КРАТКО:

Каждый DOM-узел принадлежит определённому классу. Классы формируют иерархию. Весь набор свойств и методов является результатом наследования.
Главные свойства DOM-узла:
`nodeType`
Свойство `nodeType` позволяет узнать тип DOM-узла. Его значение – числовое: `1` для элементов, `3` для текстовых узлов, и т.д. Только для чтения.
`nodeName/tagName`
Для элементов это свойство возвращает название тега (записывается в верхнем регистре, за исключением XML-режима). Для узлов-неэлементов `nodeName` описывает, что это за узел. Только для чтения.
`innerHTML`
Внутреннее HTML-содержимое узла-элемента. Можно изменять.
`outerHTML`
Полный HTML узла-элемента. Запись в `elem.outerHTML` не меняет `elem`. Вместо этого она заменяет его во внешнем контексте.
`nodeValue/data`
Содержимое узла-неэлемента (текст, комментарий). Эти свойства практически одинаковые, обычно мы используем `data`. Можно изменять.
`textContent`
Текст внутри элемента: HTML за вычетом всех `<тегов>`. Запись в него помещает текст в элемент, при этом все специальные символы и теги интерпретируются как текст. Можно использовать для защиты от вставки произвольного HTML кода.
`hidden`
Когда значение установлено в `true`, делает то же самое, что и CSS `display:none`.
В зависимости от своего класса DOM-узлы имеют и другие свойства. Например у элементов `<input>` (`HTMLInputElement`) есть свойства `value`, `type`, у элементов `<a>` (`HTMLAnchorElement`) есть`href` и т.д. Большинство стандартных HTML-атрибутов имеют соответствующие свойства DOM.

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>
```