getElementsBy*

Существуют также другие методы поиска элементов по тегу, классу и так далее.
На данный момент, они скорее исторические, так как `querySelector` более чем эффективен.
Здесь мы рассмотрим их для полноты картины, также вы можете встретить их в старом коде.

- `elem.getElementsByTagName(tag)` ищет элементы с данным тегом и возвращает их коллекцию. Передав `"*"` вместо тега, можно получить всех потомков.
- `elem.getElementsByClassName(className)` возвращает элементы, которые имеют данный CSS-класс.
- `document.getElementsByName(name)` возвращает элементы с заданным атрибутом `name`. Очень редко используется.

Безусловно, наиболее часто используемыми в настоящее время являются методы `querySelector` и `querySelectorAll`, но и методы `getElementBy*` могут быть полезны в отдельных случаях, а также встречаются в старом коде.
Кроме того:

- Есть метод `elem.matches(css)`, который проверяет, удовлетворяет ли элемент CSS-селектору.
- Метод `elem.closest(css)` ищет ближайшего по иерархии предка, соответствующему данному CSS-селектору. Сам элемент также включён в поиск.
-

6. Внутреннее устройство поисковых методов

Несмотря на схожесть в синтаксисе, поисковые методы `get*` и `querySelector*` внутри устроены очень по-разному.
document.getElementById(id)
Браузер поддерживает у себя внутреннее соответствие `id -> элемент`. Поэтому нужный элемент возвращается сразу. Это очень быстро.
elem.querySelector(query), elem.querySelectorAll(query)
Чтобы найти элементы, удовлетворяющие поисковому запросу, браузер не использует никаких сложных структур данных.
Он просто перебирает все подэлементы внутри элемента `elem`(или по всему документу, если вызов в контексте документа) и проверяет каждый элемент на соответствие запросу `query`.
Вызов `querySelector` прекращает перебор после первого же найденного элемента, а `querySelectorAll` собирает найденные элементы в «псевдомассив»: внутреннюю структуру данных, по сути аналогичную массиву JavaScript.
Этот перебор происходит очень быстро, так как осуществляется непосредственно движком браузера, а не JavaScript-кодом.
Оптимизации:

- В случае поиска по ID: `elem.querySelector('#id')`, большинство браузеров оптимизируют поиск, используя вызов `getElementById`.
- Последние результаты поиска сохраняются в кеше. Но это до тех пор, пока документ как-нибудь не изменится.

elem.getElementsBy*(...)
Результаты поиска `getElementsBy*` – живые! При изменении документа – изменяется и результат запроса.

Способ Firefox
Перебрать подэлементы `document.body` в порядке их появления в поддереве. Запоминать *все найденные элементы* во внутренней структуре данных, чтобы при повторном обращении обойтись без поиска.
Разбор действий браузера при выполнении кода выше:

1. Браузер создаёт пустую «живую коллекцию» `elems`. Пока ничего не ищет.
2. Перебирает элементы, пока не найдёт первый `div`. Запоминает его и возвращает.
3. Перебирает элементы дальше, пока не найдёт элемент с индексом `995`. Запоминает все найденные.
4. Возвращает ранее запомненный элемент с индексом `500`, без дополнительного поиска!
5. Продолжает обход поддерева с элемента, на котором остановился (`995`) и до конца. Запоминает найденные элементы и возвращает их количество.