or JavaScript code that will affect only the content of the shadow root. External CSS will not apply inside the shadow root.

This is an important feature as it protects the content of a new "widget" that is hidden in the shadow root, from external CSS/scripts, etc.

AN EXAMPLE THAT MIXES TEMPLATES AND SHADOW DOM:

HTML part:

```
<template id="mytemplate">
   <style>
     h1 {color:white; background:red}
   </style>
   <h1>This is a shadowed H1</h1>
</template>
```

The JavaScript part:

```
/ Instanciate the template
var t =document.querySelector('#mytemplate');
// Create a root node under our H1 title
var host = document.querySelector('h1');
var root = host.createShadowRoot();
```

```
// Put template content in the root node
root.appendChild(document.importNode(t.content, true));
```

Online example at JSBin:

Note that once again, the content shown is the shadow root + the styles applied. The styles applied are those defined *in the template's content* that has been cloned and put inside the shadow root.

Internal CSS will not apply outside the template/shadow DOM

The CSS inside the template will not affect other H1 elements in the page. This CSS rule (*lines 2-4* in the HTML part) will only apply to the template's content, with no side-effects on other elements outside.

Look at this example at JSBin that uses two H1s in the document: one is associated with a shadow root (defined in a template with an embedded CSS that selects H1 elements and makes them white on red), the other one is located in the body of the document is not affected by the CSS from the Web Component.

Beware: the included polyfill will not emulate CSS encapsulation. To see the real behavior, try with Chrome or Opera!

The HTML part:

```
<template id="mytemplate">
   <style>
     h1 {color:white; background:red}
```

```
    </style>
    <h1>This is a shadowed H1</h1>
</template>
<body>
    <h1 id="withShadowDom">This is a text header</h1>

    <h1>Normal header with no shadow DOM associated.</h1>
</body>
```

We added a new H1 at *line 11*. We added an `id` attribute to the first H1 and modified the JavaScript part to just select the first H1 in order to add a shadow root to it (*line 5* below).

JavaScript code:

```
/ Instantiate the template
var t =document.querySelector('#mytemplate');
// Create a root node under our H1 title
var host =document.querySelector('#withShadowDom');
var root = host.createShadowRoot();
// Put template content in the root node
root.appendChild(document.importNode(t.content, true));
```

And here is the result:

The second H1 is not affected by the CSS defined in the template used by the first H1.