JavaScript Day - 10

DOM(Document Object Model)

What is DOM (Document Object Model)

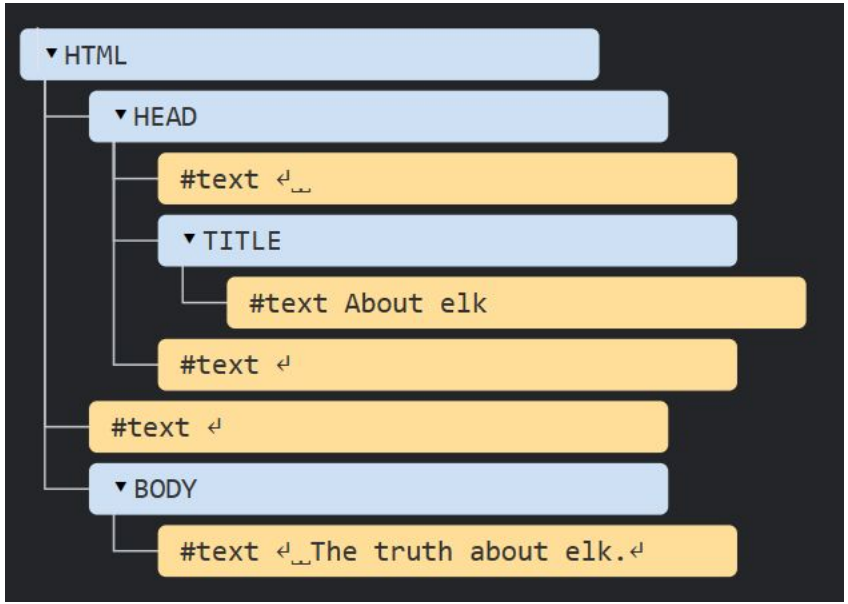
The Document Object Model (DOM) is a programming interface that represents an HTML document as a tree structure, **where each node is an object** representing a part of the document.

It allows programs and scripts to dynamically access and update the content, structure, and style of a document.

What is DOM (Document Object Model) ?

The DOM represents HTML as a tree structure of tags.

Here's how it looks:



DOM (Document Object Model)

The Document Object Model, or DOM for short, **represents all page content as objects that can be modified.**

The **document object** is the main “entry point” to the page. We can change or create anything on the page using it.

DOM (Document Object Model) Tree

According to the Document Object Model (DOM), **every HTML tag is an object**. **Nested tags are “children” of the enclosing one**.

Walking the DOM

The DOM allows us to do anything with elements and their contents, but first we need to reach the corresponding DOM object.

All operations on the DOM start with the document object. That's the main "entry point" to DOM. From it we can access any node.

Walking the DOM

The topmost tree nodes are available directly as document properties:

- <html> = document.documentElement
- <body> = document.body
- <head> = document.head

Walking the DOM

How can we access nested elements inside the top level elements ?

- childNodes
- firstChild
- lastChild

Walking the DOM

As we can see, childNodes looks like an array. But actually it's not an array, but rather a collection – a special array-like iterable object.

There are two important consequences:

1. We can use for..of to iterate over it
2. Array methods won't work, because it's not an array
3.

The first thing is nice. The second is tolerable, because we can use **Array.from** to **create a "real" array from the collection**, if we want array methods:

Walking the DOM

How can we access siblings and parents ?

1. nextSibling
2. previousSibling
3. parentNode

Walking the DOM

How can we do element only Navigation i.e **Ignore** #**text nodes**

1. children
2. firstElementChild, lastElementChild
3. previousElementSibling, nextElementSibling

Questions ?

If elem – is an arbitrary DOM element node…

1. Is it true that elem.lastChild.nextSibling is always null?
2. Is it true that elem.children[0].previousSibling is always null ?

Answers

1. Yes, true. The element elem.lastChild is always the last one, it has no nextSibling.

2. No, wrong, because elem.children[0] is the first child among elements. But there may exist non-element nodes before it. So previousSibling may be a text node.