

# Virtual DOM & Reconciliation

React & Next.js Official Docs Study

DongMin Kim

# Contents

## 1. Virtual DOM

**\* What is Virtual DOM?**

## 2. Reconciliation

**\* What is Reconciliation?**

# DOM

## Document Object Model (DOM)

### ▼ Guides

[Introduction to the DOM](#)[Using the Document Object Model](#)[Traversing an HTML table with JavaScript and DOM Interfaces](#)[Locating DOM elements using selectors](#)[Introduction to events](#)[How whitespace is handled by HTML, CSS, and in the DOM](#)

# Document Object Model (DOM)

The **Document Object Model (DOM)** connects web pages to scripts or programming languages by representing the structure of a document—such as the HTML representing a web page—in memory.

Usually it refers to JavaScript, even though modeling HTML, SVG, or XML documents as objects are not part of the core JavaScript language.

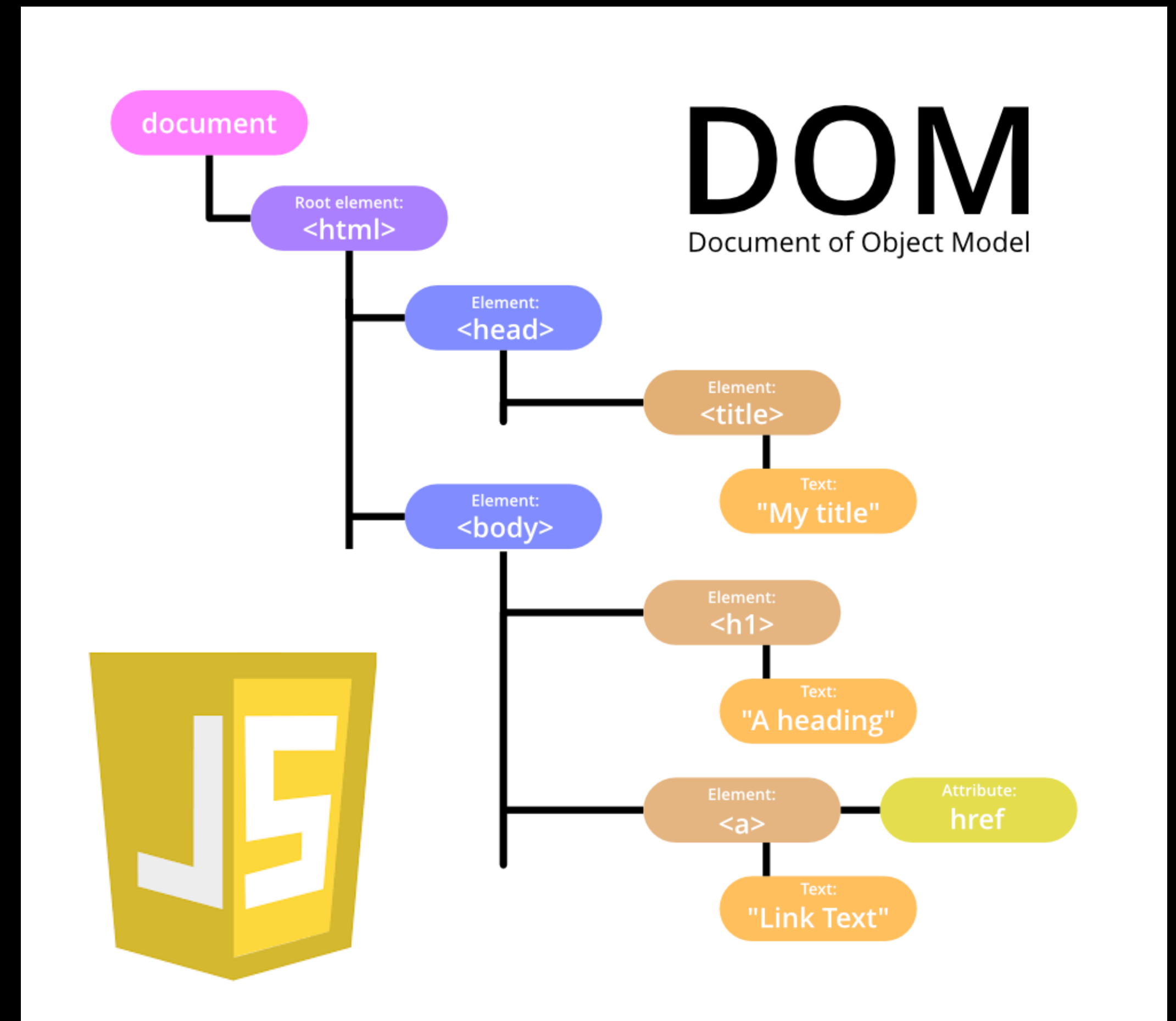
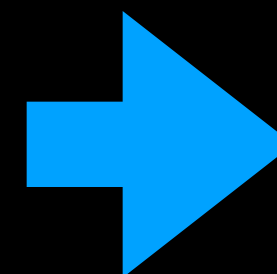
The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree. With them, you can change the document's structure, style, or content.

Nodes can also have event handlers attached to them. Once an event is triggered, the event handlers get executed.

## In this article

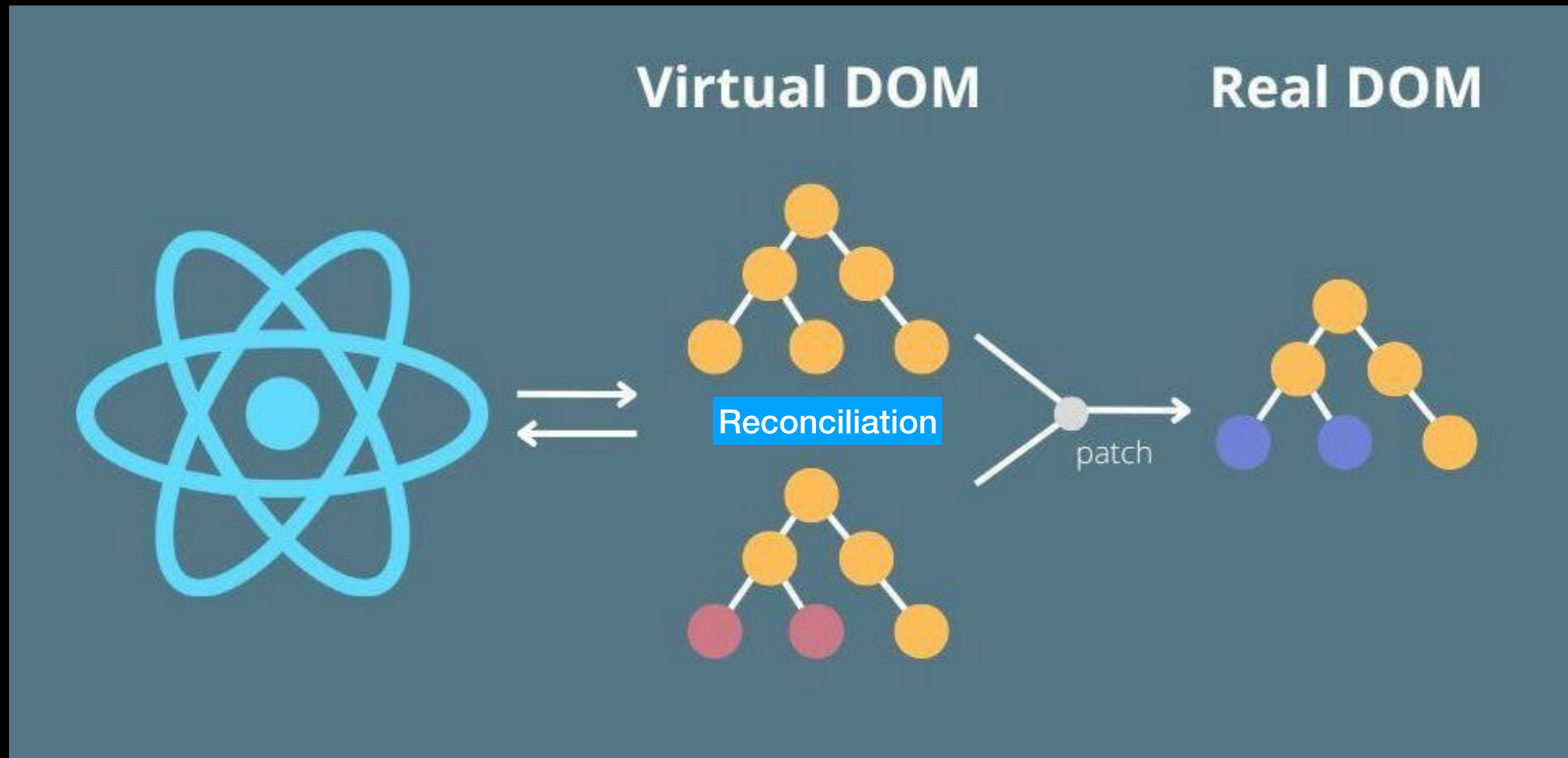
[DOM interfaces](#)[HTML DOM](#)[SVG DOM](#)[Specifications](#)[See also](#)

# DOM

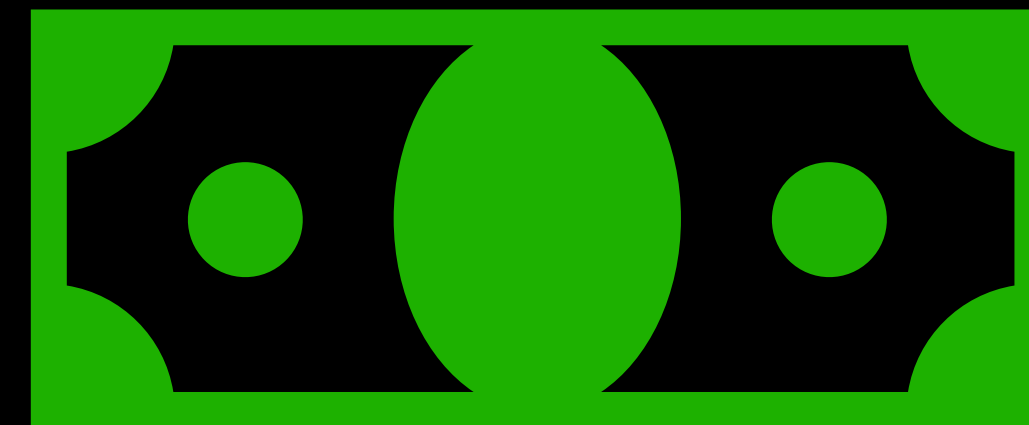
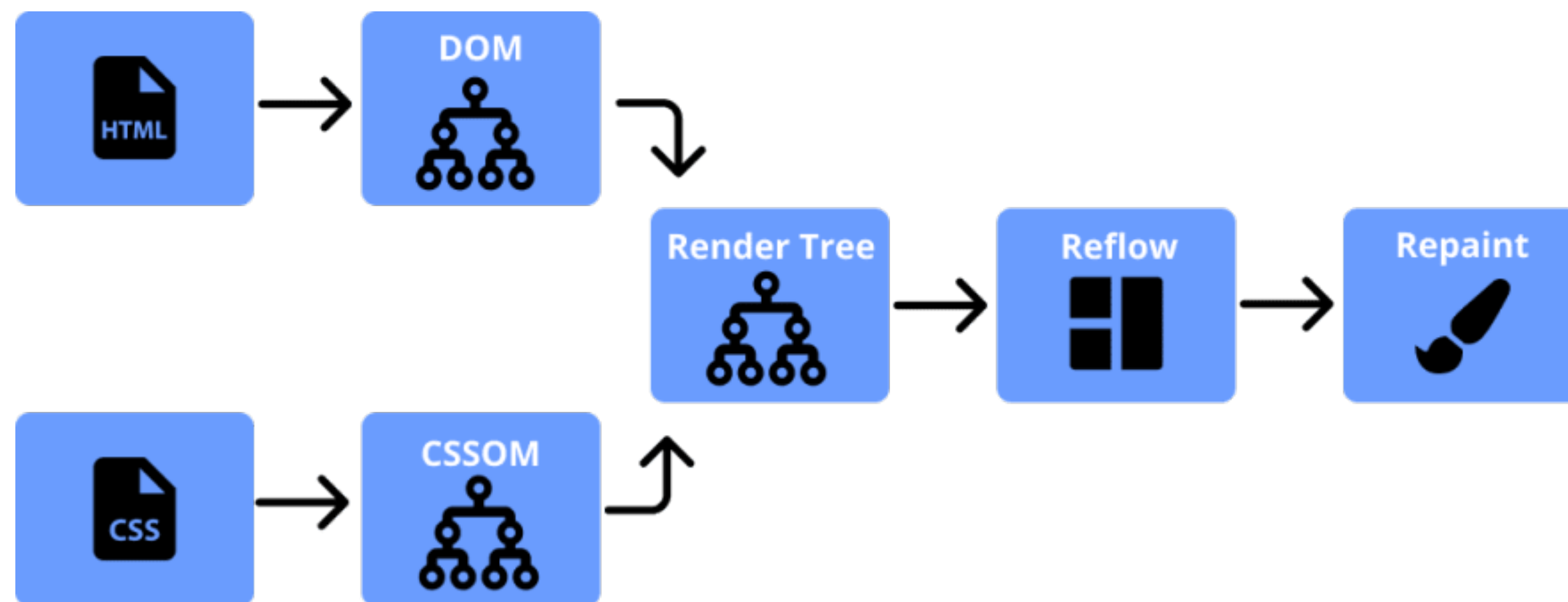




# Virtual DOM

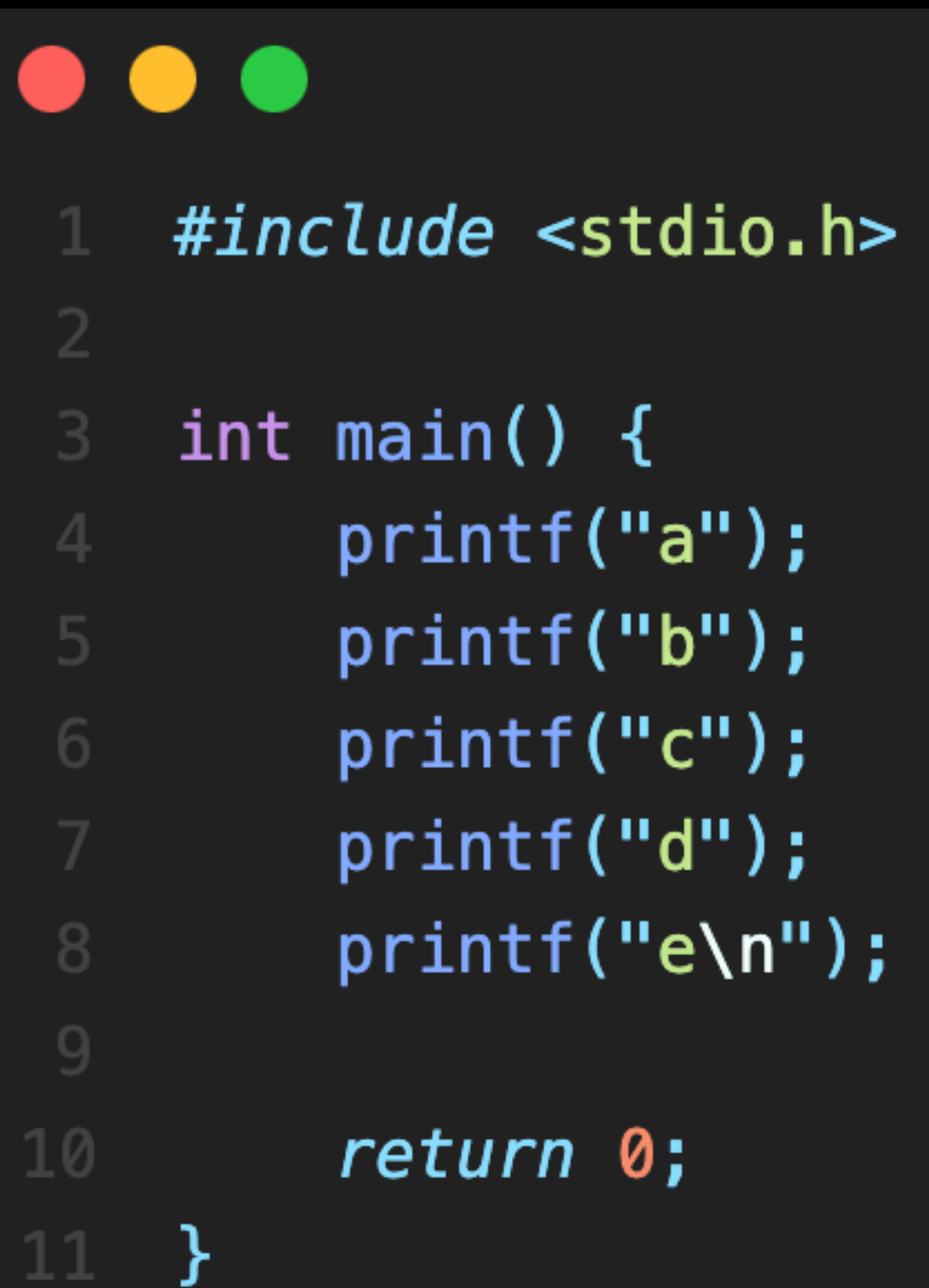


# Virtual DOM



```
1 const btn = document.getElementById("myButton");  
2 const width = btn.offsetWidth;
```

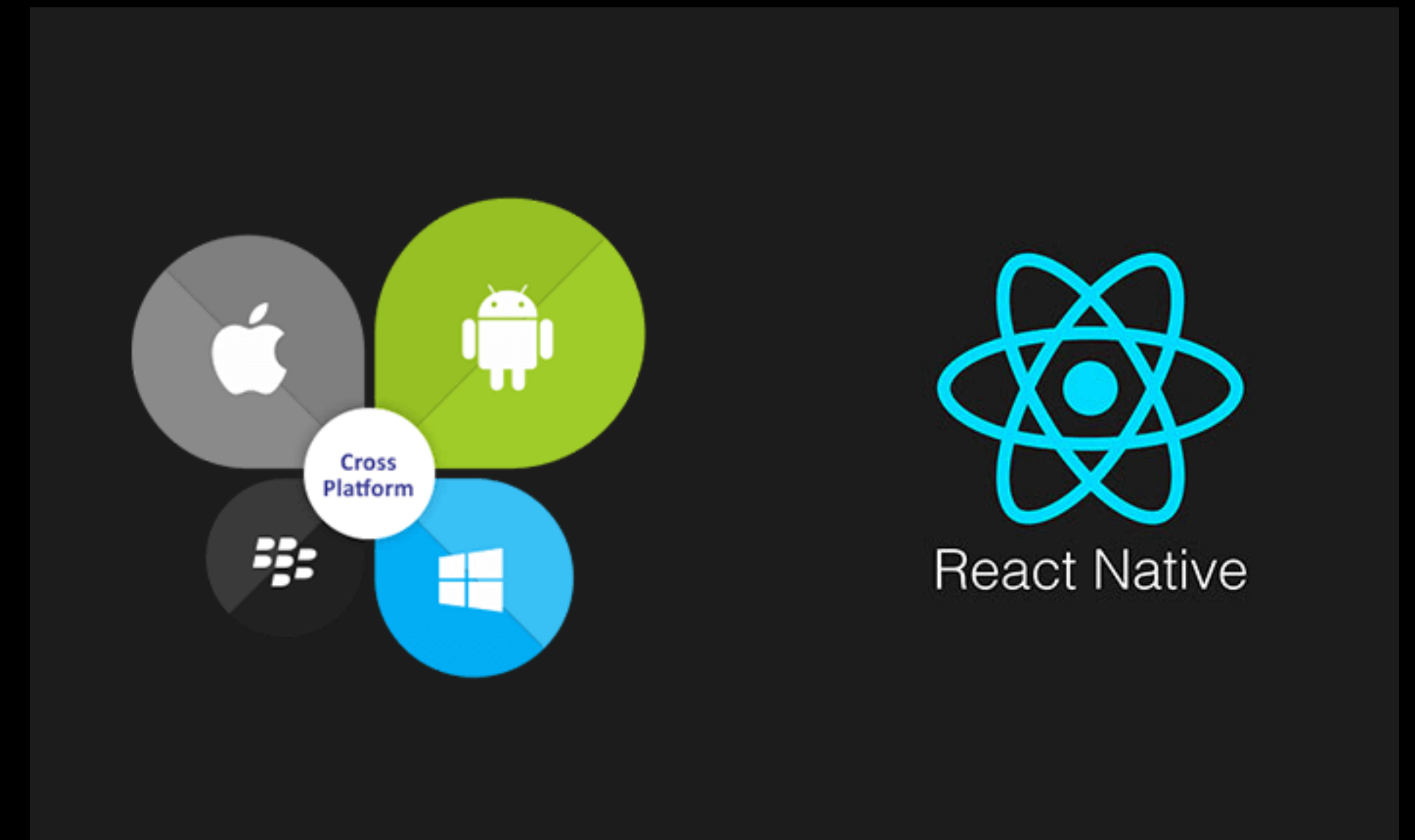
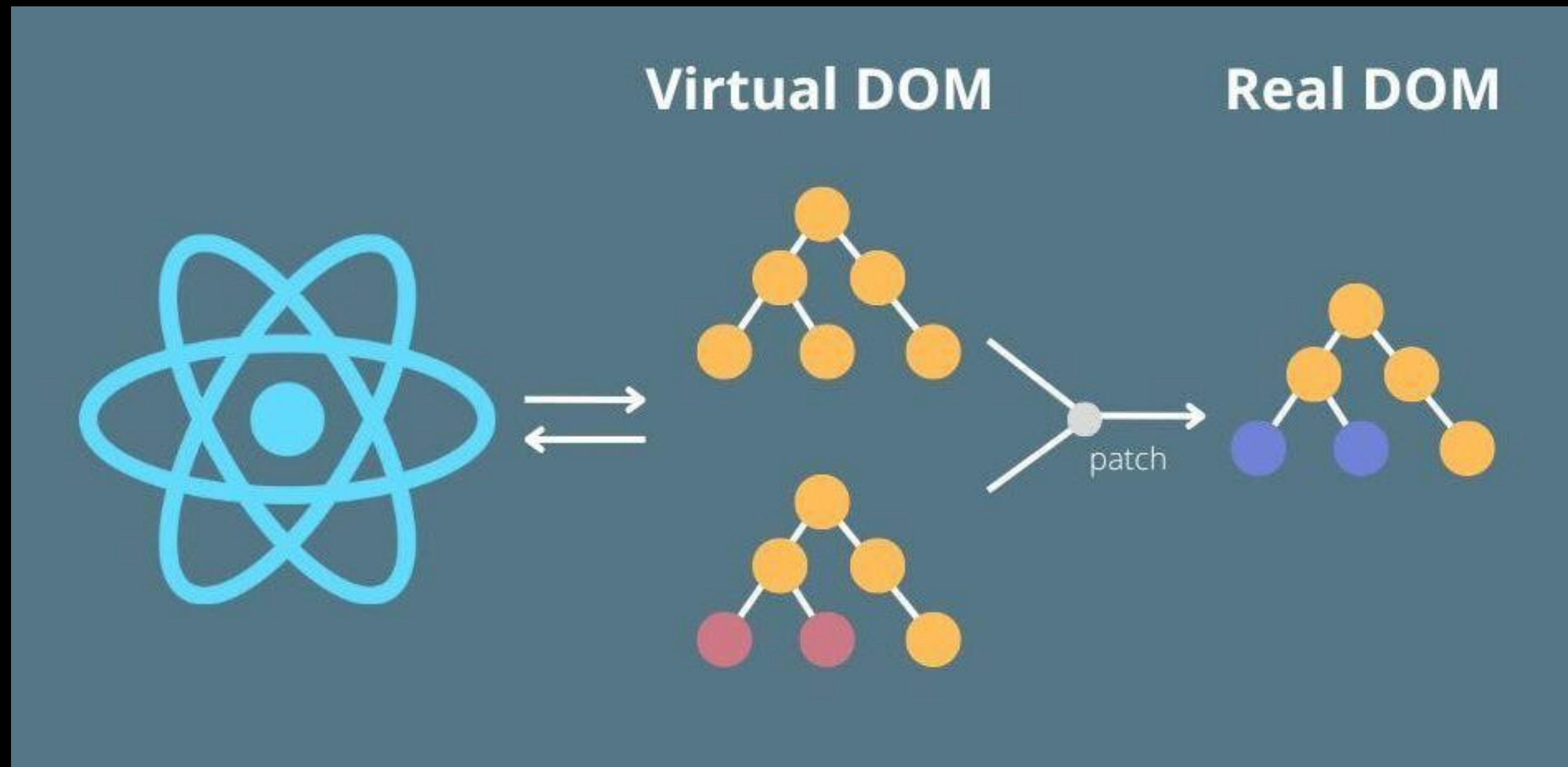
# Virtual DOM



```
1  #include <stdio.h>
2
3  int main() {
4      printf("a");
5      printf("b");
6      printf("c");
7      printf("d");
8      printf("e\n");
9
10     return 0;
11 }
```

```
abcde
Process exited with status 0
```

# Virtual DOM





# React Element



```
1 <div class="my-class">Hello World</div>
```

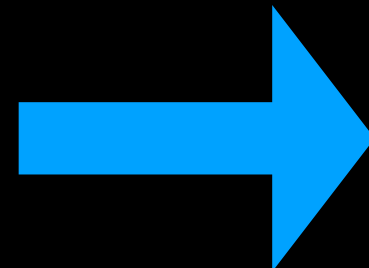
```
▼ Object i  
  $$typeof: Symbol(react.element)  
  key: null  
  ▶ props: {className: 'my-class', children: 'Hello World'}  
  ref: null  
  type: "div"  
  _owner: null  
  ▶ _store: {validated: false}  
  _self: null  
  _source: null  
  ▶ [[Prototype]]: Object
```

```
1 const divElement = React.createElement(  
2   "div",  
3   { className: "my-class" },  
4   "Hello World"  
5 );  
6  
7 const divElement = document.createElement("div");  
8 divElement.className = "my-class";  
9 divElement.textContent = "Hello World";
```

# React Element

```
1 import { useState } from "react";
2
3 function App() {
4   const [count, setCount] = useState(0);
5
6   return (
7     <div>
8       <h1>카운트 : {count}</h1>
9       <button onClick={() => setCount(count + 1)}>증가</button>
10    </div>
11  );
12 }
13
14 export default App;
```

BABEL



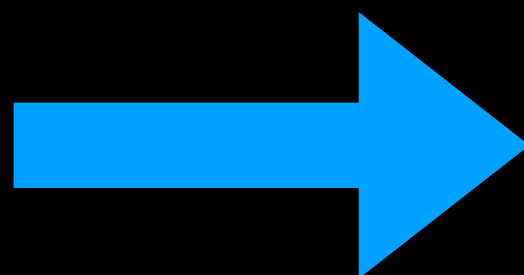
```
1 import React, { useState } from "react";
2
3 function App() {
4   const [count, setCount] = useState(0);
5
6   return React.createElement(
7     "div",
8     null,
9     React.createElement("h1", null, "카운트 : ", count),
10    React.createElement(
11      "button",
12      { onClick: () => setCount(count + 1) },
13      "증가"
14    )
15  );
16 }
17
18 export default App;
```

# Fiber

```
▼ Object ⓘ  
  $$typeof: Symbol(react.element)  
  key: null  
  ▶ props: {className: 'my-class', children: 'Hello World'}  
  ref: null  
  type: "div"  
  _owner: null  
  ▶ _store: {validated: false}  
  _self: null  
  _source: null  
  ▶ [[Prototype]]: Object
```

## React element

createFiberFromTypeAndProps()



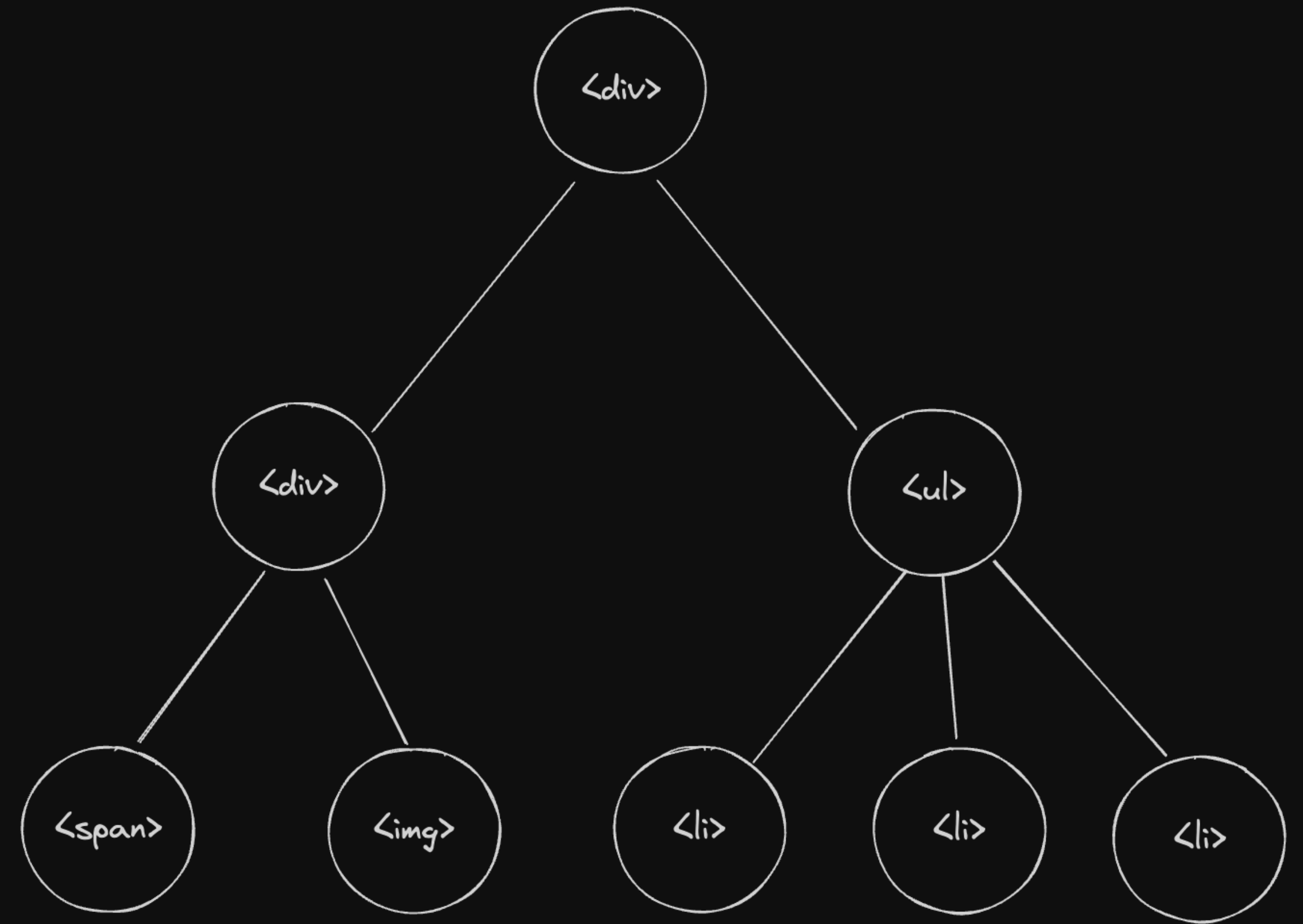
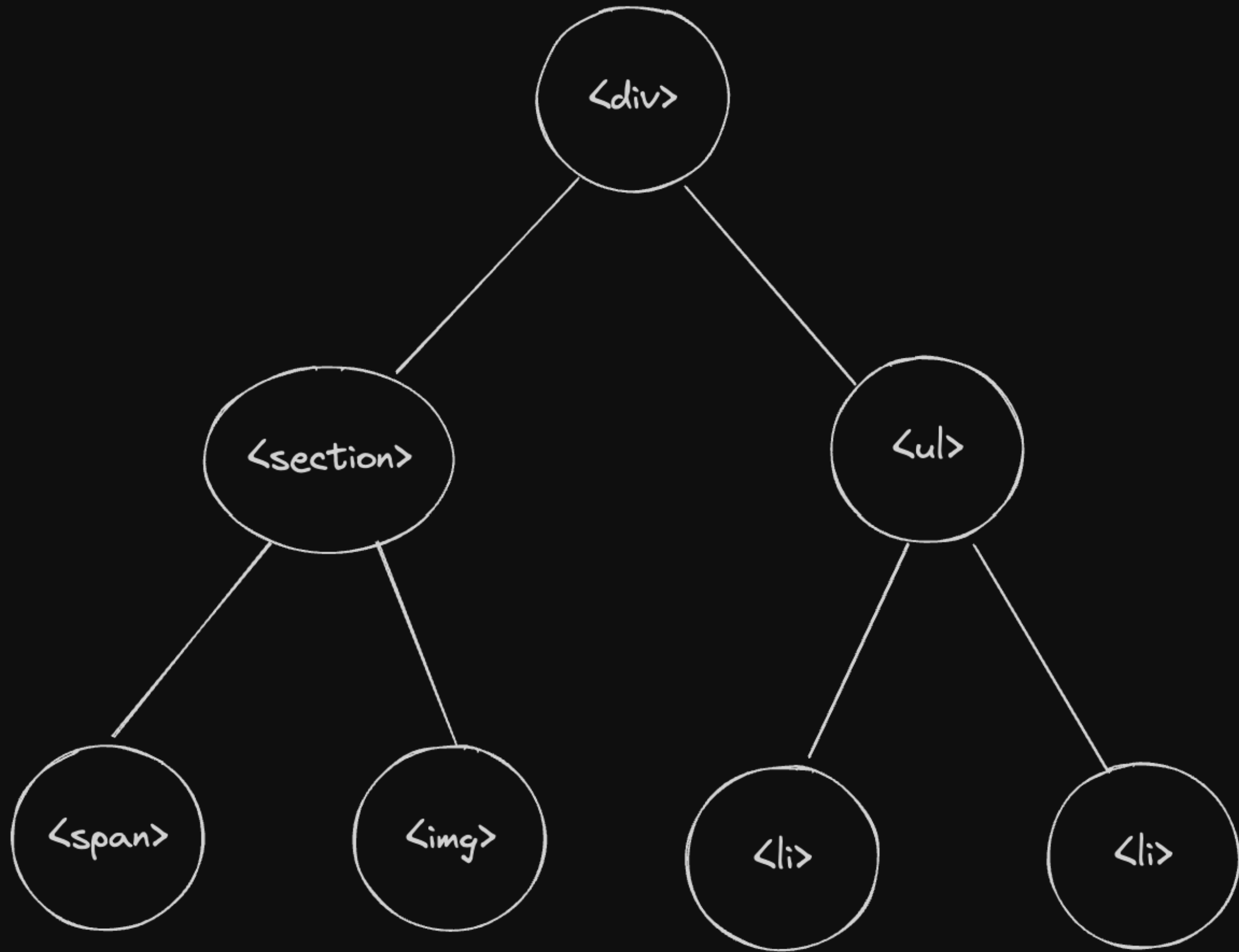
```
1  type Fiber = {  
2    tag: WorkTag,  
3    key: null | string,  
4    elementType: any,  
5    type: any,  
6    stateNode: any,  
7    return: Fiber | null,  
8    child: Fiber | null,  
9    sibling: Fiber | null,  
10   index: number,  
11   ref:  
12     | null  
13     | (((handle: any) => void) & { _stringRef: string | null })  
14     | RefObject,  
15   refCleanup: null | (() => void),  
16   ...  
17 }
```

## React fiber

# Fiber

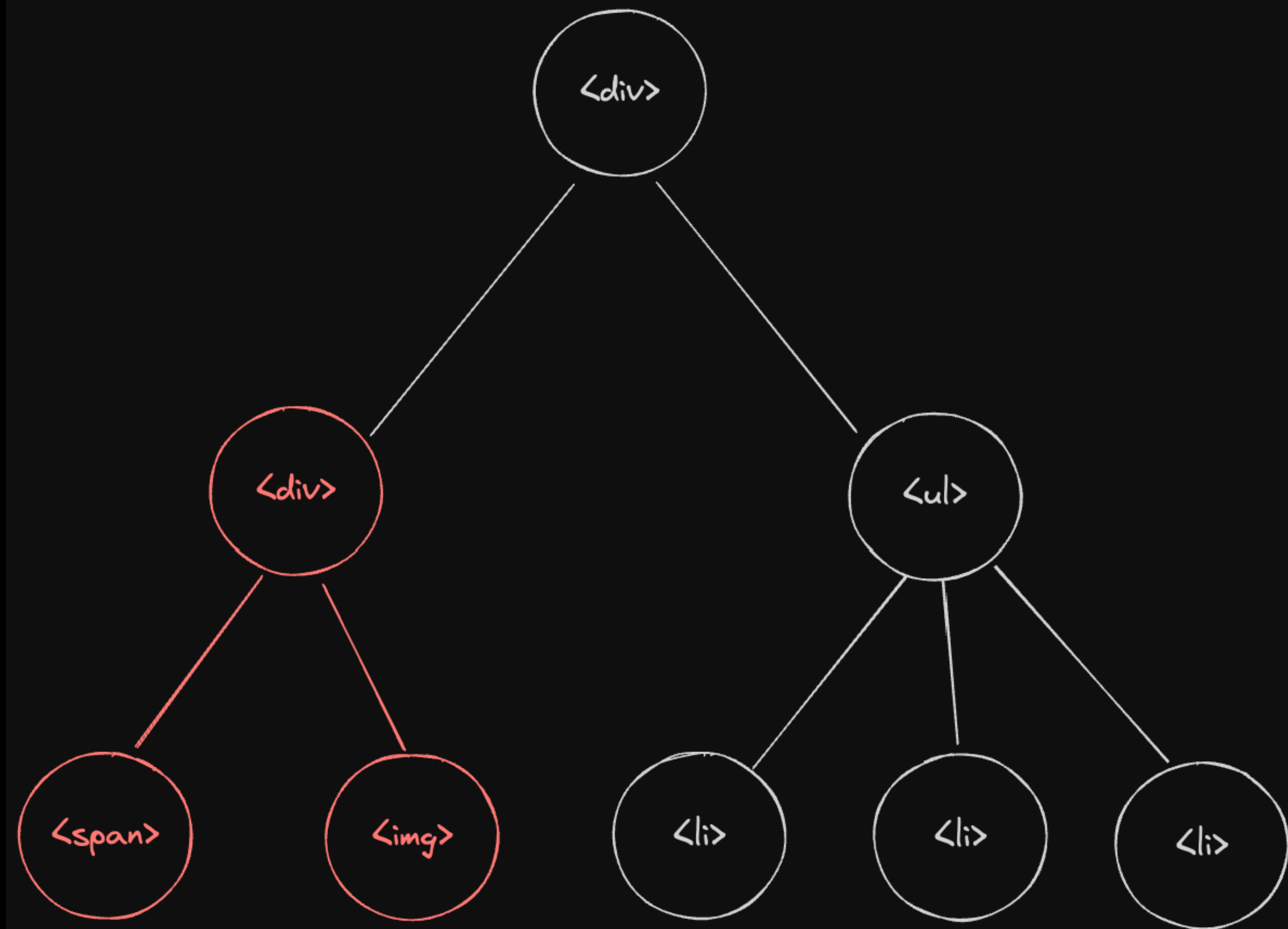
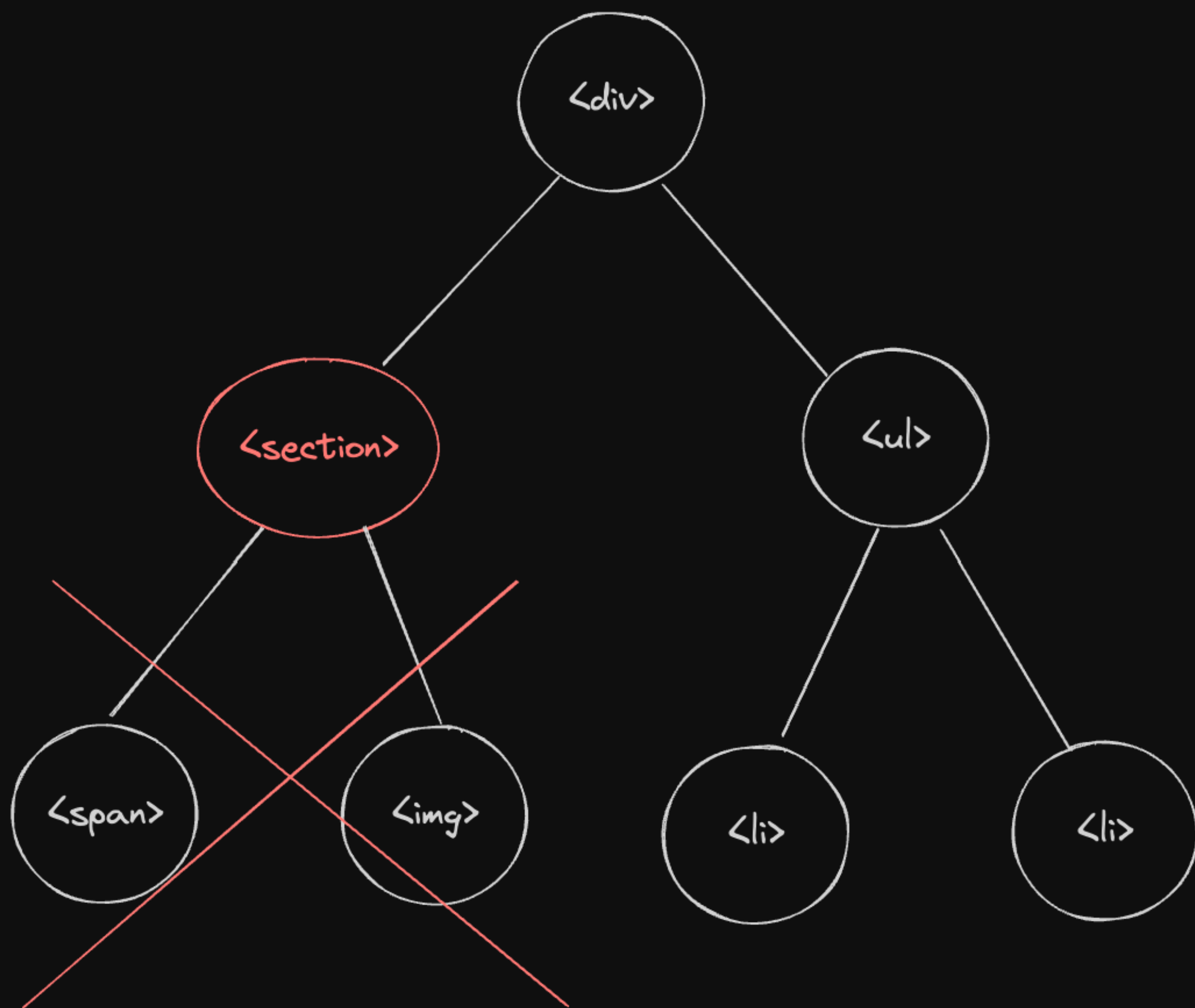


# Diff

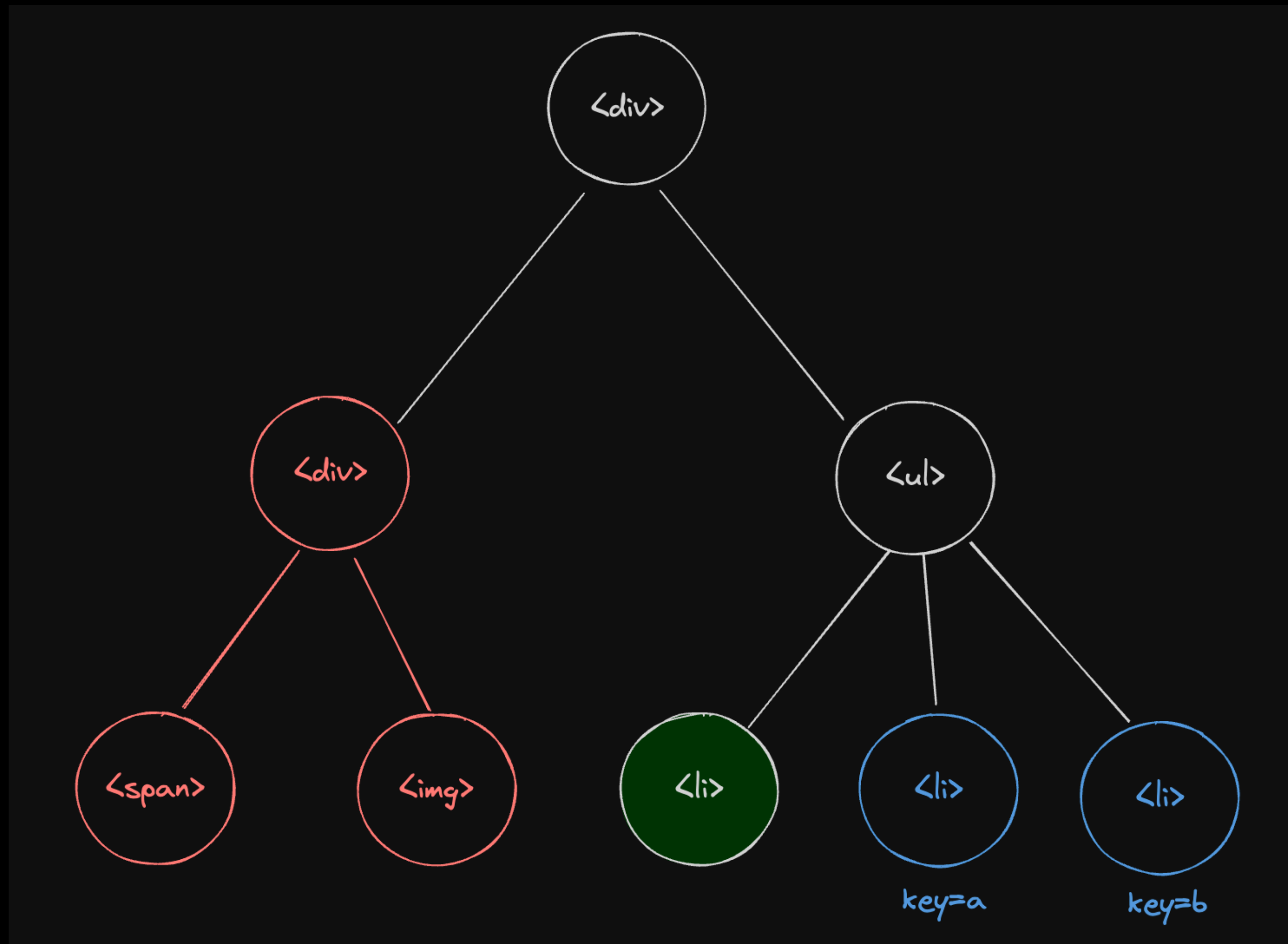
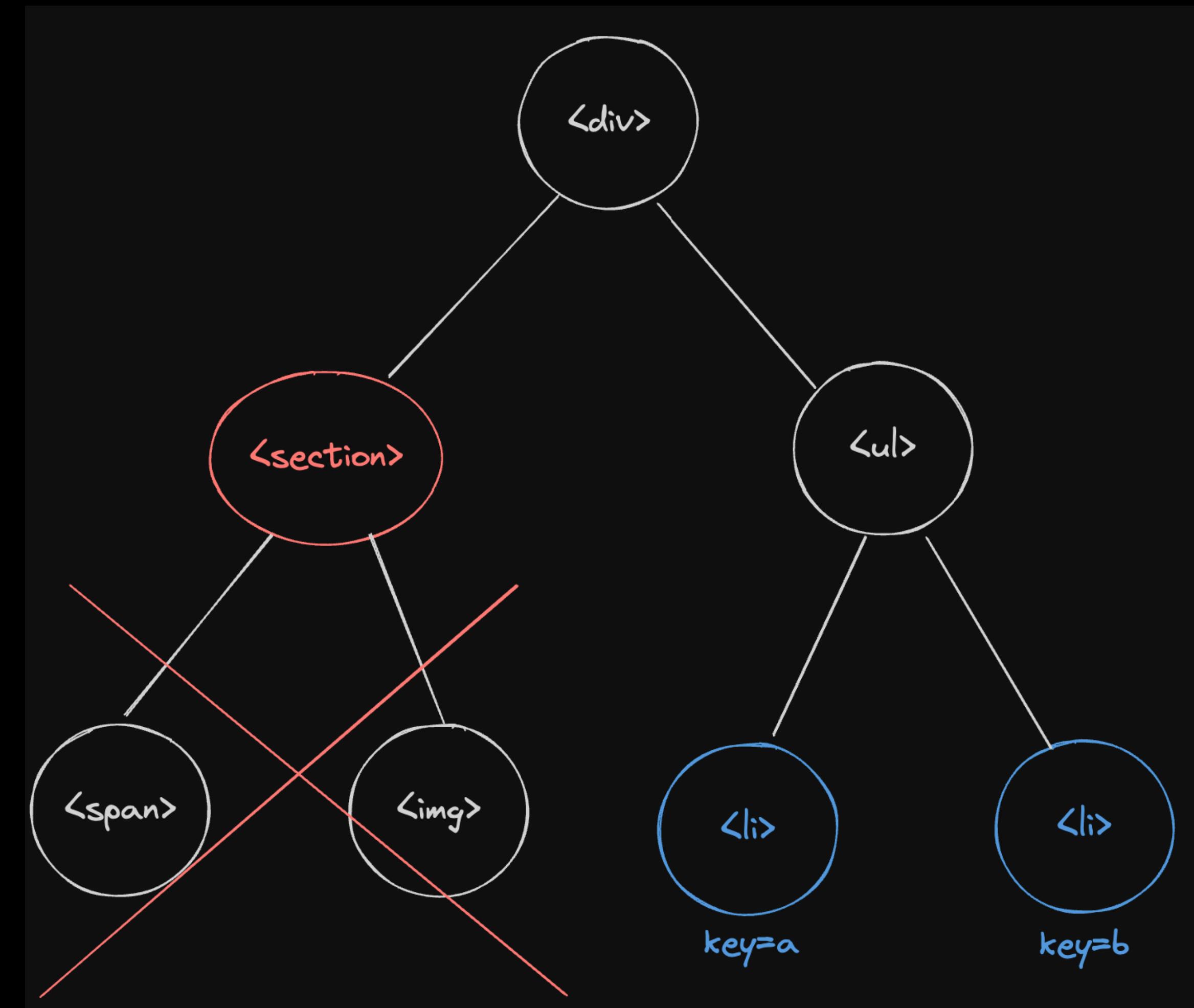




# Diff



# Diff



# Review

Virtual DOM

DOM

React element

Diff

Fiber

# Truth

Since "virtual DOM" is more of a pattern than a specific technology, people sometimes say it to mean different things. In React world, the term "virtual DOM" is usually associated with React elements since they are the objects representing the user interface. React, however, also uses internal objects called "fibers" to hold additional information about the component tree. They may also be considered a part of "virtual DOM" implementation in React.

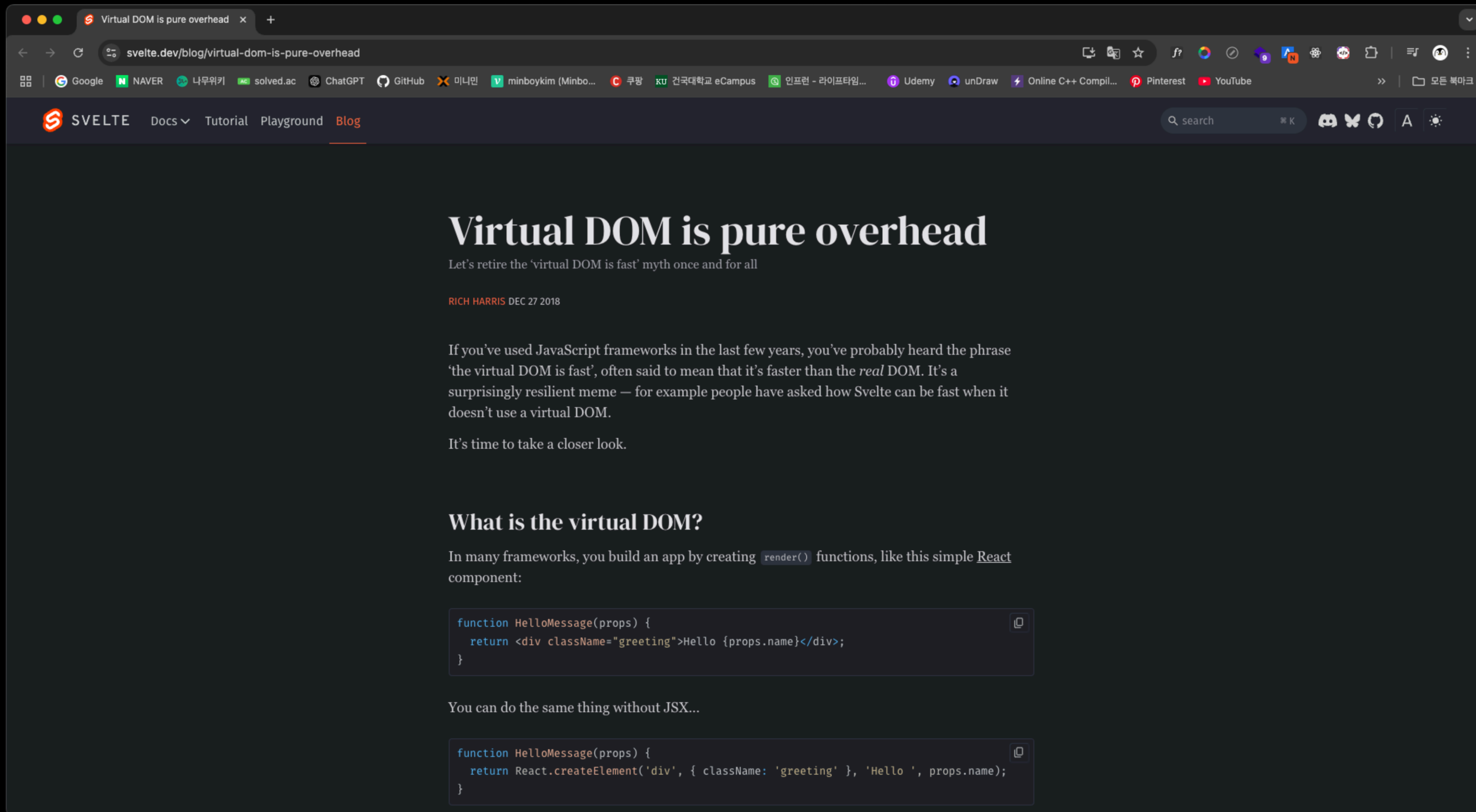


**Dan Abramov**

74K Followers

Working on @reactjs. Co-author of Redux and Create React App. Building tools for humans.

# Truth





**Thank you**

# QnA & Discussion

# Retrospect

NICKNAME.MD

in 01\_20\_VirtualDOM&Reconciliation/Retrospect