Способ WebKit
Перебирать подэлементы `document.body`. Запоминать только один, *последний найденный*, элемент, а также, по окончании перебора – длину коллекции.
Здесь кеширование используется меньше.
Разбор действий браузера по строкам:

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 его и возвращает.
4. Браузер запоминает только последний найденный, поэтому не помнит об элементе `500`. Нужно найти его перебором поддерева. Этот перебор можно начать либо с начала – вперёд по поддереву, 500-й по счету) либо с элемента `995` – назад по поддереву, 495-й по счету. Так как назад разница в индексах меньше, то браузер выбирает второй путь и идёт от 995-го назад 495 раз. Запоминает теперь уже 500-й элемент и возвращает его.
5. Продолжает обход поддерева с 500-го (не 995-го!) элемента и до конца. Запоминает число найденных элементов и возвращает его.

Основное различие – в том, что Firefox запоминает все найденные, а Webkit – только последний. Таким образом, «метод Firefox» требует больше памяти, но гораздо эффективнее при повторном доступе к предыдущим элементам.
А «метод Webkit» ест меньше памяти и при этом работает не хуже в самом важном и частом случае – последовательном переборе коллекции, без возврата к ранее выбранным.
Запомненные элементы сбрасываются при изменениях DOM.
Документ может меняться. При этом, если изменение может повлиять на результаты поиска, то запомненные элементы необходимо сбросить. Например, добавление нового узла `div` сбросит запомненные элементы коллекции `elem.getElementsByTagName('div')`.
Сбрасывание запомненных элементов при изменении документа выполняется интеллектуально.

1. Во-первых, при добавлении элемента будут сброшены только те коллекции, которые могли быть затронуты обновлением. Например, если в документе есть два независимых раздела `<section>`, и поисковая коллекция привязана к первому из них, то при добавлении во второй – она сброшена не будет.
   Если точнее – будут сброшены все коллекции, привязанные к элементам вверх по иерархии от непосредственного родителя нового `div` и выше, то есть такие, которые потенциально могли измениться. И только они.
2. Во-вторых, если добавлен только `div`, то не будут сброшены запомненные элементы для поиска по другим тегам, например `elem.getElementsByTagName('a')`.
3. …И, конечно же, не любые изменения DOM приводят к сбросу кешей, а только те, которые могут повлиять на коллекцию. Если где-то добавлен новый атрибут элементу – с кешем для `getElementsByTagName` ничего не произойдёт, так как атрибут никак не может повлиять на результат поиска по тегу.

7. Свойства узлов: тип, тег и содержимое

Классы DOM-узлов
У разных DOM-узлов могут быть разные свойства. Например, у узла, соответствующего тегу `<a>`, есть свойства, связанные со ссылками, а у соответствующего тегу `<input>` – свойства, связанные с полем ввода и т.д.
Текстовые узлы отличаются от узлов-элементов. Но у них есть общие свойства и методы, потому что все классы DOM-узлов образуют единую иерархию.
Каждый DOM-узел принадлежит соответствующему встроенному классу.
Корнем иерархии является EventTarget, от него наследует Node и остальные DOM-узлы.

Существуют следующие классы:

- EventTarget – это корневой «абстрактный» класс. Объекты этого класса никогда не создаются. Он служит основой, благодаря которой все DOM-узлы поддерживают так называемые «события», о которых мы поговорим позже.
- Node – также является «абстрактным» классом, и служит основой для DOM-узлов. Он обеспечивает базовую функциональность: `parentNode`, `nextSibling`, `childNodes` и т.д. (это геттеры). Объекты класса `Node` никогда не создаются. Но есть определённые классы узлов, которые наследуют от него: `Text` – для текстовых узлов, `Element` – для узлов-элементов и более экзотический `Comment` – для узлов-комментариев.
- Element – это базовый класс для DOM-элементов. Он обеспечивает навигацию на уровне элементов: `nextElementSibling`, `children` и методы поиска: `getElementsByTagName`, `querySelector`. Браузер поддерживает не только HTML, но также XML и SVG. Класс Element служит базой для следующих классов: `SVGElement`, `XMLElement` и `HTMLElement`.
- HTMLElement – является базовым классом для всех остальных HTML-элементов. От него наследуют конкретные элементы:
    - HTMLInputElement – класс для тега `<input>`,
    - HTMLBodyElement – класс для тега `<body>`,
    - HTMLAnchorElement – класс для тега `<a>`,
    - …и т.д, каждому тегу соответствует свой класс, который предоставляет определённые свойства и методы.

Свойство «nodeType»
Свойство `nodeType` предоставляет ещё один, «старомодный» способ узнать «тип» DOM-узла.
Его значением является цифра:

- `elem.nodeType == 1` для узлов-элементов,
- `elem.nodeType == 3` для текстовых узлов,
- `elem.nodeType == 9` для объектов документа,