Introduction to the Document Object Model

You Will Be Able To…

- Explain what the DOM is
- Explain why the DOM is important for front-end web developers
- Name and use methods to *search* the DOM
- Name and use methods to *traverse* the DOM
- Manipulate the DOM

How does a browser render a webpage?

- **Step One - The browser makes a request to a server**
  - The request is typically triggered by the URL bar, or by clicking a link
  - The server responds by sending back HTML to the browser

- **Step Two - ‘deserialize’ HTML (text) into an data structure of connected objects.**

- **Step Three - Use these connected objects, which have dynamic properties, to ‘paint’ a vizualization**

How does a browser render a webpage?

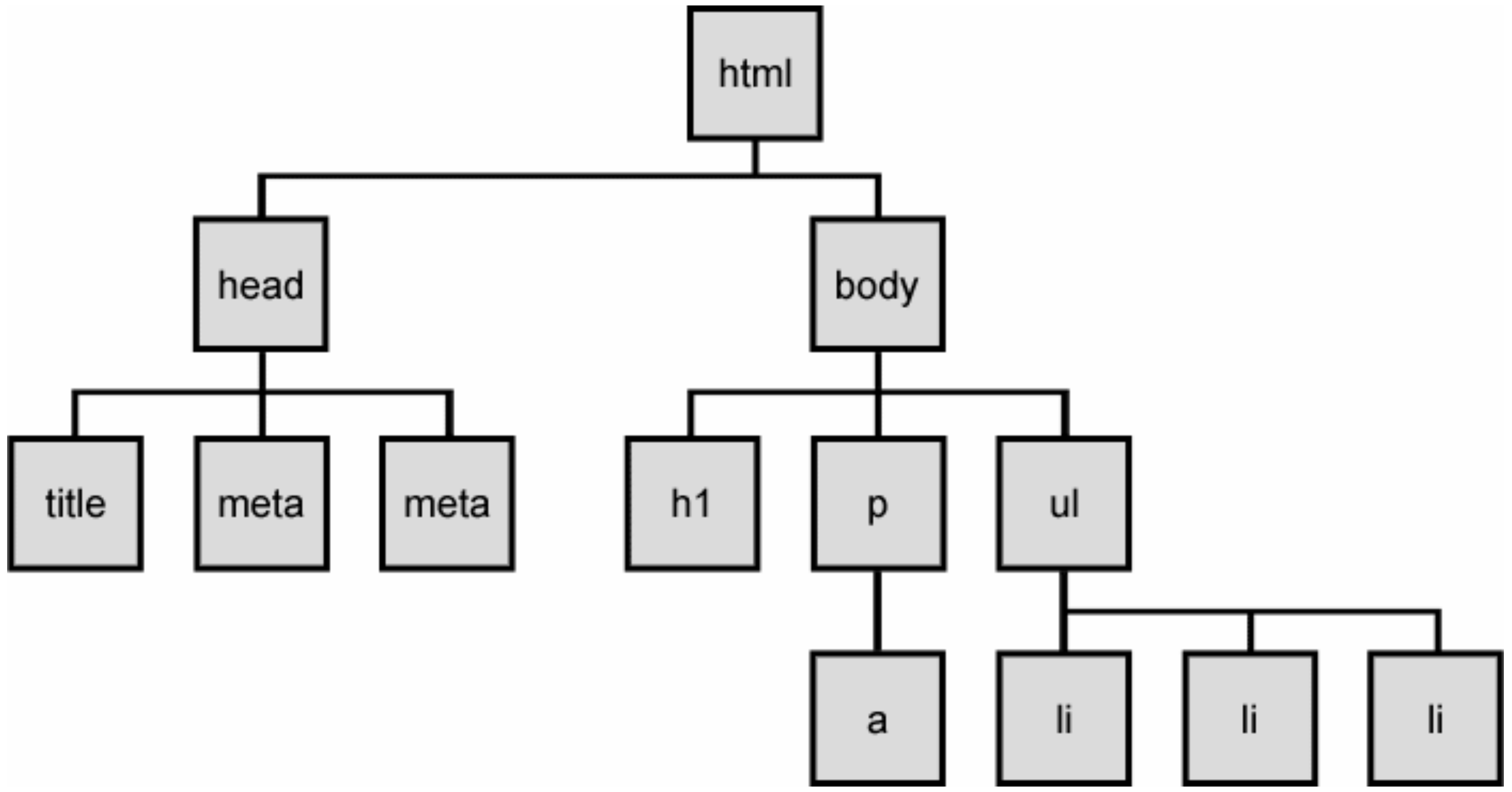
- Step One - The browser makes a request to a server
    - The request is typically triggered by the URL bar, or by clicking a link
    - The server responds by sending back HTML to the browser

- Step Two - ‘deserialize’ HTML (text) into a data structure of connected objects.

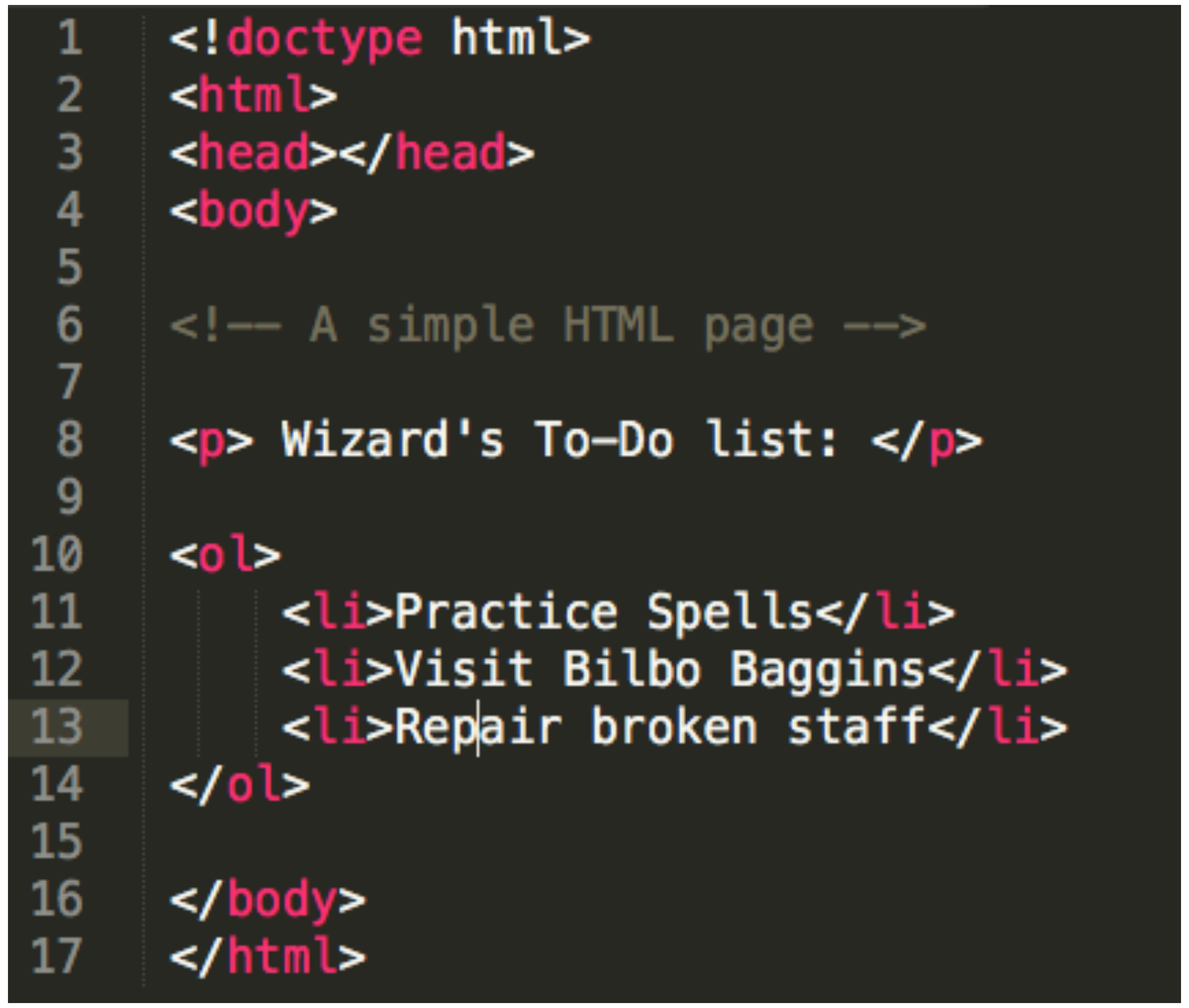
- Step Three - Use these connected objects, which have dynamic properties, to ‘paint’ a vizualization

The DOM is a tree

- Trees are a ubiquitous data structure
- Every DOM element is a node
- There are nodes that branch into other nodes



The DOM is a tree



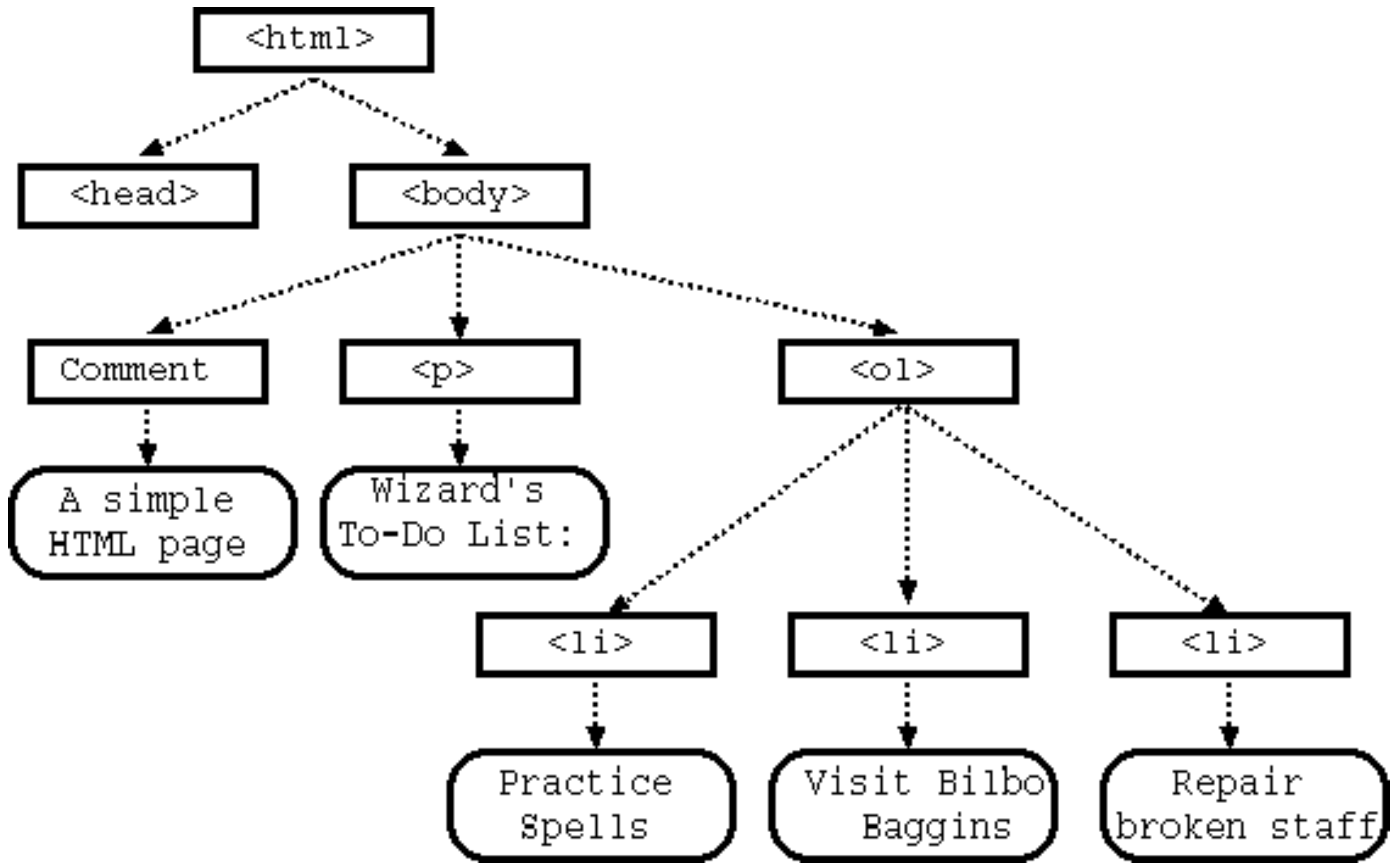
```html
<!doctype html>
<html>
<head></head>
<body>

<!-- A simple HTML page -->

<p> Wizard's To-Do list: </p>

<ol>
    <li>Practice Spells</li>
    <li>Visit Bilbo Baggins</li>
    <li>Repair broken staff</li>
</ol>

</body>
</html>
```



http://www.codingtree.com/javascript/javascript-DOM-introduction.html

Shhh... Do You Want to See a *real* DOM?

Why study the DOM?

- **The Document Object Model is:**
  - The browser's 'internal representation' of the webpage
  - What allows web pages to render, respond to user events and change
  - **It effectively 'connects JavaScript to HTML' ...kind of**

How does a browser render a webpage?

- **Step One - The browser makes a request to a server**
  - The request is typically triggered by the URL bar, or by clicking a link
  - The server responds by sending back HTML to the browser

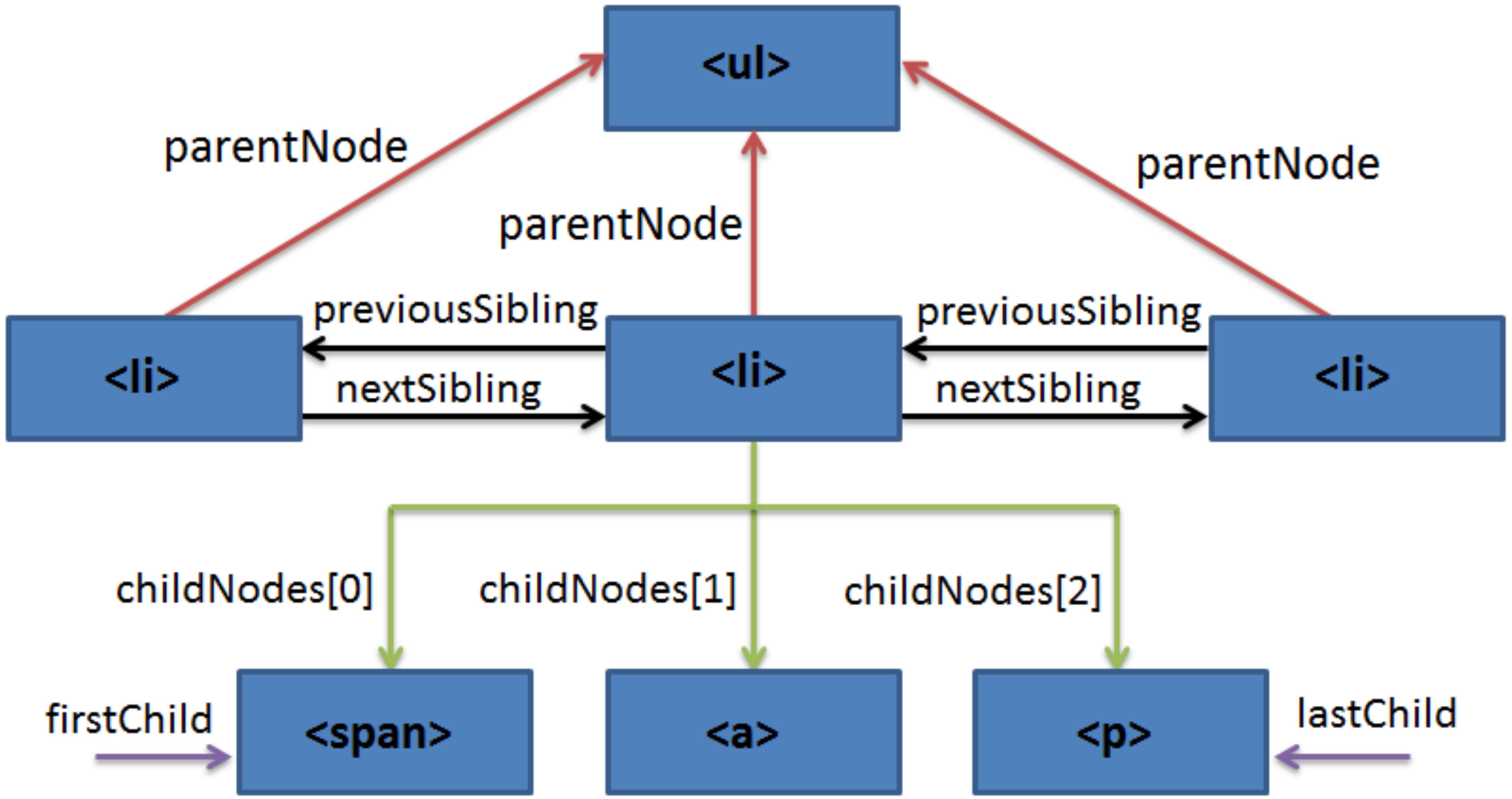
- **Step Two - ‘deserialize’ HTML (text) into an data structure of connected objects.**

- **Step Three - Use these connected objects, which have dynamic properties, to ‘paint’ a vizualization**

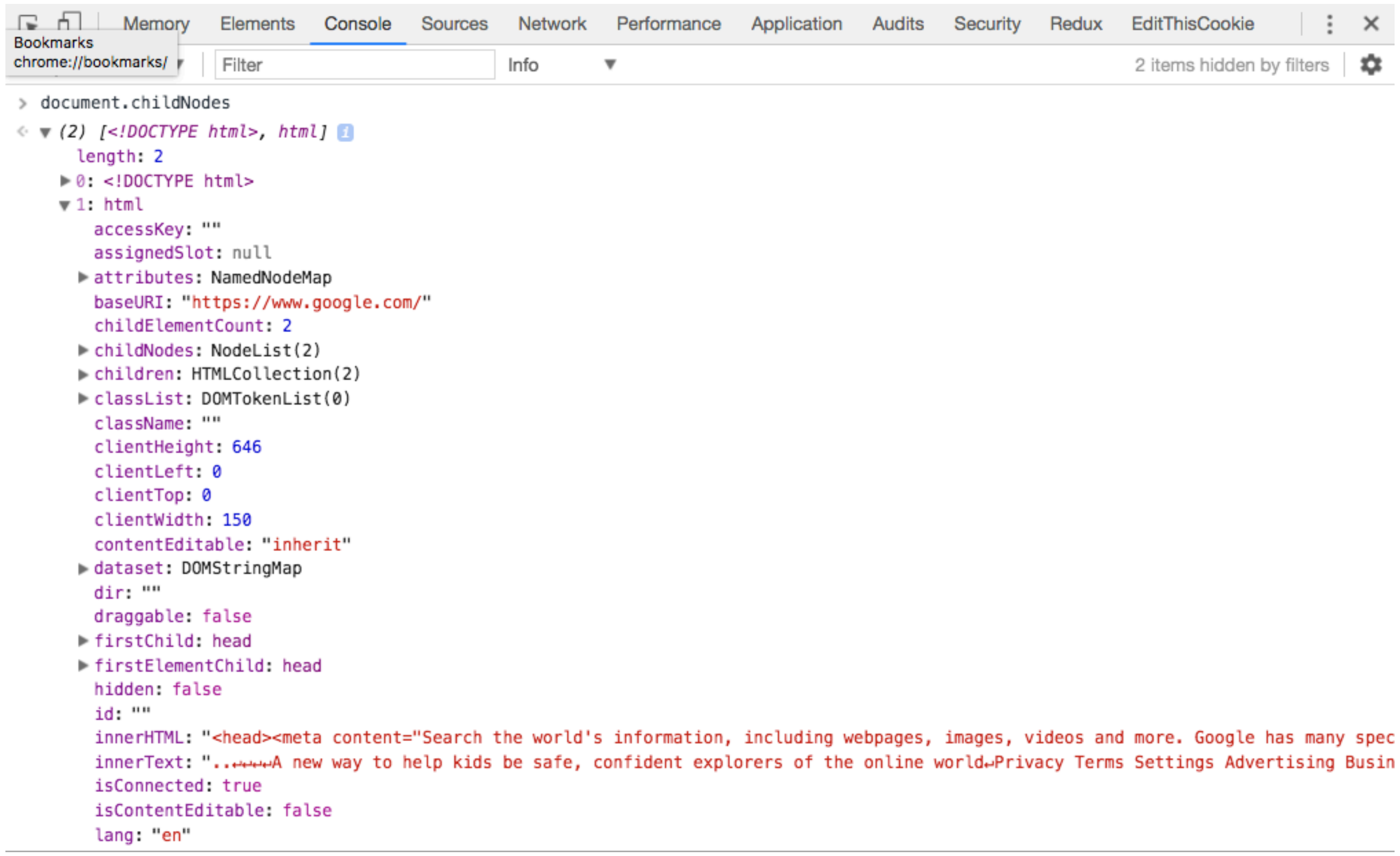
How do I get access to the DOM?!

- **We use the 'document' object**
  - This is a big object that contains methods for *navigating* the DOM and *manipulating* the DOM. It is the root of the DOM.
- **This 'document' object is the bridge between the DOM and our javascript**
- **It is easy to navigate through:**
  - **At any point in the DOM you are at a node**
  - **All nodes share similar navigation methods**

DOM navigation



Nodes have lots of attributes



```
> document.childNodes
<· ▼ (2) [<!DOCTYPE html>, html]
      length: 2
    ▶ 0: <!DOCTYPE html>
    ▼ 1: html
        accessKey: ""
        assignedSlot: null
      ▶ attributes: NamedNodeMap
        baseURI: "https://www.google.com/"
        childElementCount: 2
      ▶ childNodes: NodeList(2)
      ▶ children: HTMLCollection(2)
      ▶ classList: DOMTokenList(0)
        className: ""
        clientHeight: 646
        clientLeft: 0
        clientTop: 0
        clientWidth: 150
        contentEditable: "inherit"
      ▶ dataset: DOMStringMap
        dir: ""
        draggable: false
      ▶ firstChild: head
      ▶ firstElementChild: head
        hidden: false
        id: ""
        innerHTML: "<head><meta content="Search the world's information, including webpages, images, videos and more. Google has many spec
        innerText: "..↵↵↵↵A new way to help kids be safe, confident explorers of the online world↵Privacy Terms Settings Advertising Busin
        isConnected: true
        isContentEditable: false
        lang: "en"
```

Searching the DOM (the easy way)

- Searching the DOM using methods on the document object
  - *getElementById* finds nodes with a certain ID
    - document.getElementById('myId') // returns a single node
  - *getElementsByClassName* finds all nodes with a certain class
    - document.getElementsByClassName('someClassName') // returns a collection of nodes
  - *getElementsByTagName* finds all nodes with a certain HTML tag
    - document.getElementsByTagName('div') // returns a collection of nodes

…traversing the DOM

- **moving around the DOM using methods that each node has**
  - *node.children // returns all childNodes that are HTML elements*
  - *node.nextElementSibling, node.nextElementSibling // returns next or previous Sibling that is an HTML element*
  - *node.parentElement // returns parent element if it is an HTML element*

…then manipulating the DOM

- innerHTML
- Changing Attributes for Style
  - User Agent Stylesheet
  - Paint and Render Cycles
- Adding event handlers
- Making Elements
- Putting them into the DOM
- Remove Elements

Searching the DOM (the even-easier way)

- document.querySelector('.something')
- document.querySelectorAll('.something')

You Will Be Able To…

- Explain what the DOM is
- Explain why the DOM is important for front-end web developers
- Name and use methods to *search* the DOM
- Name and use methods to *traverse* the DOM
- Manipulate the DOM

Workshop

- You will be recreating *querySelectorAll()* and calling it $
- We will be doing it in 3 parts/functions:
    - 1 - A function that identifies the selector the user is passing into $ (are they searching for an id, class, tag, or tag with class?)
    - 2 - A function uses function number 1 to check which type of selector is being searched for. It then returns a new function which tests if a given element matches the specific selector a user wants
    - 3 - A function that traverses all nodes in the DOM and applies the function above (number 2) to it.