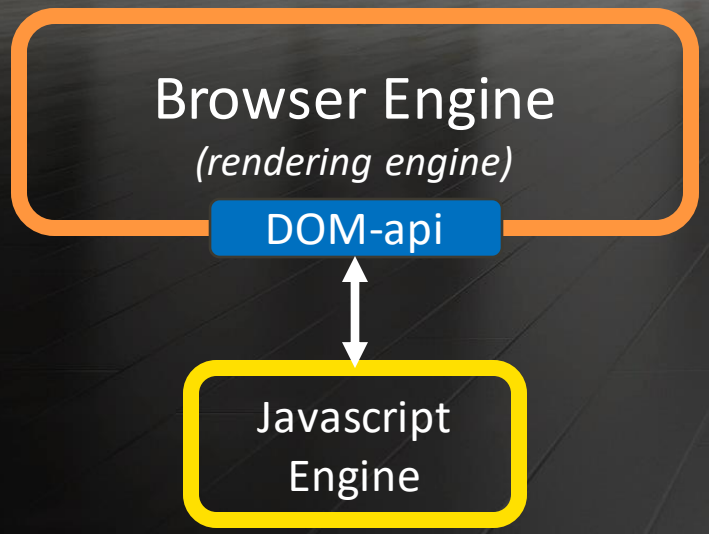
HTML Elements

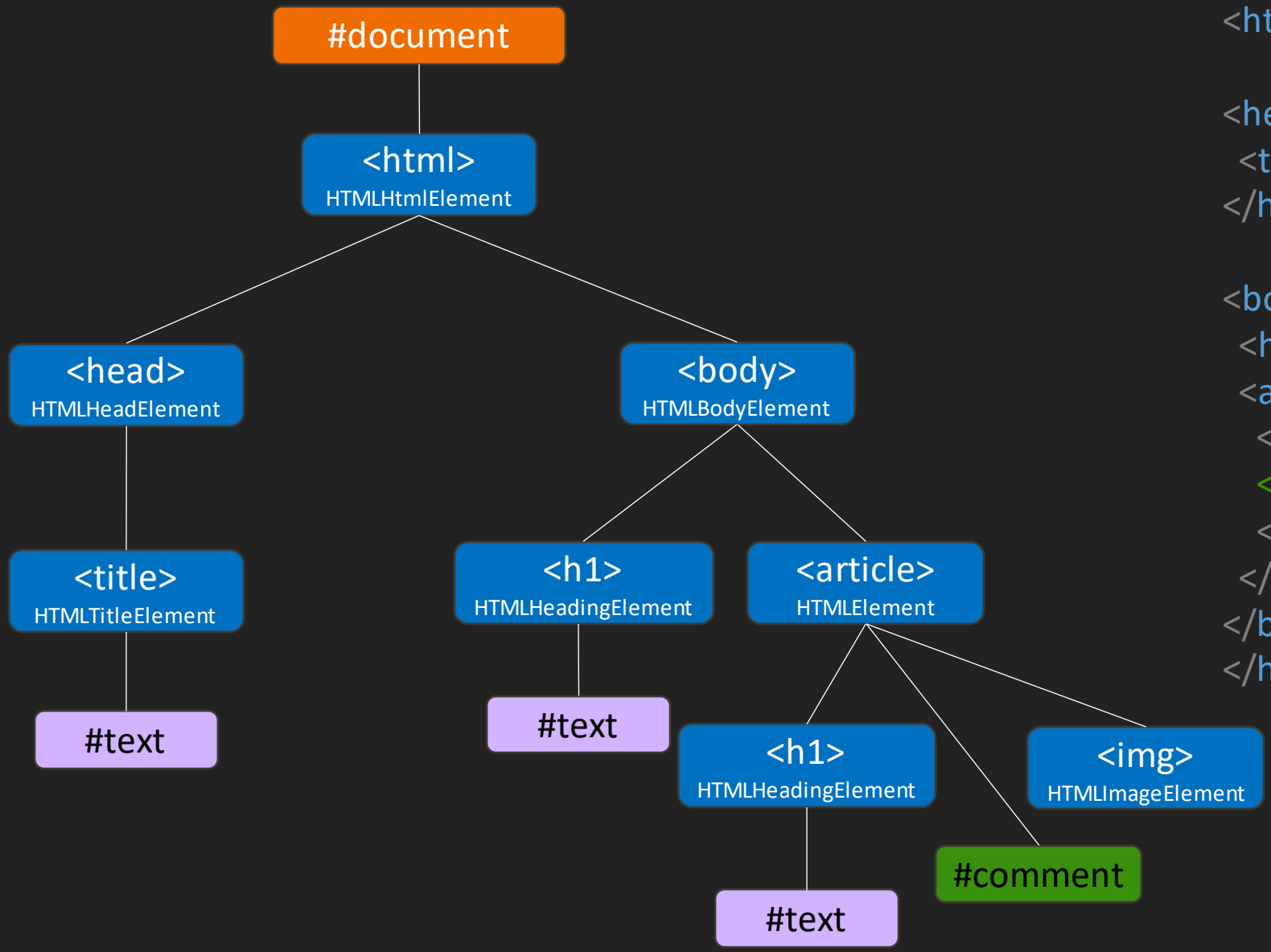
The Document Object Model

Document Object Model (DOM)

The DOM is a browser API that makes the javascript engine communicate with the browser engine.



Document Object Model (DOM)

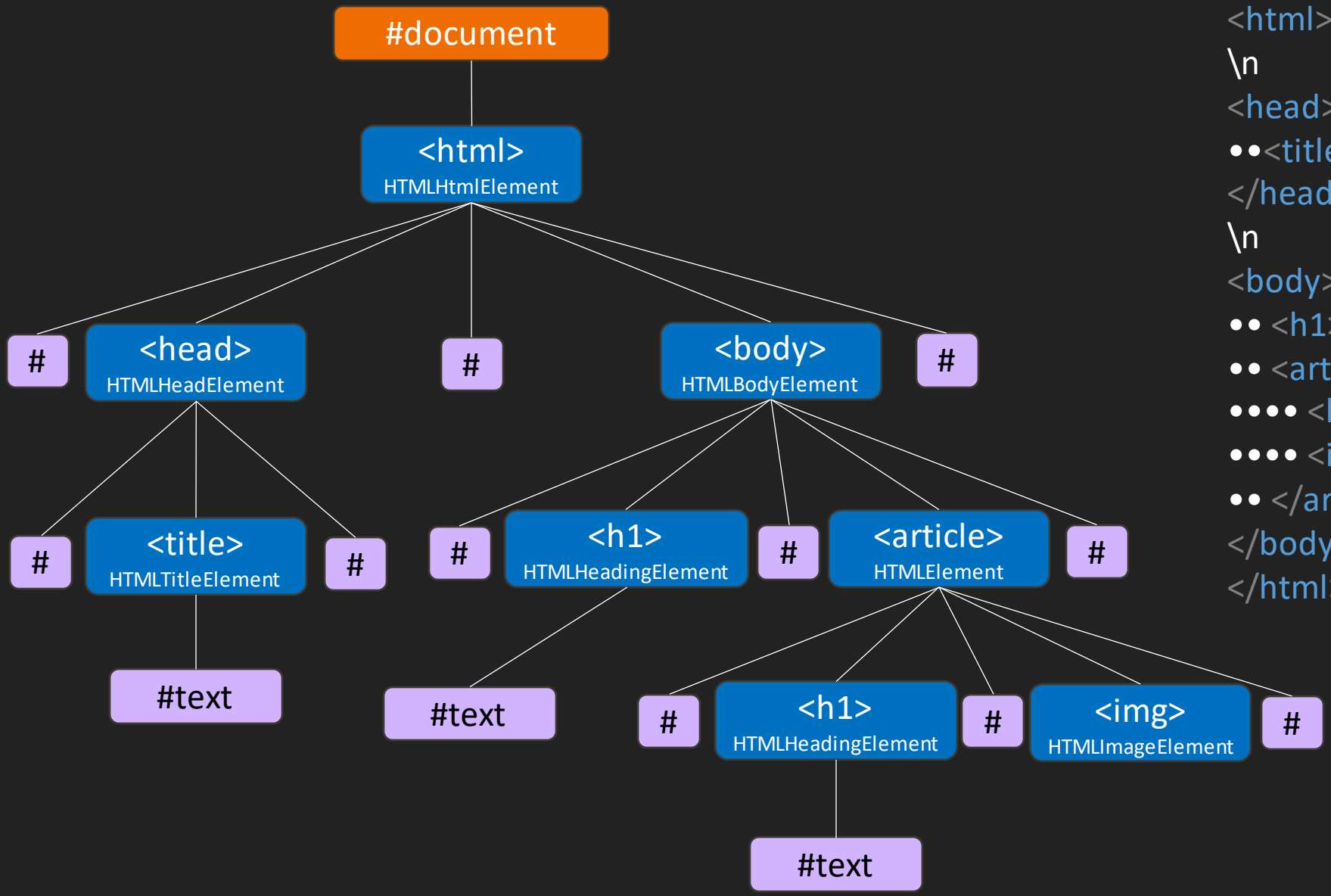



```
<!DOCTYPE html>
<html>

<head>
 <title>Page 1</title>
</head>

<body>
 <h1>Incidents</h1>
 <article>
  <h1>Bugs</h1>
  <!– TODO: Write your..
  <img />
 </article>
</body>
</html>
```

Node types

	Name	Value
<h1> HTMLHeadingElement	ELEMENT_NODE	1
#text	TEXT_NODE	3
#comment	COMMENT_NODE	8
#document	DOCUMENT_NODE	9

White spaces




```
<!DOCTYPE html>\n
<html>\n
\n
<head>\n
••<title>Page 1</title>\n
</head>\n
\n
<body>\n
•• <h1>Incidents</h1>\n
•• <article>\n
•••• <h1>Bugs</h1>\n
•••• <img />\n
•• </article>\n
</body>\n
</html>\n
```

Selecting Nodes

	Argument	Returns
getElementById	Id	Element
getElementsByTagName	tag-name	HTMLCollection *live*
getElementsByClassName	class-name	HTMLCollection *live*

js/index.js

```
const mainDiv = document.getElementById('main')

const pTags = document.getElementsByTagName('p')
console.log(pTags.length)

const tags = document.getElementsByClassName('post')
console.log(tags.length)
```

Selectors API

One API to rule them all

	returns
querySelector	Element
querySelectorAll	Node-list (static)

./js/index.js

```
const mainDiv = document.querySelector('#main')

const pTags = document.querySelectorAll('p')

const tags = document.querySelectorAll('p.tag')
```

Node-list and HTML-Collection

NodeList.prototype !== Array.prototype

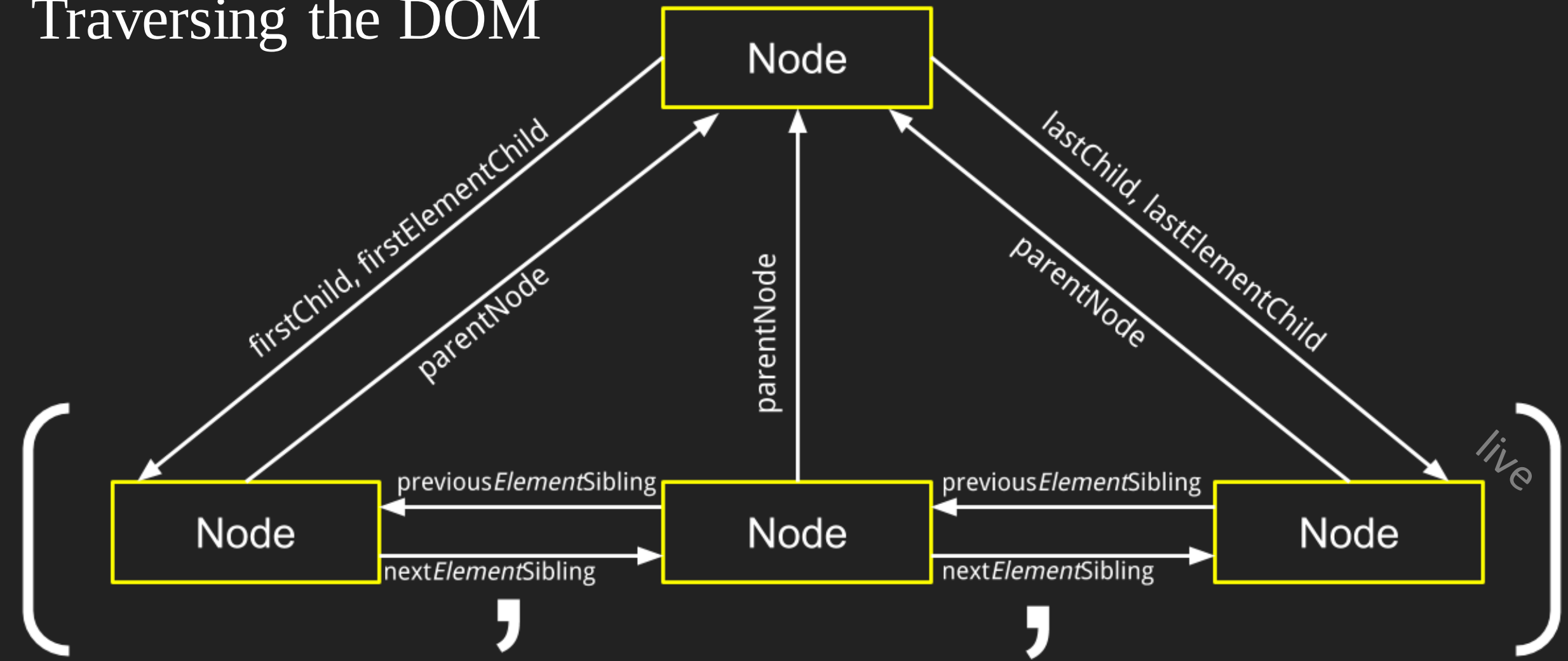
forEach implemented on Node-list but not on HTMLCollections

```
pTagsNodeList.forEach(element => {
  // Do something
})
```

Convert to an array to use with other array-methods

```
const pTagsArray = Array.from(pTagsNodeList)
```

Traversing the DOM

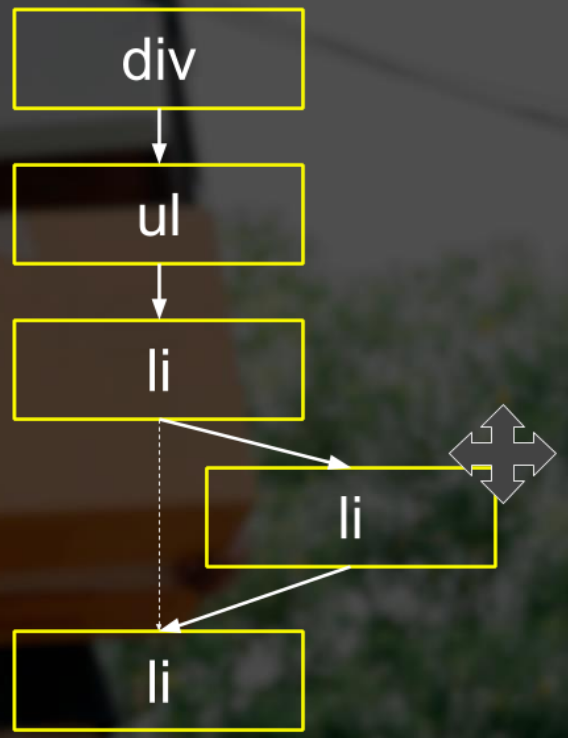


childNodes *All nodes including textnodes (NodeList)*
children *All element nodes (HTMLCollection)*

Manipulation

How to move, insert, delete, copy nodes in the DOM

- Adding nodes
  ```
  node.appendChild(newNode)
  node.insertBefore(newNode, beforeNode)
  ```
- Replacing nodes
  ```
  node.replaceChild(newNode, oldNode)
  ```
- Removing nodes
  ```
  node.removeChild(oldNode)
  ```
- Making clones
  ```
  node.cloneNode(bool)
  ```



Creating Nodes

document.createElement – Create a new ELEMENT_NODE

document.createTextNode – Create a new TEXT_NODE

./js/index.js

```
const newTag = document.createElement('p')
const newText = document.createTextNode('Cool text!')

newTag.appendChild(newText)

document.querySelector('#main').appendChild(newTag)
```

Shortcuts

- textContent
  - Creates one new TEXT_NODE
  - Removes all children form element

```
element.textContent = 'Text...'
```

- innerHTML
  - Parses the string
  - Creates elements and text nodes
  - Removes all children from element
  - Less secure. Use textContent!

```
element.innerHTML = '<p>Some text</p>'
```

Attributes

- `getAttribute('attribute name')`
- `setAttribute('attribute name', value)`
- `removeAttribute('attribute name')`

```
const newTag = document.createElement('img')

newTag.setAttribute('src', 'image/picture.svg')

// The browser fetches and caches the 'picture.svg'

document.querySelector('#main').appendChild(newTag)
```

Separation of concerns SOC

.html	Structure
.css	Presentation
.js	Behavior

Styles

./js/index.js

```
const node = document.querySelector('#discovery')

node.style.color = '#AA5698'
```

./index.html

```
<p id="discovery" style="color: #AA5698">Hello</p>
```

CSS Property	Identifier
font-size	fontSize
margin-left	marginLeft
...	...
float	cssFloat

Use Classes Instead

./js/index.js

```
const node = document.querySelector('#discovery')

node.classList.add('jschanged')
```

./index.html

```
<p id="discovery" class="whatever jschanged">Hello</p>
```

./css/styles.css

```
.jschanged {
  color: #AA5698;
}
```

classList.add
classList.remove
classList.toggle
classList.contains

Mix HTML and JS?

./js/index.js

```
let div = document.createElement('div')
div.classList.add('post')
let ul = document.createElement('ul')
let li1 = document.createElement('li')
let li2 = document.createElement('li')
li1.classList.add('active')
let a1 = document.createElement('a')
a1.setAttribute('href', '#')
let text1 = document.createTextNode('The first link')
a1.appendChild(text1)
let a2 = document.createElement('a')
a2.setAttribute('href', '#')
let text2 = document.createTextNode('The second link')
a2.appendChild(text2)
li1.appendChild(a1)
```

Templates

./index.html

```
<template id='post-template’>
 <div class='post’>
  <ul>
   <li class='active’>
    <a href='#'>The first link</a>
   </li>
   <li>
    <a href='#'>The second link</a>
   </li>
  </ul>
 </div>
</template>
```

./js/index.js

```
const template = document.querySelector('#post-template')
const clone = template.content.cloneNode(true)
document.querySelector('body').appendChild(clone)
```

Mustache, Underscore Templates, HandlebarsJS, Pug....

Templates as strings in JavaScript

./js/index.js

```
const template = document.createElement('template')
template.innerHTML = `
 <div class='post'>
  <ul>
   <li class='active'>
    <a href='#'>The first link</a>
   </li>
   <li>
    <a href='#'>The second link</a>
   </li>
  </ul>
 </div>`
```

./js/index.js

```
const clone = template.content.cloneNode(true)
document.querySelector('body').appendChild(clone)
```