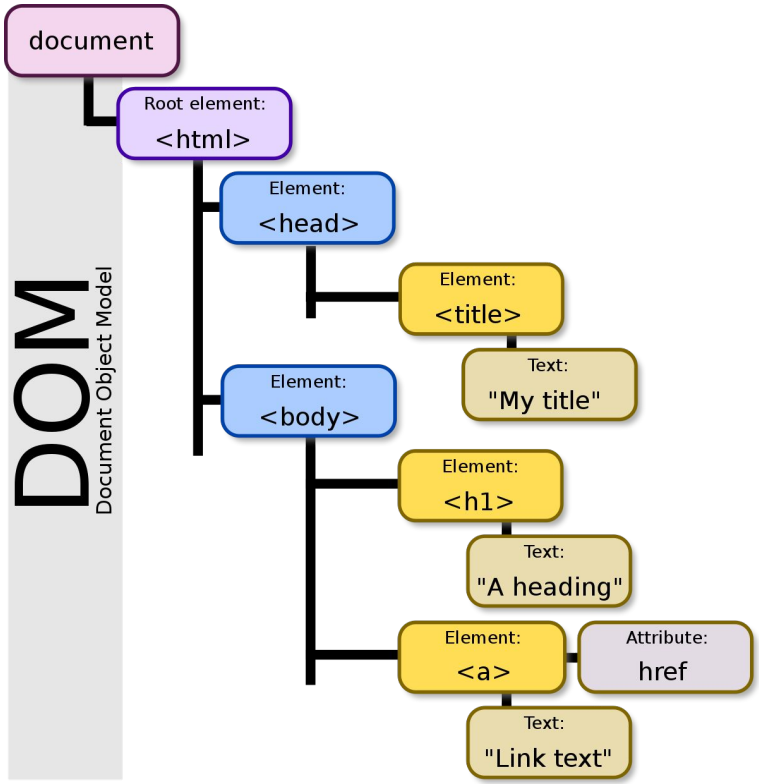
Основы JavaScript

Document Object Model

Понятие DOM

DOM – это представление HTML-документа в виде дерева объектов, доступное для изменения через JavaScript.



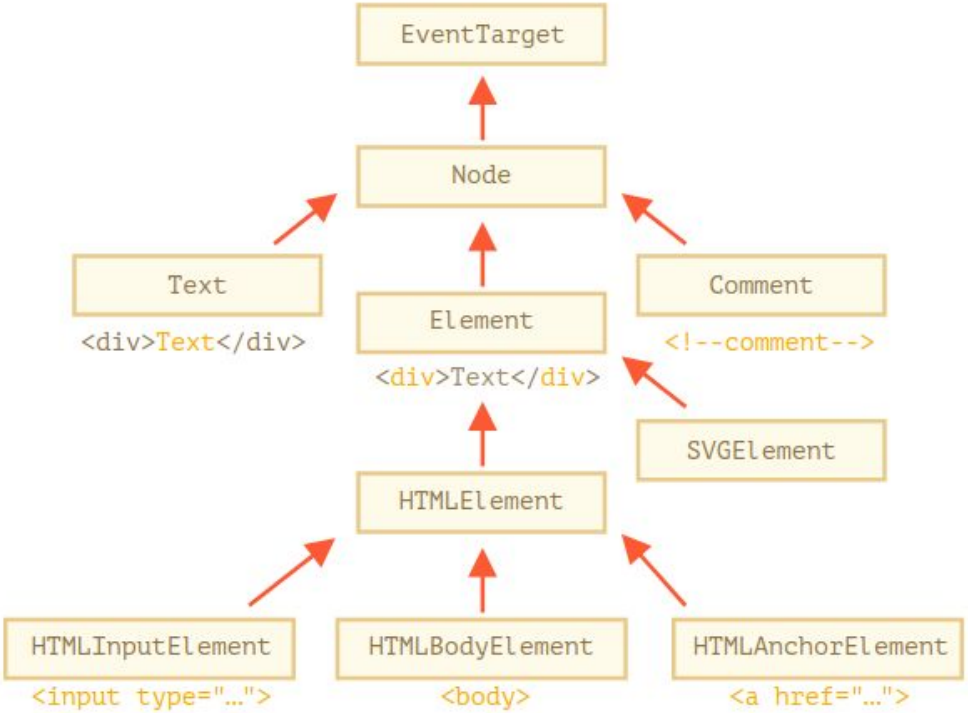


Схема наследования классов DOM API

Навигация по DOM

`document`

Объект для взаимодействия с DOM

`document.documentElement`

Получение ссылки на объект элемента <html>

`document.body`

Получение ссылки на объект элемента <body>

Навигация по DOM

Перемещение по узлам:

- `element.parentNode`
- `element.previousSibling`
- `element.nextSibling`
- `element.firstChild`
- `element.lastChild`
- `element.childNodes` (коллекция, НЕ массив)

Навигация по DOM

Перемещение по элементам:

- `element.parentElement`
- `element.previousElementSibling`
- `element.nextElementSibling`
- `element.firstElementChild`
- `element.lastElementChild`
- `element.children` (коллекция, НЕ массив)

Работа с Таблицами (Свойства Таблицы)

Таблицы имеют свои дополнительные ссылки для более удобной навигации.

Начнем с самого тега таблицы. Пускай переменная **table** содержит ссылку на элемент таблицы на странице.

- **`table.rows`** – коллекция строк TR таблицы
- **`table.caption/tHead/tFoot`** – ссылки на элементы таблицы CAPTION, THEAD, TFOOT
- **`table.tBodies`** – коллекция элементов таблицы TBODY, по спецификации их может быть несколько

Работа с Таблицами (Свойства Строки)

`tr.cells`

Коллекция ячеек <td> и <th>

`tr.sectionRowIndex`

Номер строки в текущей секции <thead> | <tbody> | <tfoot>

`tr.rowIndex`

Номер строки в таблице

(**`tr`** это переменная, которая содержит ссылку на элемент строки таблицы)

Работа с Таблицами (Свойства Ячейки)

Пусть **td** это переменная, которая содержит ссылку на элемент строки таблицы.

`td.cellIndex`

Номер ячейки таблицы в строке

Поиск элементов

Методы типа getElement(s):

- `document.getElementById(elementId)`
- `document.getElementsByTagName(tagName)`
- `document.getElementsByName(elementName)`
- `document.getElementsByClassName(className)`

Поиск элементов

Селекторы:

- `document.querySelector(selector)`
- `document.querySelectorAll(selector)`

Поиск элементов

```
element.matches(selector)
```

Определяет, соответствует ли селектор элементу

```
element.closest(selector)
```

Возвращает ближайший родительский элемент (или сам элемент), который соответствует заданному CSS-селектору или null, если таковых элементов вообще нет

Взаимодействие с узлами

- element.innerText - (!) НЕстандартное свойство
- element.innerHTML
- element.textContent
- a
  - element.href
- input, select, textarea
  - element.value
- element.id

Работа с атрибутами

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

Эти методы работают со значением, которое находится в HTML.

Data-атрибуты

С помощью нестандартных атрибутов можно привязать к элементу данные, которые будут доступны в JavaScript. Как правило, это делается при помощи атрибутов с названиями, начинающимися на **data-**.

```
<div id="elem" data-about="Elephant" data-user-location="street">
  По улице прошёлся слон. Весьма красив и толст был он.
</div>

<script>
  alert( elem.dataset.about ); // Elephant
  alert( elem.dataset.userLocation ); // street
</script>
```

Создание элемента

Для создания элементов используются следующие методы:

- **`document.createElement(tag)`**
- **`document.createTextNode(text)`**

Добавление элементов

Для вставки внутрь parentElem есть следующий метод:

```
parentElement.appendChild(elem)
```

Для вставки **element** в коллекцию детей **parentElement**, перед элементом **nextSibling** используется метод:

```
parentElem.insertBefore(element, nextSibling)
```

Клонирование / Удаление / Замена Узла

- **`element.cloneNode(deep)`** – клонирует элемент, если deep == true, то со всеми потомками, если false – без потомков
- **`parent.removeChild(element)`** – удаляет элемент из родительского
- **`parent.replaceChild(newElement, element)`** – заменяет элемент другим внутри родителя

Стилизация элементов. Свойство style

Свойство element.style возвращает объект, который дает доступ к стилю элемента на чтение и запись.

С его помощью можно изменять большинство CSS-свойств, например **`element.style.width="100px`**" работает так, как будто у элемента в атрибуте прописано: style="width:100px".

```
element.style.backgroundColor = “#0088ee”
```

Стилизация элементов. Функция getComputedStyle()

Для того, чтобы получить текущее используемое значение свойства, используется метод **`window.getComputedStyle(element[, pseudo])`**.

element

Элемент, значения для которого нужно получить

pseudo

Указывается, если нужен стиль псевдо-элемента, например "::before". Пустая строка или отсутствие аргумента означают сам элемент.

Свойства .className и .classList