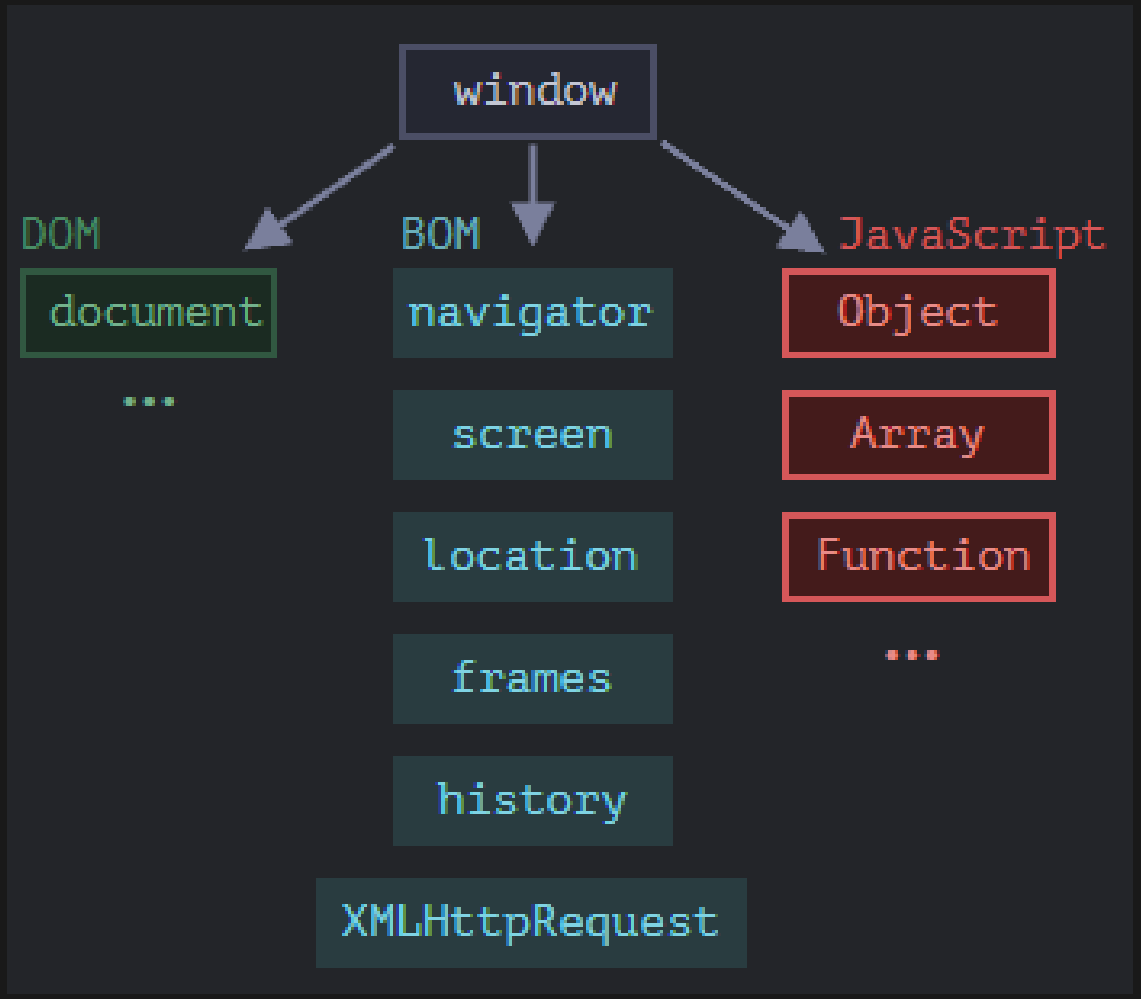
BROWSER JAVASCRIPT

ANDREAS DRANIDIS

BROWSER ENVIRONMENT

There’s a “root” object called window. It has two roles:

- it is a global object for JavaScript code
- it represents the “browser window” and provides methods to control it



BROWSER ENVIRONMENT

```
// as a global object
function sayHi() {
  alert("Hello");
}

// global functions are methods of the global object:
window.sayHi();

// as a browser window
alert(window.innerHeight); // inner window height
```

DOM (DOCUMENT OBJECT MODEL)

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.

```
// change the background color to red
document.body.style.background = "red";

// change it back after 1 second
setTimeout(() => document.body.style.background = "", 1000);
```

BOM (BROWSER OBJECT MODEL)

The Browser Object Model (BOM) represents additional objects provided by the browser (host environment) for working with everything except the document.

- The navigator object provides background information about the browser and the operating system.
- The location object allows us to read the current URL and can redirect the browser to a new one.

```
alert(location.href); // shows current URL
if (confirm("Go to Wikipedia?")) {
  location.href = "https://wikipedia.org"; // redirect the browser to another URL
}
```

DOM TREE

According to the Document Object Model (DOM), every HTML tag is an object. Nested tags are “children” of the enclosing one. The text inside a tag is an object as well.

For example, document.body is the object representing the <body> tag.

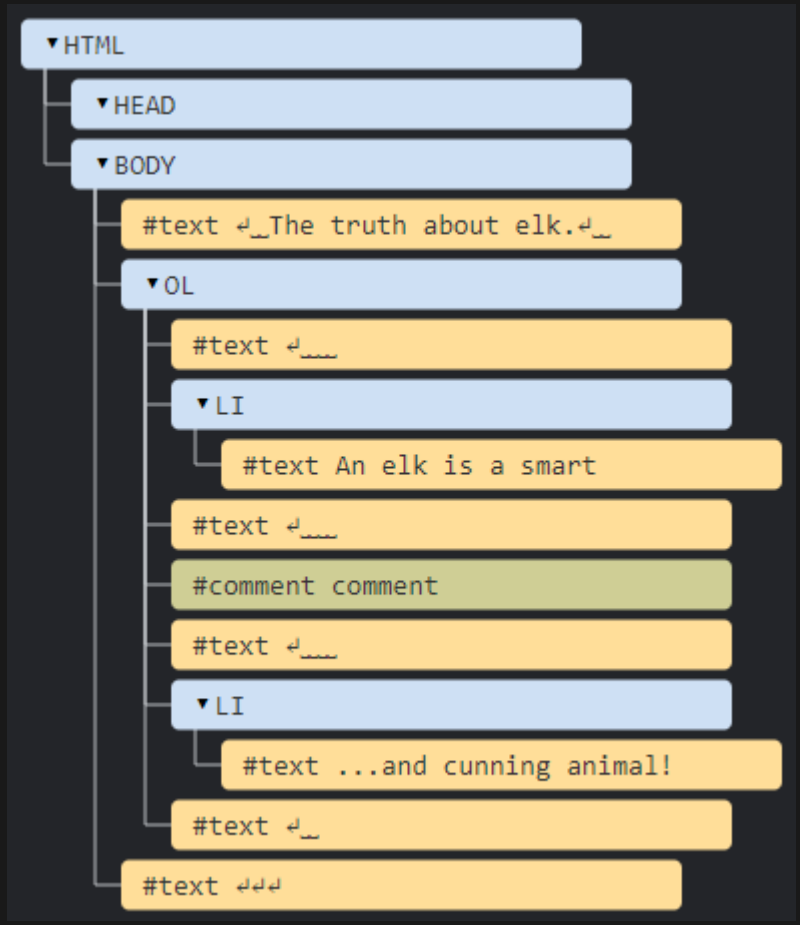
```
document.body.style.background = 'red';

setTimeout(() => document.body.style.background = '', 3000);
```

NODE TYPES

```
<!DOCTYPE HTML>
<html>
<body>
  The truth about elk.
  <ol>
    <li>An elk is a smart</li>
    <!-- comment -->
    <li>...and cunning animal!</li>
  </ol>
</body>
</html>
```

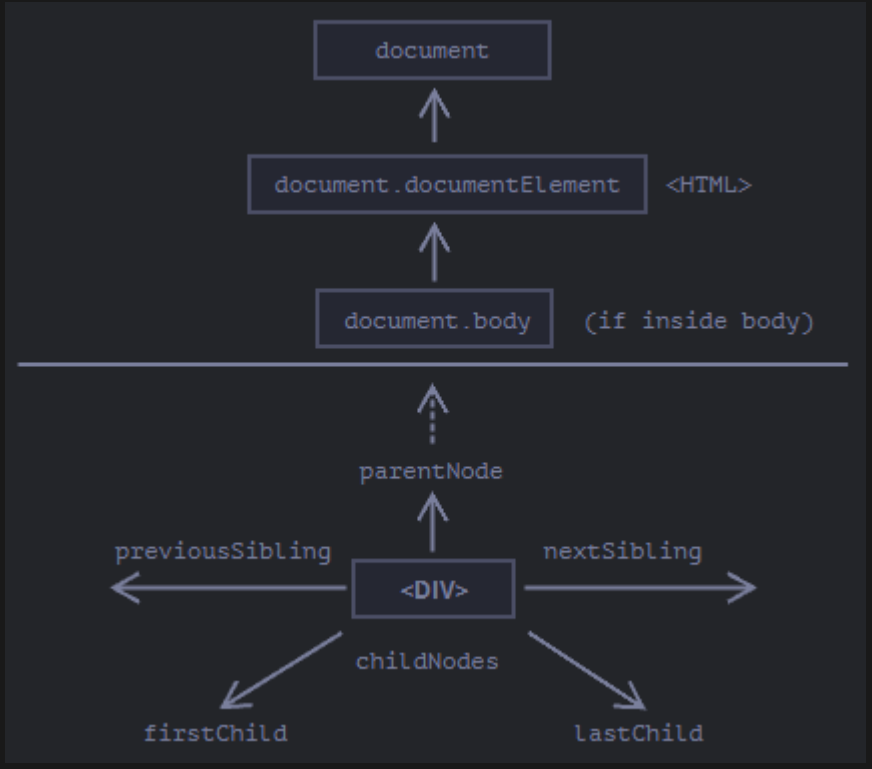
- Tags are element nodes
- The text inside elements are text nodes
- Comments are comment nodes

Everything in HTML, even comments, becomes a part of the DOM.



WALKING THE DOM

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.



- <html> = document.documentElement
- <body> = document.body
- <head> = document.head
- Children: childNodes, firstChild, lastChild
- Siblings: nextSibling, previousSibling
- Parent: parentNode

ELEMENT-ONLY NAVIGATION

- Children: children , firstElementChild, lastElementChild
- Siblings: previousElementSibling, nextElementSibling
- Parent: parentElement

SEARCHING THE DOM

document.getElementById or just id: If an element has the id attribute, we can get the element using the method document.getElementById(id), no matter where it is.

```
<div id="elem">
  <div id="elem-content">Element</div>
</div>

<script>
  // get the element
  let elem = document.getElementById('elem');

  // make its background red
  elem.style.background = 'red';

  // elem is a reference to DOM-element with id="elem"
  elem.style.background = 'red';

  // id="elem-content" has a hyphen inside, so it can't be a variable name
  // ...but we can access it using square brackets: window['elem-content']
```

THE ID MUST BE UNIQUE

SEARCHING THE DOM

querySelectorAll & querySelector

```
<ul>
  <li>The</li>
  <li>test</li>
</ul>
<ul>
  <li>has</li>
  <li>passed</li>
</ul>
<script>
  let elements = document.querySelectorAll('ul > li:last-child');

  for (let elem of elements) {
    alert(elem.innerHTML); // "test", "passed"
  }

  // elem.querySelector(css) is the same as elem.querySelectorAll(css)[0]
```

DOM CONTENTS

innerHTML: the contents

```
<body>
  <p>A paragraph</p>
  <div>A div</div>

  script
    alert( document.body.innerHTML ); // read the current contents
    document.body.innerHTML = 'The new BODY!'; // replace it
  script

</body>
```

outerHTML: full HTML of the element

```
<div id="elem">Hello <b>World</b></div>

script
  alert(elem.outerHTML); // <div id="elem">Hello <b>World</b></div>
script
```

HTML ATTRIBUTES

In HTML, tags may have attributes. When the browser parses the HTML to create DOM objects for tags, it recognizes standard attributes and creates DOM properties from them.

```
<body id="body" type="...">
  <input id="input" type="text">
  script
    alert(input.type); // text
    alert(body.type); // undefined: DOM property not created, because it's non-standard
  script
</body>
```

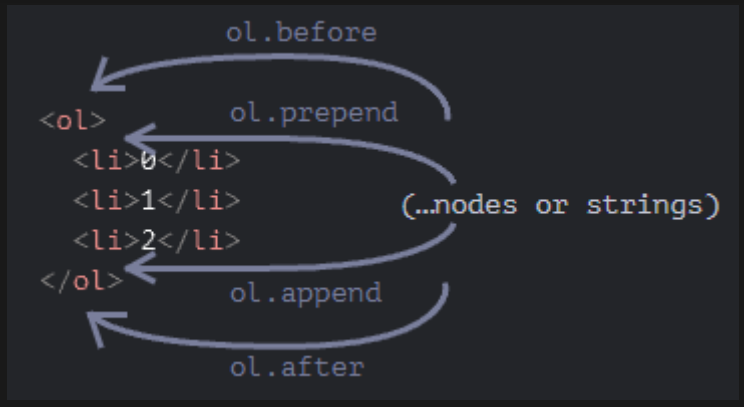
MODIFYING THE DOCUMENT

Creating an element

```
// 1. Create <div> element
let div = document.createElement('div');

// 2. Set its class to "alert"
div.className = "alert";

// 3. Fill it with the content
div.innerHTML = "<strong>Hi there!</strong> You've read an important message.";
```

To make the div show up, we need to insert it somewhere into document. For instance, into <body> element, referenced by document.body.

```
document.body.append(div);
```



STYLES AND CLASSES

className and classList

```
<body class="main page">
  script
    alert(document.body.className); // main page

    // add a class
    document.body.classList.add('article');

    alert(document.body.className); // main page article
  script
</body>
```

```
Methods of classList:

elem.classList.add/remove("class") // adds/removes the class.
elem.classList.toggle("class") // adds the class if it doesn't exist, otherwise removes it.
elem.classList.contains("class") // checks for the given class, returns true/false.
```

STYLES AND CLASSES

Element style

```
elem.style.width="100px"

background-color  => elem.style.backgroundColor
z-index           => elem.style.zIndex
border-left-width => elem.style.borderLeftWidth
```

BROWSER EVENTS

```
Mouse events:

click // when the mouse clicks on an element (touchscreen devices generate it on a tap).
contextmenu // when the mouse right-clicks on an element.
mouseover / mouseout // when the mouse cursor comes over / leaves an element.
mousedown / mouseup // when the mouse button is pressed / released over an element.
mousemove // when the mouse is moved.

Keyboard events:

keydown and keyup // when a keyboard key is pressed and released.

Form element events:

submit // when the visitor submits a <form>.
focus // when the visitor focuses on an element, e.g. on an <input>.

Document events:

DOMContentLoaded // when the HTML is loaded and processed, DOM is fully built.
```

EVENT HANDLERS

To react on events we can assign a handler – a function that runs in case of an event.

HTML-attribute

```
<input value="Click me" onclick="alert('Click!')" type="button">

script
  function countRabbits() {
    for(let i=1; i<=3; i++) {
      alert("Rabbit number " + i);
    }
  }
script

<input type="button" onclick="countRabbits()" value="Count rabbits!">
```

DOM property

```
<input id="elem" type="button" value="Click me">
script
  elem.onclick = function() {
    alert('Thank you');
  };
script
```

ACCESSING THE ELEMENT: THIS

The value of this inside a handler is the element. The one which has the handler on it.

```
<button onclick="alert(this.innerHTML)">Click me</button>
```

POSSIBLE MISTAKES

```
// right
button.onclick = sayThanks;

// wrong
button.onclick = sayThanks();

<input type="button" id="button" onclick="sayThanks()">

button.onclick = function() {
  sayThanks(); // <-- the attribute content goes here
};
```

ADDEVENTLISTENER

```
input.onclick = function() { alert(1); }
// ...
input.onclick = function() { alert(2); } // replaces the previous handler
```

```
element.addEventListener(event, handler, [options]);

elem.addEventListener( "click" , () => alert('Thanks!'));
// ....
elem.removeEventListener( "click", () => alert('Thanks!'));

function handler() {
  alert( 'Thanks!' );
}

input.addEventListener("click", handler);
// ....
input.removeEventListener("click", handler);
```

EVENT OBJECT

To properly handle an event we'd want to know more about what's happened. Not just a "click" or a "keydown", but what were the pointer coordinates? Which key was pressed?

```
<input type="button" value="Click me" id="elem">

script
  elem.onclick = function(event) {
    // show event type, element and coordinates of the click
    alert(event.type + " at " + event.currentTarget);
    alert("Coordinates: " + event.clientX + ":" + event.clientY);
  };
script
```

BROWSER DEFAULT ACTIONS

Many events automatically lead to certain actions performed by the browser.

- A click on a link – initiates navigation to its URL.
- A click on a form submit button – initiates its submission to the server.
- Pressing a mouse button over a text and moving it – selects the text.

```
<a href="/" onclick="return false">Click here</a>
or
<a href="/" onclick="event.preventDefault()">here</a>
```

LET'S PRACTICE

- Change the contents of a paragraph
  - by selecting with id
  - by selecting with css selector
- Change anchor url link
- Display the value of a text input
- Block copying of paragraph and show message
- Block form submission when input is empty
- Check if address has a number upon submission
- Check if email has '@' symbol
- Change img src (does this work?)