9. Атрибуты и DOM-свойства

При чтении HTML браузер генерирует DOM-модель. При этом большинство стандартных HTML-атрибутов становятся свойствами соответствующих объектов.

Свои DOM-свойства
Ранее мы видели некоторые встроенные свойства DOM-узлов. Но, технически, никто нас ими не ограничивает.
Узел DOM – это объект, поэтому, как и любой объект в JavaScript, он может содержать пользовательские свойства и методы.

DOM-свойства:
- Могут иметь любое значение.
- Названия свойств *чувствительны* к регистру.
- Работают за счёт того, что DOM-узлы являются объектами JavaScript.

Атрибуты
Элементам DOM, с другой стороны, соответствуют HTML-теги, у которых есть текстовые атрибуты.

Доступ к атрибутам осуществляется при помощи стандартных методов:
- `elem.hasAttribute(name)` – проверяет наличие атрибута
- `elem.getAttribute(name)` – получает значение атрибута
- `elem.setAttribute(name, value)` – устанавливает атрибут
- `elem.removeAttribute(name)` – удаляет атрибут

Также все атрибуты элемента можно получить с помощью свойства `elem.attributes`, которое содержит псевдо-массив объектов типа Attr.
В отличие от свойств, атрибуты:
- Всегда являются строками.
- Их имя *нечувствительно* к регистру (ведь это HTML)
- Видны в `innerHTML` (за исключением старых IE)

Исходное значение value
Изменение некоторых свойств обновляет атрибут. Но это скорее исключение, чем правило.
Чаще синхронизация – односторонняя: свойство зависит от атрибута, но не наоборот.

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`.