8. Современный DOM: полифилы

Для поиска полифила обычно достаточно ввести в поисковике `"polyfill"`, и нужное свойство либо метод. Как правило, полифилы идут в виде коллекций скриптов.

Полифилы хороши тем, что мы просто подключаем их и используем везде современный DOM/JS, а когда старые браузеры окончательно отомрут – просто выкинем полифил, без изменения кода.

Типичная схема работы полифила DOM-свойства или метода:

- Создаётся элемент, который его, в теории, должен поддерживать.
- Соответствующее свойство сравнивается с `undefined`.
- Если его нет – модифицируется прототип, обычно это `Element.prototype` – в него дописываются новые геттеры и функции.

Другие полифилы сделать сложнее. Например, полифил, который хочет добавить в браузер поддержку элементов вида `<input type="range">`, может найти все такие элементы на странице и обработать их, меняя внешний вид и работу через JavaScript. Это возможно. Но если уже существующему `<input>` поменять `type` на `range` – полифил не «подхватит» его автоматически.

Описанная ситуация нормальна. Не всегда полифил обеспечивает идеальную поддержку наравне с родными свойствами. Но если мы не собираемся так делать, то подобный полифил вполне подойдёт.

Один из лучших сервисов для полифилов: polyfill.io. Он даёт возможность вставлять на свою страницу скрипт с запросом к сервису, например:

```
<script src="//cdn.polyfill.io/v1/polyfill.js?features=es6"></
script>
```

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.