Стили и классы

Как правило, существует два способа задания стилей для элемента: Создать класс в CSS и использовать его: или писать стили непосредственно в атрибуте style: . JavaScript может менять и классы, и свойство style. Классы – всегда предпочтительный вариант по сравнению со style. Мы должны манипулировать свойством style только в том случае, если классы «не могут справиться». Например, использование style является приемлемым, если мы вычисляем координаты элемента динамически и хотим установить их из JavaScript.

Классы

elem.className – свойство для классов. Если мы присваиваем что-то elem.className, то это заменяет всю строку с классами.

elem.classList – объект с методами add/remove/toggle/contains, удобно для управления отдельными классами:

- elem.classList.**add/remove**("class") – добавить/удалить класс.
- elem.classList.**toggle**("class") – добавить класс, если его нет, иначе удалить.
- elem.classList.**contains**("class") – проверка наличия класса, возвращает true/false.

classList является перебираемым, поэтому можно перечислить все классы при помощи **for..of**

Стили

elem.style – это объект, который соответствует тому, что написано в атрибуте "style. Установка стиля elem.style.width="100px" работает так же, как наличие в атрибуте style строки width:100px.

Мы можем получить доступ с помощью elem.style только к свойствам из **атрибута style**.

Для свойства из нескольких слов используется camelCase:

```
background-color  => elem.style.backgroundColor
z-index           => elem.style.zIndex
border-left-width => elem.style.borderLeftWidth
button.style.MozBorderRadius = '5px';
button.style.WebkitBorderRadius = '5px';
```

Сброс стилей

Иногда нам нужно добавить свойство стиля, а потом, позже, убрать его.
Например, чтобы скрыть элемент, мы можем задать **elem.style.display = "none".**
Затем мы можем удалить свойство style.display, чтобы вернуться к первоначальному состоянию. Вместо delete elem.style.display мы должны присвоить ему пустую строку: **elem.style.display = "".**
Если мы установим в style.display пустую строку, то браузер применит CSS-классы и встроенные стили, как если бы такого свойства style.display вообще не было.

style.cssText - для задания нескольких стилей в одной строке используется специальное свойство:

```
div.style.cssText=`color: red !important;
text-align: center; `;
```

Это свойство редко используется, потому что такое присваивание удаляет все существующие стили: оно не добавляет, а заменяет их.

Вычисление стилей

getComputedStyle(element, [pseudo])

- element - Элемент, значения для которого нужно получить
- pseudo - Указывается, если нужен стиль псевдоэлемента, например ::before. Пустая строка или отсутствие аргумента означают сам элемент.

Результат вызова – объект со стилями, похожий на elem.style, но с учётом всех CSS-классов. Например:

```
var result = getComputedStyle(h3, ':after').content;
```

```
<style> body { color: red; margin: 5px } </style>
<script>
  let computedStyle = getComputedStyle(document.body);

  // сейчас мы можем прочитать отступ и цвет
  alert( computedStyle.marginTop ); // 5px
  alert( computedStyle.color ); // rgb(255, 0, 0)
</script
```

Вычисленное (computed) и окончательное (resolved) значения

Есть две концепции в CSS:

Вычисленное (computed) значение – это то, которое получено после применения всех CSS-правил и CSS-наследования. Например, height:1em или font-size:125%.

Окончательное (resolved) значение – непосредственно применяемое к элементу. Значения 1em или 125% являются относительными. Браузер берёт вычисленное значение и делает все единицы измерения фиксированными и абсолютными, например, height:20px или font-size:16px. Для геометрических свойств разрешённые значения могут иметь плавающую точку, например, width:50.5px.

Давным-давно getComputedStyle был создан для получения вычисленных значений, но оказалось, что окончательные значения гораздо удобнее, и стандарт изменился.Так что, в настоящее время getComputedStyle фактически возвращает **окончательное значение свойства**, для геометрии оно обычно в пикселях.

Для правильного получения значения нужно указать точное свойство. Например: paddingLeft, marginTop, borderTopWidth. При обращении к сокращённому: padding, margin, border – правильный результат не гарантируется.

Посещённые ссылки могут быть окрашены с помощью псевдокласса :visited.

Но getComputedStyle не даёт доступ к этой информации, чтобы произвольная страница не могла определить, посещал ли пользователь ту или иную ссылку, проверив стили.

JavaScript не видит стили, применяемые с помощью :**visited.** Кроме того, в CSS есть ограничение, которое запрещает в целях безопасности применять к :visited CSS-стили, изменяющие геометрию элемента.

На заметку:

Для свойств, имеющих еденицы измерения, полезно применять **parseInt**

```
const elementStyle = getComputedStyle(element); //20px
const paddingLeft = parseInt(elementStyle.paddingLeft);
```