Настройка стилей теневого DOM

Теневой DOM может содержать теги `<style>` и `<link rel="stylesheet" href="…">`. В последнем случае таблицы стилей кешируются по протоколу HTTP, так что они не будут загружаться повторно при использовании одного шаблона для многих компонентов.

Как правило, локальные стили работают только внутри теневого DOM, а стили документа – вне его. Но есть несколько исключений.

:host

Селектор `:host` позволяет выбрать элемент-хозяин (элемент, содержащий теневое дерево).

Например, мы создаём элемент `<custom-dialog>` который нужно расположить по-центру. Для этого нам необходимо стилизовать сам элемент `<custom-dialog>`.

Это именно то, что делает `:host`:

```
<template id="tmpl">
  <style>
    /* стиль будет применён изнутри к элементу <custom-
    :host {
      position: fixed;
      left: 50%;
      top: 50%;
      transform: translate(-50%, -50%);
      display: inline-block;
      border: 1px solid red;
      padding: 10px;
    }
  </style>
  <slot></slot>
</template>

<script>
customElements.define('custom-dialog', class extends HT
  connectedCallback() {
    this.attachShadow({mode: 'open'}).append(tmpl.conte
  }
});
</script>

<custom-dialog>
  Hello!
</custom-dialog>
```

Hello!

Каскадирование

Элемент-хозяин (элемент `<custom-dialog>`) находится в светлом DOM, поэтому к нему применяются CSS-стили документа.

Если есть некоторое свойство, стилизованное как в `:host` локально, так и в документе, то стиль документа будет приоритетным.

Например, если в документе из примера поставить:

```
<style>
```


```
2  custom-dialog {
3    padding: 0;
4  }
5  </style>
```

...то `<custom-dialog>` будет без `padding`.

Это очень удобно, поскольку мы можем задать стили «по умолчанию» в компоненте в его правиле `:host`, а затем, при желании, легко переопределить их в документе.

Исключение составляет тот случай, когда локальное свойство помечено как `!important`, для таких свойств приоритет имеют локальные стили.

:host(selector)

То же, что и `:host`, но применяется только в случае, если элемент-хозяин подходит под селектор `selector`.

Например, мы бы хотели выровнять по центру `<custom-dialog>`, только если он содержит атрибут `centered`:

```
1  <template id="tmpl">
2    <style>
3      :host([centered]) {
4        position: fixed;
5        left: 50%;
6        top: 50%;
7        transform: translate(-50%, -50%);
8        border-color: blue;
9      }
10
11     :host {
12       display: inline-block;
13       border: 1px solid red;
14       padding: 10px;
15     }
16   </style>
17   <slot></slot>
18 </template>
19
20 <script>
21 customElements.define('custom-dialog', class extends HTI
22   connectedCallback() {
23     this.attachShadow({mode: 'open'}).append(tmpl.conte
24   }
25 });
26 </script>
27
28
29 <custom-dialog centered>
30   Centered!
31 </custom-dialog>
32
33 <custom-dialog>
34   Not centered.
35 </custom-dialog>
```

Not centered.

Centered!

Теперь дополнительные стили для выравнивания по центру применяются только к первому элементу: `<custom-dialog centered>`.

:host-context(selector)

То же самое, что и `:host`, но применяется только в том случае, если элемент-хозяин или любой из его предков во внешнем документе подходит под селектор `selector`.

Например: правила в `:host-context(.dark-theme)` применятся, только если на элементе `<custom-dialog>` или где-то выше есть класс `dark-theme`:

```
<body class="dark-theme">
  <!--
    :host-context(.dark-theme) применится к custom-dial
  -->
  <custom-dialog>...</custom-dialog>
</body>
```

Подводя итог, мы можем использовать семейство селекторов `:host` для стилизации основного элемента компонента в зависимости от контекста. Эти стили (если только не стоит !important) могут быть переопределены документом.

Применение стилей к содержимому слотов

Теперь давайте рассмотрим ситуацию со слотами.

Элементы слотов происходят из светлого DOM, поэтому они используют стили документа. Локальные стили не влияют на содержимое слотов.

В примере ниже текст в `<span>` жирный в соответствии со стилями документа, но не берёт `background` из локальных стилей:

```
<style>
  span { font-weight: bold }
</style>

<user-card>
  <div slot="username"><span>John Smith</span></div>
</user-card>

<script>
customElements.define('user-card', class extends HTMLEl
  connectedCallback() {
    this.attachShadow({mode: 'open'});
    this.shadowRoot.innerHTML = `
      <style>
      span { background: red; }
      </style>
      Имя: <slot name="username"></slot>
    `;
  }
});
</script>
```

Имя:
John Smith

В результате текст жирный, но не красный.

Если мы хотим стилизовать слотовые элементы в нашем компоненте, то есть два варианта.

Первое – можно стилизовать сам `<slot>` и полагаться на наследование CSS:

```
<user-card>
  <div slot="username"><span>John Smith</span></div>
</user-card>

<script>
customElements.define('user-card', class extends HTMLEl
  connectedCallback() {
    this.attachShadow({mode: 'open'});
    this.shadowRoot.innerHTML = `
```

```
      <style>
      slot[name="username"] { font-weight: bold; }
      </style>
      Имя: <slot name="username"></slot>
    `;
  }
});
</script>
```

Имя:
John Smith

Здесь `<p>John Smith</p>` выделяется жирным шрифтом, потому что наследование CSS действует между `<slot>` и его содержимым. Но в CSS как таковом не все свойства наследуются.

Другой вариант – использовать псевдокласс `::slotted(селектор)`. Соответствует элементам, если выполняются два условия:

1. Это слотовый элемент, пришедший из светлого DOM. Имя слота не имеет значения. Просто любой элемент, вставленный в `<slot>`, но только сам элемент, а не его потомки.
2. Элемент соответствует `селектору`.

В нашем примере `::slotted(div)` выбирает в точности `<div slot="username">`, но не его дочерние элементы:

```
<user-card>
  <div slot="username">
    <div>John Smith</div>
  </div>
</user-card>

<script>
customElements.define('user-card', class extends HTMLEl
  connectedCallback() {
    this.attachShadow({mode: 'open'});
    this.shadowRoot.innerHTML = `
      <style>
      ::slotted(div) { border: 1px solid red; }
      </style>
      Name: <slot name="username"></slot>
    `;
  }
});
</script>
```

Name:
John Smith

Обратите внимание, что селектор `::slotted` не может спускаться дальше в слот. Эти селекторы недействительны:

```
::slotted(div span) {
  /* наш слот <div> не соответствует этому */
}

::slotted(div) p {
  /* не может войти в светлый DOM */
}
```

Кроме того, `::slotted` можно использовать только в CSS. Мы не можем использовать его в `querySelector`.

CSS-хуки с пользовательскими свойствами

Как можно стилизовать внутренние элементы компонента из основного документа?

Селекторы типа `:host` применяют правила к элементу `<custom-dialog>` или `<user-card>`, но как стилизовать элементы теневого DOM внутри них? Например, в `<user-card>` мы хотели бы разрешить внешнему документу изменять внешний вид пользовательских полей.

Аналогично тому, как мы предусматриваем у компонента методы, чтобы взаимодействовать с ним, мы можем использовать переменные CSS (пользовательские свойства CSS) для его стилизации.

Пользовательские свойства CSS существуют одновременно на всех уровнях, как светлом, так и в тёмном DOM.

Например, в теневом DOM мы можем использовать CSS-переменную `--user-card-field-color` для стилизации полей, а документ будет её устанавливать:

```
<style>
  .field {
    color: var(--user-card-field-color, black);
    /* если переменная --user-card-field-color не опред
  }
</style>
<div class="field">Имя: <slot name="username"></slot></
<div class="field">Дата рождения: <slot name="birthday"
```

Затем мы можем объявить это свойство во внешнем документе для `<user-card>`:

```
user-card {
  --user-card-field-color: green;
}
```

Пользовательские CSS свойства проникают через теневой DOM, они видны повсюду, поэтому внутреннее правило `.field` будет использовать его.

Вот полный пример::

```
<style>
  user-card {
    --user-card-field-color: green;
  }
</style>

<template id="tmpl">
  <style>
    .field {
      color: var(--user-card-field-color, black);
    }
  </style>
  <div class="field">Имя: <slot name="username"></slot>
  <div class="field">Дата рождения: <slot name="birthda
</template>

<script>
customElements.define('user-card', class extends HTMLEl
  connectedCallback() {
    this.attachShadow({mode: 'open'});
    this.shadowRoot.append(document.getElementById('tmp
  }
});
</script>

<user-card>
  <span slot="username">John Smith</span>
  <span slot="birthday">01.01.2001</span>
</user-card>
```

```
Имя: John Smith
Дата рождения: 01.01.2001
```


Итого

Теневой DOM может включать в себя стили, такие как `<style>` или `<link rel="stylesheet">`.

Локальные стили могут влиять на:

- теневое дерево,
- элемент-хозяин, при помощи псевдоклассов семейства `:host`,
- слотовые элементы (из светлого DOM), `::slotted(селектор)` позволяет стилизовать сами слотовые элементы, но не их дочерние элементы.

Стили документов могут влиять на:

- элемент-хозяин (так как он находится во внешнем документе)
- слотовые элементы и их содержимое (так как они также физически присутствуют во внешнем документе)

Когда свойства CSS конфликтуют, обычно стили документа имеют приоритет, если только свойство не помечено как `!important`. Тогда предпочтение отдаётся локальным стилям.

Пользовательские свойства CSS проникают через теневой DOM. Они используются как «хуки» для придания элементам стиля:

1. Компонент использует пользовательское CSS-свойство для стилизации ключевых элементов, например `var(--component-name-title, <значение по умолчанию>)`.
2. Автор компонента публикует эти свойства для разработчиков, они так же важны, как и другие общедоступные методы компонента.
3. Когда разработчик хочет стилизовать заголовок, он назначает CSS-свойство `--component-name-title` для элемента-хозяина или выше.
4. Profit!

Комментарии

перед тем как писать...