18-го августа 2019

Итого

Материал на этой странице устарел, поэтому скрыт из оглавления сайта.

В этой главе кратко перечислены основные свойства и методы DOM, которые мы изучили. Их уже довольно много.

Используйте её, чтобы по-быстрому вспомнить и прокрутить в голове то, что изучали ранее. Все ли эти свойства вам знакомы?

Кое-где стоит ограничение на версии IE, но на все свойства можно сделать или найти полифил, с которым их можно использовать везде.

`document.createElement(tag)`

Создать элемент с тегом `tag`

`document.createTextNode(txt)`

Создать текстовый узел с текстом `txt`

`node.cloneNode(deep)`

Клонировать существующий узел, если `deep=false`, то без потомков.

Свойства узлов

`node.nodeType`

Тип узла: 1(элемент) / 3(текст) / другие.

`elem.tagName`

Тег элемента.

`elem.innerHTML`

HTML внутри элемента.

`elem.outerHTML`

Весь HTML элемента, включая сам тег. На запись использовать с осторожностью, так как не модифицирует элемент, а вставляет новый вместо него.

`node.data / node.nodeValue`

Содержимое узла любого типа, кроме элемента.

`node.textContent`

Текстовое содержимое узла, для элементов содержит текст с вырезанными тегами (IE9+).

`elem.hidden`

Если поставить `true`, то элемент будет скрыт (IE10+).

Атрибуты

`elem.getAttribute(name)`, `elem.hasAttribute(name)`, `elem.setAttribute(name, value)`

Чтение атрибута, проверка наличия и запись.

`elem.dataset.*`

Значения атрибутов вида `data-*` (IE10+).

Ссылки

`document.documentElement`

Элемент `<HTML>`

`document.body`

Элемент `<BODY>`

`document.head`

Элемент `<HEAD>` (IE9+)

По всем узлам:

- `parentNode`
- `nextSibling previousSibling`
- `childNodes firstChild lastChild`

Только по элементам:

- `parentElement`
- `nextElementSibling previousElementSibling`
- `children`, `firstElementChild lastElementChild`

Все они IE9+, кроме `children`, который работает в IE8-, но содержит не только элементы, но и комментарии (ошибка в браузере).

Дополнительно у некоторых типов элементов могут быть и другие ссылки, свойства, коллекции для навигации, например для таблиц:

`table.rows[N]`

строка `TR` номер `N`.

`tr.cells[N]`

ячейка `TH/TD` номер `N`.

`tr.sectionRowIndex`

номер строки в таблице в секции `THEAD/TBODY`.

`td.cellIndex`

номер ячейки в строке.

Поиск

`*.querySelector(css)`

По селектору, только первый элемент

`*.querySelectorAll(css)`

По селектору CSS3, в IE8 по CSS 2.1

`document.getElementById(id)`

По уникальному `id`

`document.getElementsByName(name)`

По атрибуту `name`, в IE9- работает только для элементов, где `name` предусмотрен стандартом.

`*.getElementsByTagName(tag)`

По тегу `tag`

`*.getElementsByClassName(class)`

По классу, IE9+, корректно работает с элементами, у которых несколько классов.

Вообще, обычно можно использовать только `querySelector/querySelectorAll`. Методы `getElement*` работают быстрее (за счёт более оптимальной внутренней реализации), но в 99% случаев это различие очень небольшое и роли не играет.

Дополнительно есть методы:

`elem.matches(css)`

Проверяет, подходит ли элемент под CSS-селектор.

`elem.closest(css)`

Ищет ближайший элемент сверху по иерархии DOM, подходящий под CSS-селектор. Первым проверяется сам `elem`. Этот элемент возвращается.

`elemA.contains(elemB)`

Возвращает `true`, если `elemA` является предком (содержит) `elemB`.

`elemA.compareDocumentPosition(elemB)`

Возвращает битовую маску, которая включает в себя отношение вложенности между `elemA` и `elemB`, а также – какой из элементов появляется в DOM первым.

Изменение

- `parent.appendChild(newChild)`
- `parent.removeChild(child)`
- `parent.insertBefore(newChild, refNode)`
- `parent.insertAdjacentHTML("beforeBegin|afterBegin|beforeEnd|afterEnd", html)`
- `parent.insertAdjacentElement("beforeBegin|...|afterEnd", element)` (кроме FF)
- `parent.insertAdjacentText("beforeBegin|...|afterEnd", text)` (кроме FF)
- `document.write(...)`

Скорее всего, понадобятся полифилы для:

- `node.append(...nodes)`
- `node.prepend(...nodes)`
- `node.after(...nodes)`,
- `node.before(...nodes)`
- `node.replaceWith(...nodes)`

Классы и стили

`elem.className`

Атрибут `class`

`elem.classList.add(class) remove(class) toggle(class) contains(class)` :Управление классами, для IE9- есть эмуляция.

`elem.style`

Стили в атрибуте `style` элемента

`getComputedStyle(elem, "")`

Стиль, с учётом всего каскада, вычисленный и применённый (только чтение)

Размеры и прокрутка элемента

`clientLeft/Top`

Ширина левой/верхней рамки `border`

`clientWidth/Height`

Ширина/высота внутренней части элемента, включая содержимое и `padding`, не включая полосу прокрутки (если есть).

`scrollWidth/Height`

Ширина/высота внутренней части элемента, с учётом прокрутки.

`scrollLeft/Top`

Ширина/высота прокрученной области.

`offsetWidth/Height`

Полный размер элемента: ширина/высота, включая `border`.

Размеры и прокрутка страницы

- ширина/высота видимой области: `document.documentElement.clientHeight`
- прокрутка(чтение): `window.pageYOffset || document.documentElement.scrollTop`
- прокрутка(изменение):
  - `window.scrollBy(x,y)` : на x,y относительно текущей позиции.
  - `window.scrollTo(pageX, pageY)` : на координаты в документе.
  - `elem.scrollIntoView(true/false)` : прокрутить, чтобы `elem` стал видимым и оказался вверху окна(`true`) или внизу(`false`)

Координаты

- относительно окна: `elem.getBoundingClientRect()`
- относительно документа: `elem.getBoundingClientRect()` + прокрутка страницы
- получить элемент по координатам: `document.elementFromPoint(clientX, clientY)`

Список намеренно сокращён, чтобы было проще найти то, что нужно.

Комментарии

перед тем как писать...