Классы в виде строки: className

Атрибуту `"class"` соответствует свойство `className`.
Так как слово `"class"` является зарезервированным словом в JavaScript, то при проектировании DOM решили, что соответствующее свойство будет называться `className`.

Классы в виде объекта: classList

Атрибут `class` – уникален. Ему соответствует аж целых два свойства!
Работать с классами как со строкой неудобно. Поэтому, кроме `className`, в современных браузерах есть свойство `classList`.
Свойство `classList` – это объект для работы с классами.
Оно поддерживается в IE начиная с IE10, но его можно эмулировать в IE8+, подключив мини-библиотеку classList.js.
Методы `classList`:

- `elem.classList.contains("class")` – возвращает `true/false`, в зависимости от того, есть ли у элемента класс `class`.
- `elem.classList.add/remove("class")` – добавляет/удаляет класс `class`
- `elem.classList.toggle("class")` – если класса `class` нет, добавляет его, если есть – удаляет.

Нестандартные атрибуты

У каждого элемента есть некоторый набор стандартных свойств, например для `<a>` это будут `href`, `name`, а для `<img>` это будут `src`, `alt`, и так далее.
Точный набор свойств описан в стандарте, обычно мы более-менее представляем, если пользуемся HTML, какие свойства могут быть, а какие – нет.
Для нестандартных атрибутов DOM-свойство не создаётся.

- Атрибуты – это то, что написано в HTML.
- Свойство – это то, что находится внутри DOM-объекта.

Синхронизация между атрибутами и свойствами:

- Стандартные свойства и атрибуты синхронизируются: установка атрибута автоматически ставит свойство DOM. Некоторые свойства синхронизируются в обе стороны.
- Бывает так, что свойство не совсем соответствует атрибуту. Например, «логические» свойства вроде `checked`, `selected` всегда имеют значение `true/false`, а в атрибут можно записать произвольную строку.Выше мы видели другие примеры на эту тему, например `href`.

Нестандартные атрибуты:

- Нестандартный атрибут (если забыть глюки старых IE) никогда не попадёт в свойство, так что для кросс-браузерного доступа к нему нужно обязательно использовать `getAttribute`.
- Атрибуты, название которых начинается с `data-`, можно прочитать через `dataset`. Эта возможность не поддерживается IE10-.

Для того, чтобы избежать проблем со старыми IE, а также для более короткого и понятного кода старайтесь везде использовать свойства, а атрибуты – только там, где это *действительно* нужно.
А *действительно* нужны атрибуты очень редко – лишь в следующих трёх случаях:

1. Когда нужно кросс-браузерно получить нестандартный HTML-атрибут.
2. Когда нужно получить «оригинальное значение» стандартного HTML-атрибута, например, `<input value="...">`.
3. Когда нужно получить список всех атрибутов, включая пользовательские. Для этого используется коллекция `attributes`.

10. Методы contains и compareDocumentPosition

Метод contains для проверки на вложенность
Синтаксис:
```
var result = parent.contains(child);
```

Возвращает `true`, если `parent` содержит `child` или `parent == child`.

Метод compareDocumentPosition для порядка узлов
Бывает, что у нас есть два элемента, к примеру, `<li>` в списке, и нужно понять, какой из них выше другого.
Метод `compareDocumentPosition` – более мощный, чем `contains`, он предоставляет одновременно информацию и о содержании и об относительном порядке элементов.
Синтаксис:
```
var result = nodeA.compareDocumentPosition(nodeB);
```

Возвращаемое значение – битовая маска.

- Для проверки, является ли один узел предком другого, достаточно метода `nodeA.contains(nodeB)`.
- Для расширенной проверки на предшествование есть метод `compareDocumentPosition`.
- Для IE8 нужен полифил для `compareDocumentPosition`.
-