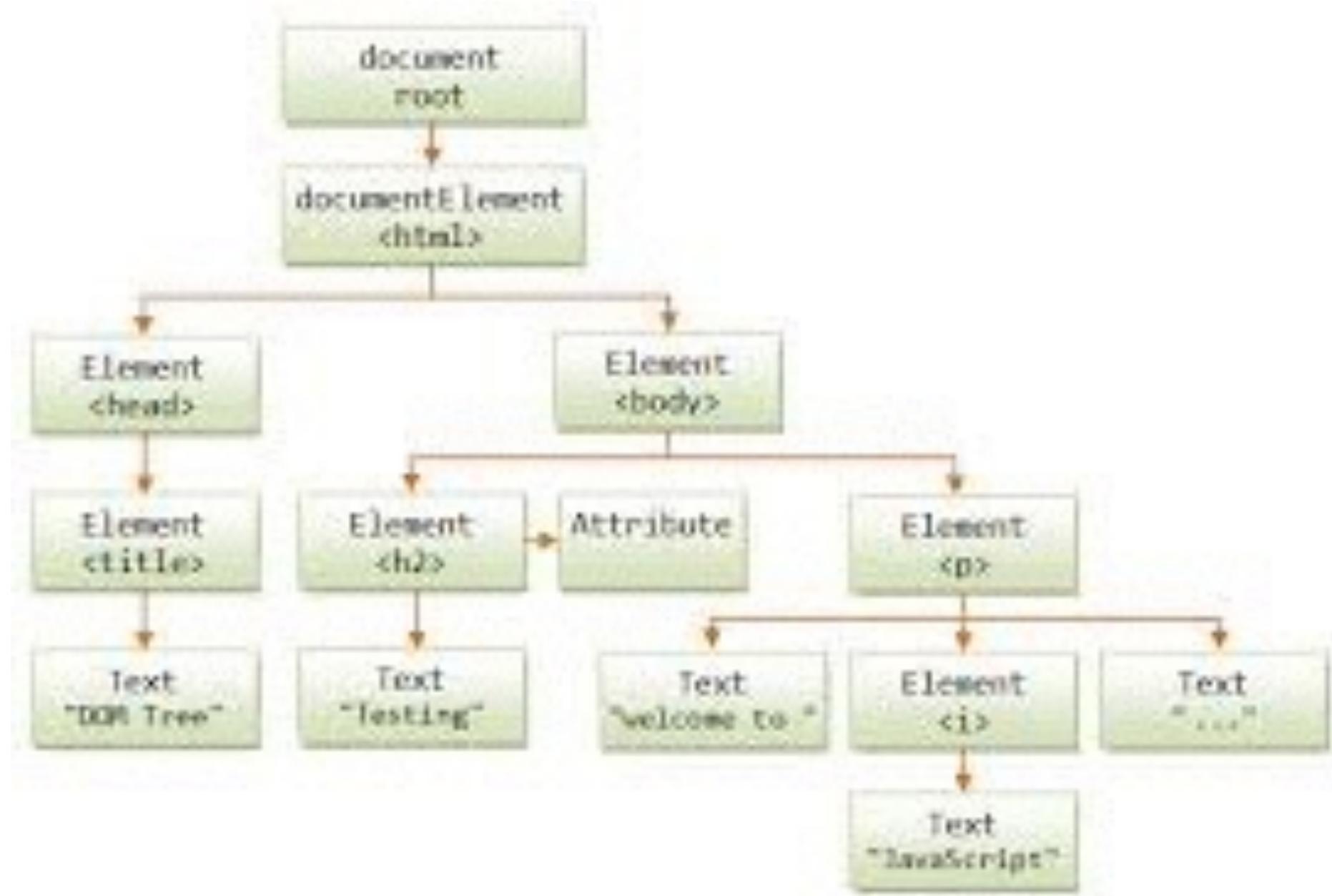


Introduction to the Document Object Model

Agenda



- Why Bother?
- Brief History
- Analogies
- Visualizations
- Searching
- Traversing
- Tools

Why study the DOM?

- The Document Object Model is:

- The most powerful publishing platform ever created
- What allows web pages to render, respond to user events and change
- Connects JavaScript to HTML

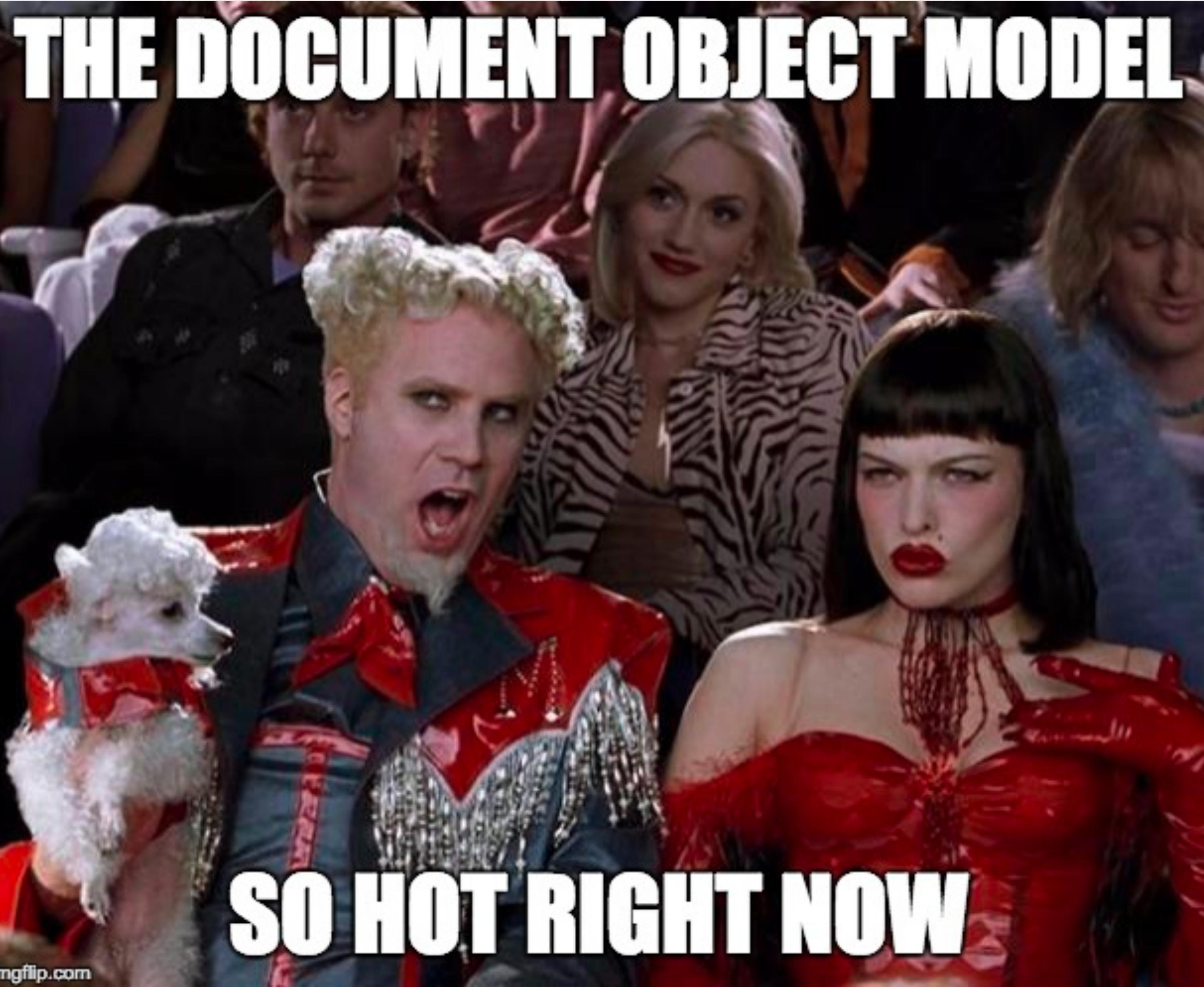
HTML

```
index.html
<html>
  <head>
    <title>...</
  title>
  </head>
  <body>
    <h4>...</h4>
    <div>
      <p>...</p>
      <p>...</p>
    </div>
  </body>
</html>
```

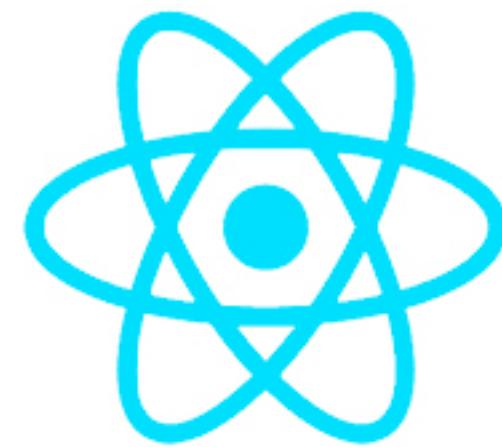
JS

```
(console)
> document
< {
  ...
  head: {
    ...
  },
  ...
  body: {
    ...
  }
  ...
}
```

THE DOCUMENT OBJECT MODEL



The DOM is Hot



React



polymer



The History of the DOM

- The original World Wide Web was a simple idea: document retrieval through hyperlinks between documents

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 news](#) , [Frequently Asked Questions](#) .

[What's out there?](#)

Pointers to the world's online information, [subjects](#) , [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#) ,X11 [Viola](#) ,[NeXTStep](#) ,[Servers](#) ,[Tools](#) ,[Mail robot](#) ,[Library](#))

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help ?](#)

If you would like to support the web..

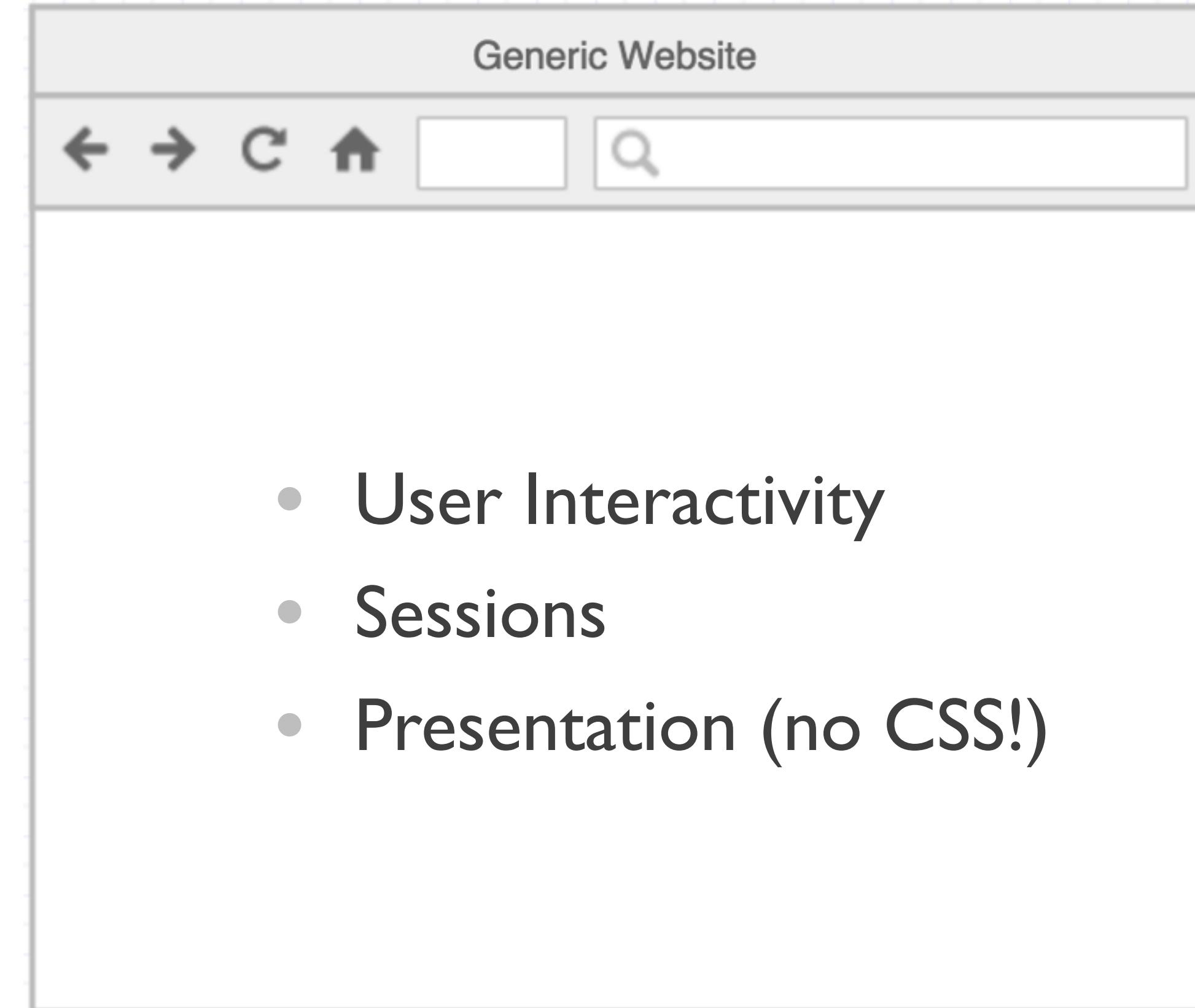
[Getting code](#)

Getting the code by [anonymous FTP](#) ,etc.

HTML

```
index.html  
  
<html>  
  <head>  
    <title>...</title>  
  </head>  
  <body>  
    <h4>...</h4>  
    <div>  
      <p>...</p>  
      <p>...</p>  
    </div>  
  </body>  
</html>
```

Web Page



A technically correct definition of the DOM

The Document Object Model (DOM) is a **cross-platform** and **language-independent** convention for representing and interacting with objects in **HTML, XHTML, and XML documents**.

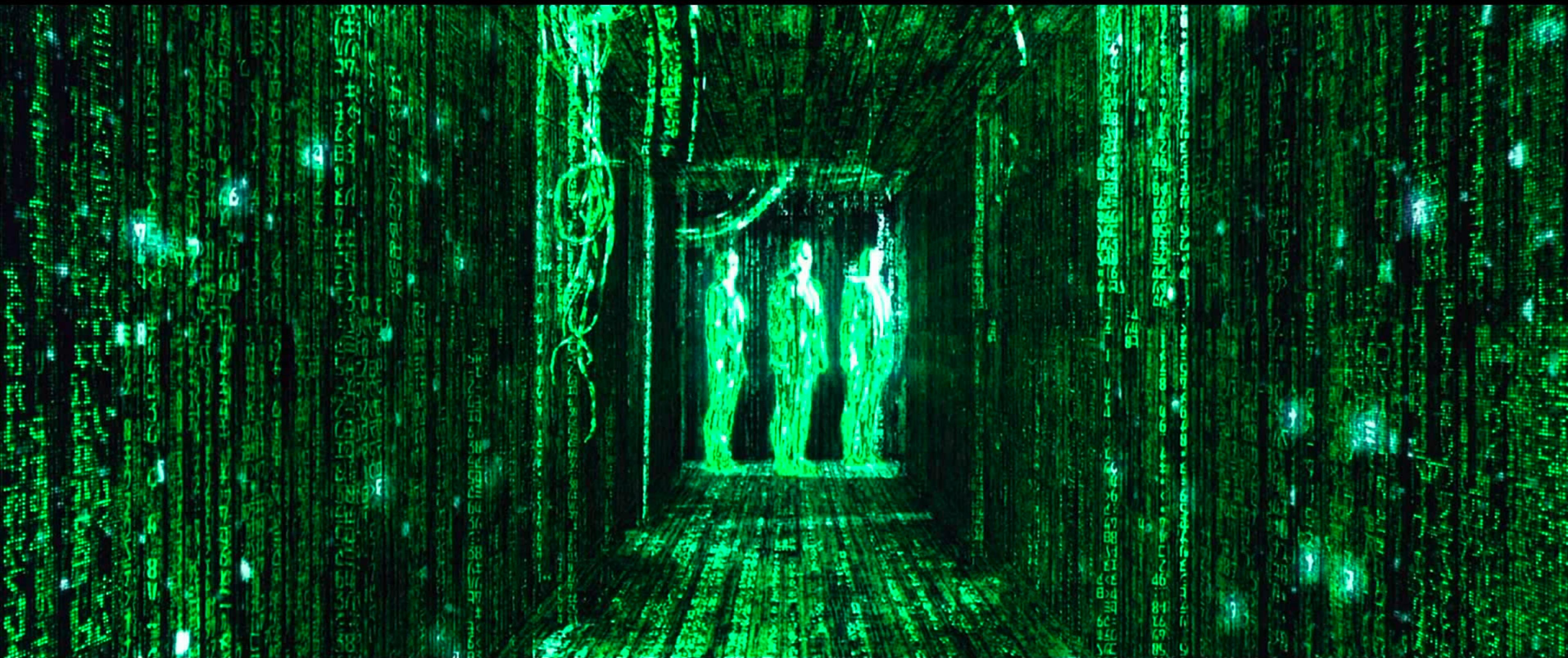
The **nodes** of every **document** are organized in a **tree structure**, called the **DOM tree**. **Objects** in the DOM tree may be addressed and manipulated by using methods on the objects. The public interface of a DOM is specified in its application programming interface (API).



Analogies

- **The Matrix**
- **Microsoft Word**
- **Serialization and JSON**





The screenshot shows a Microsoft Word document titled "Document1" with the following content:

```
<h1>My first website!
<h2>Hello world!
<p>I learned how to make websites at Fullstack and am
using the awesome web editor known as Microsoft
Word!
```

Annotations with arrows point from the text to the corresponding HTML tags:

- An arrow points from the text "My first website!" to the opening tag "<h1>".
- An arrow points from the text "Hello world!" to the opening tag "<h2>".
- An arrow points from the text "I learned how to make websites at Fullstack and am
using the awesome web editor known as Microsoft
Word!" to the opening tag "<p>".
- An arrow points from the image of a man holding a baby to the closing tag "".

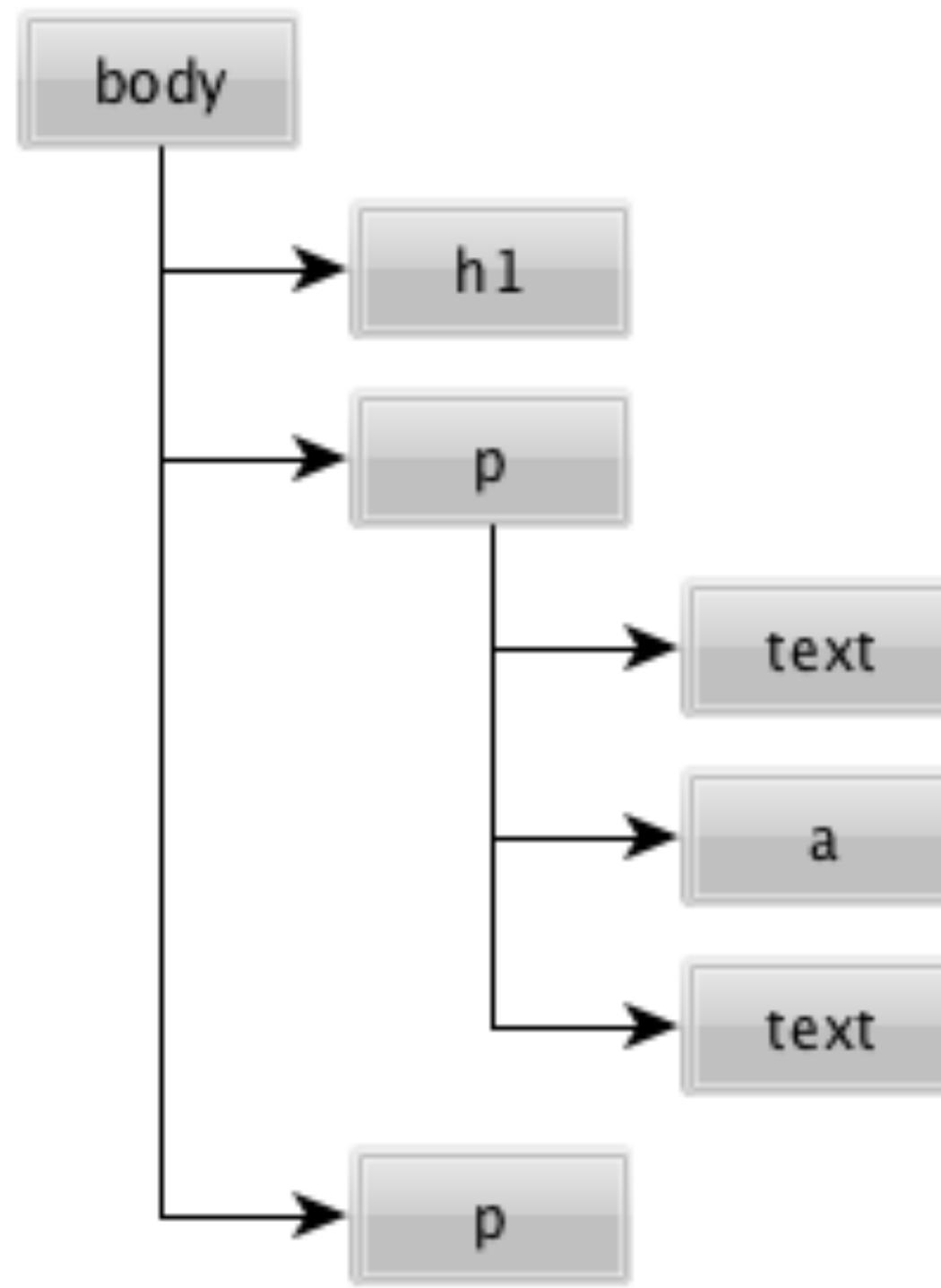
Serialization

- In programming we often have a "thing" and a way to represent that "thing"
- We refer to that representation as the **Serialization**
- Human Example: Thoughts into words
- HTML: the **serialized version of the DOM**
- The **DOM** is the "deserialized" version of HTML
- **JSON** - serialized object!

Visualizing the DOM

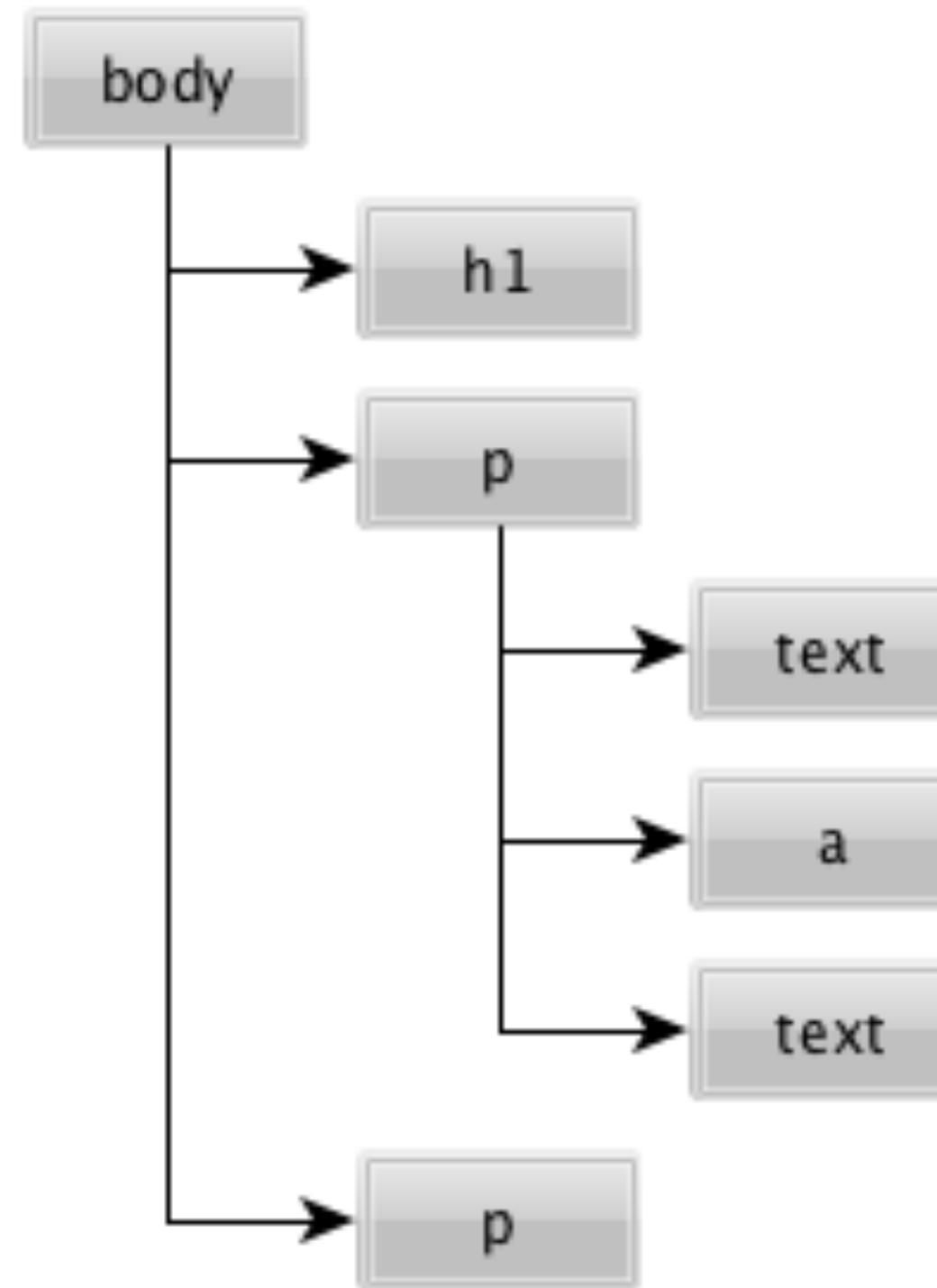
The DOM is a Tree

- Trees are a data structure from computer science
- The main idea here: There is a Node that branches into other Nodes (its children Nodes)
 - Each Node can have 0 to many children Nodes
 - Nodes can have 0 or 1 parent
 - Nodes can have 0 to many Sibling Nodes



The DOM is a Tree

```
<body>  
  
  <h1>Hello</h1>  
  <p>  
    Check out my  
    <a href="/page">Page!</a>  
    It's the best page out there  
  </p>  
  <p>Come back soon!</p>  
  
</body>
```



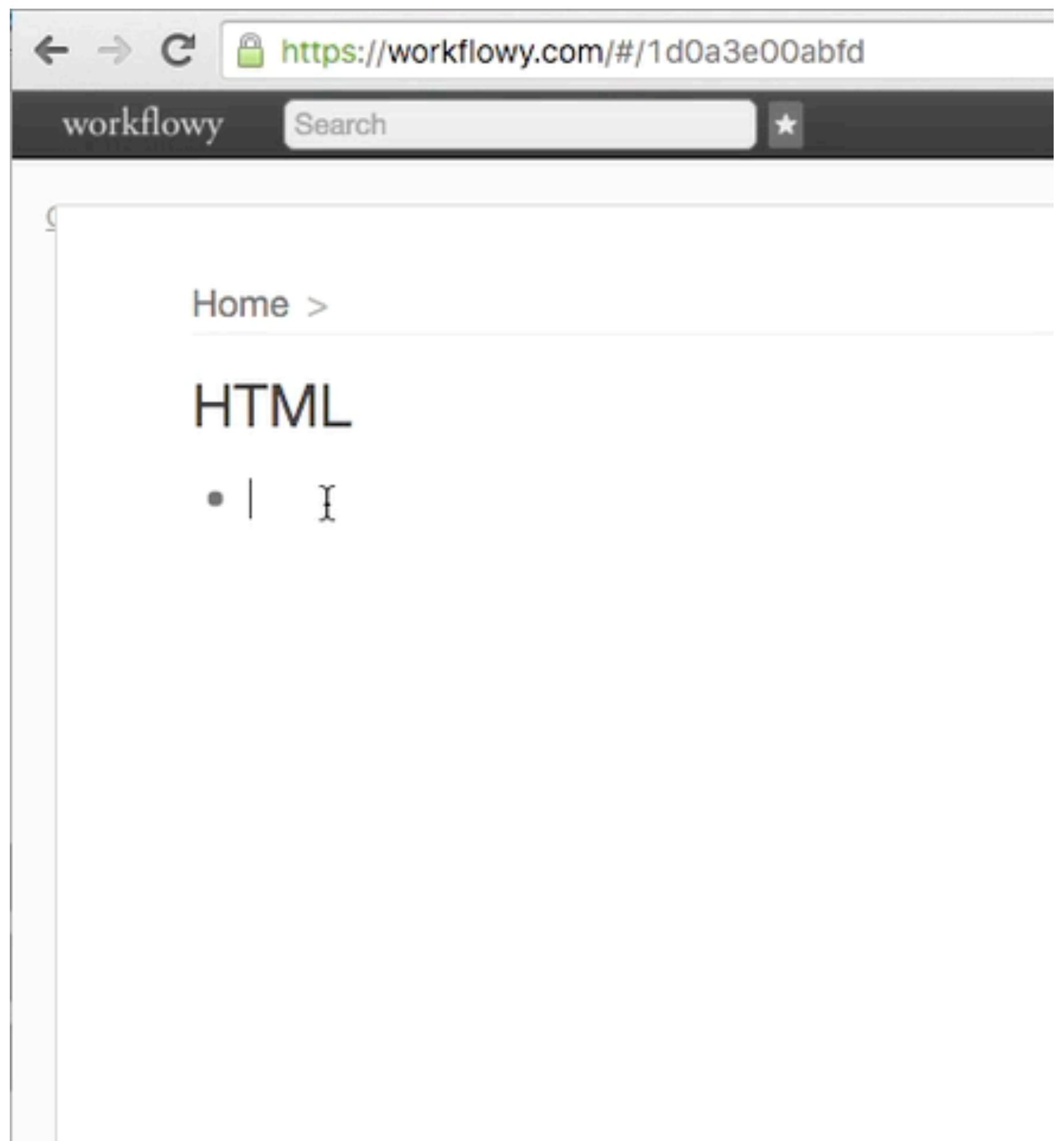
Indentation Is Important!

- No indentation in your HTML file makes it hard to see the tree structure:

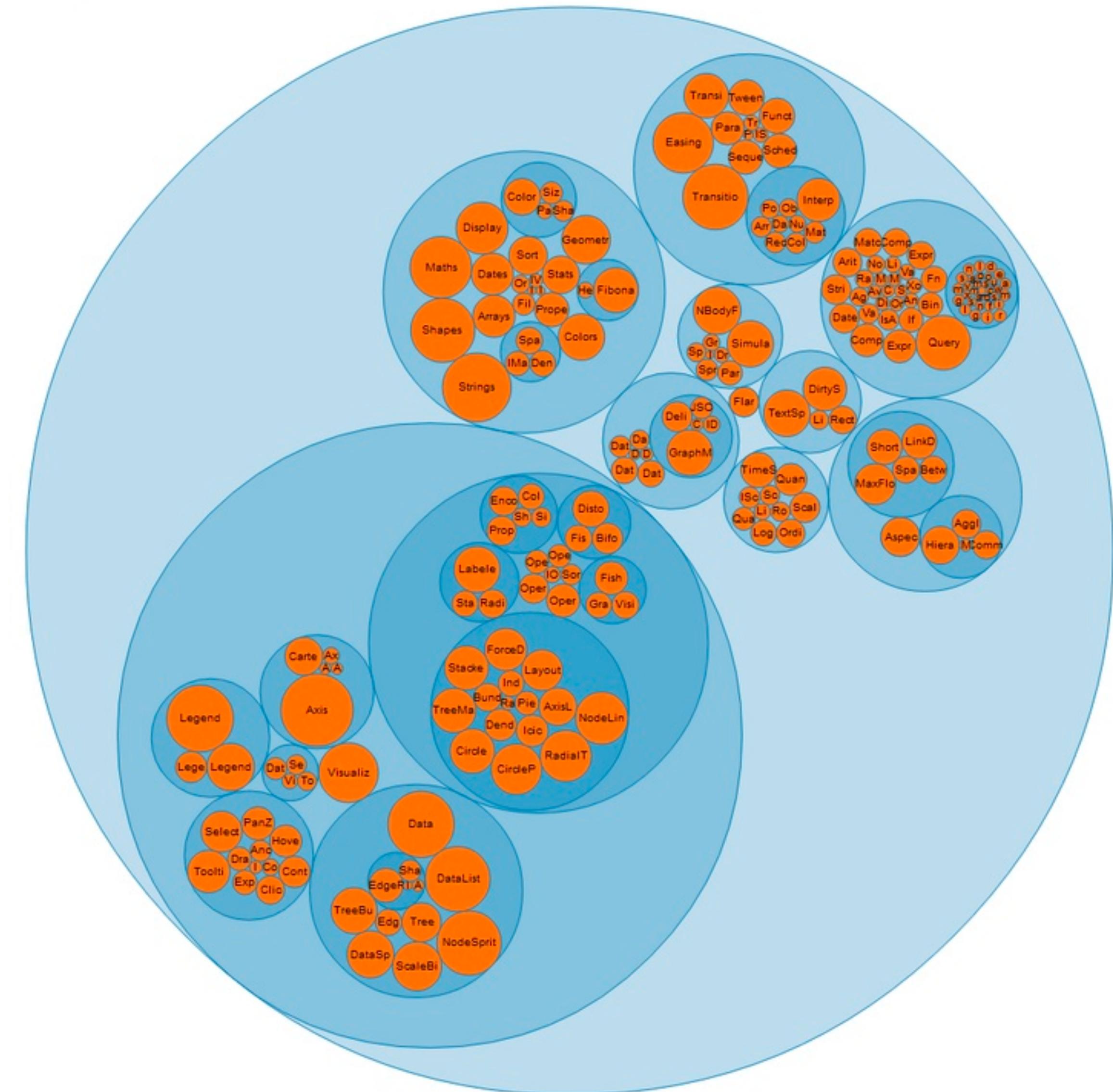
```
<body>
<h1>Hello</h1>
<p>
Check out my
<a href="/page">Page!</a>
It's the best page out there
</p>
<p>Come back soon!</p>
</body>
```

Tree

- Workflow-y: outlines with nested bullet points
- A hierarchy



Circle Packing

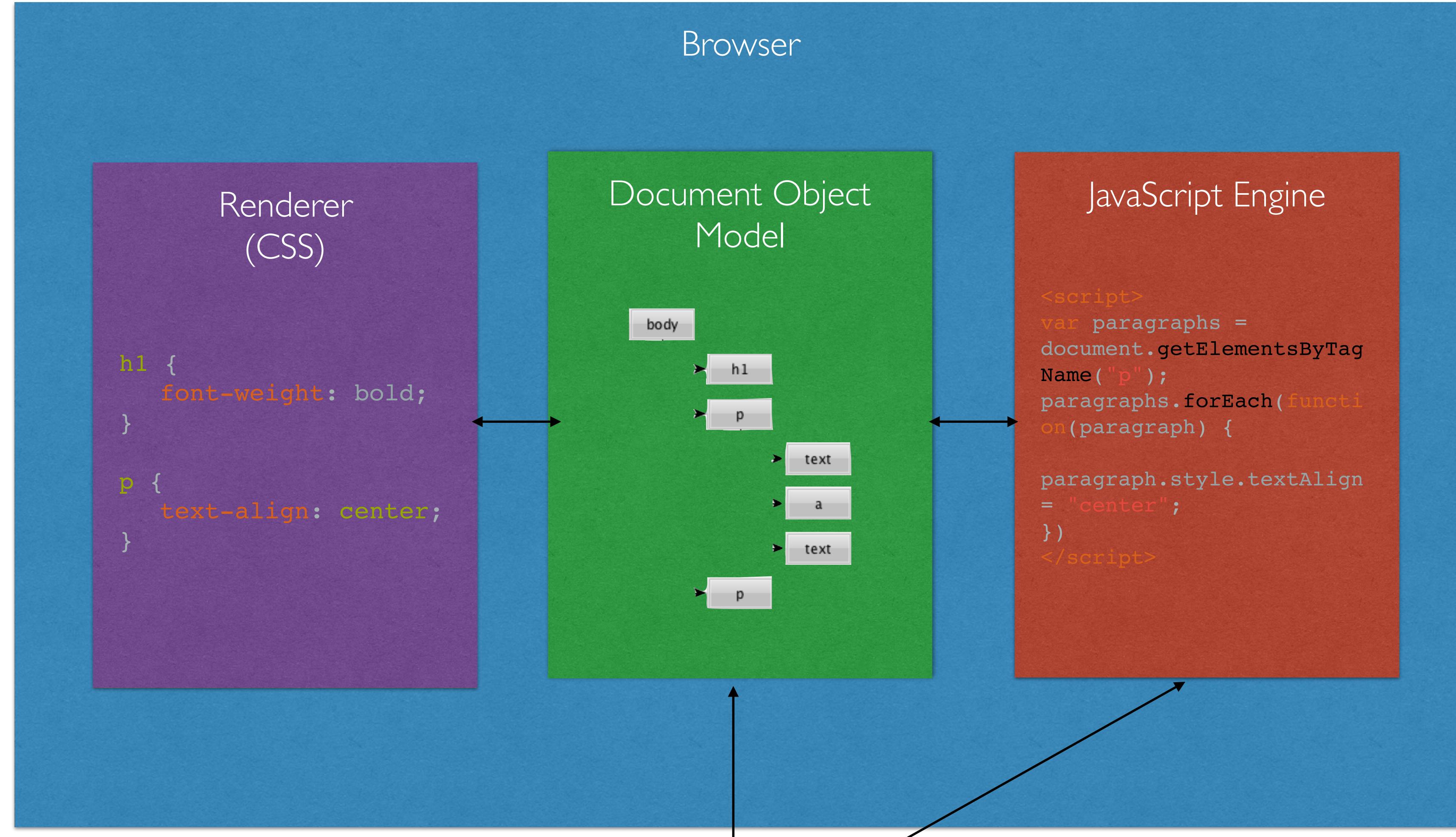


JavaScript ❤️ DOM

- We have this Object Model of an HTML document, how do we search for it, get to it, change it?
 - <script> elements!
 - We can put <script> elements of JavaScript into our DOM that can interact with the DOM
- How do you reference the DOM inside JavaScript?
 - The *document* object

The *document* Object

- Global reference to the HTML document
- Provides methods for:
 - Navigating the DOM
 - Manipulating the DOM
- The *document* object is the important connection between the DOM and JavaScript code



the **document** object is what connects these two things

Tree Structures are easy to navigate

- At any point in the DOM you are at a Node
- No matter where you go, you're still at a Node
 - Child
 - Parent
 - Sibling
 - All return Nodes
- All Nodes share similar DOM navigation methods

Traversing the DOM

- **Access children**

- `element.children`, `element.lastChild`, `element.firstChild`

- **Access siblings**

- `element.nextElementSibling`, `element.previousElementSibling`

- **Access parent**

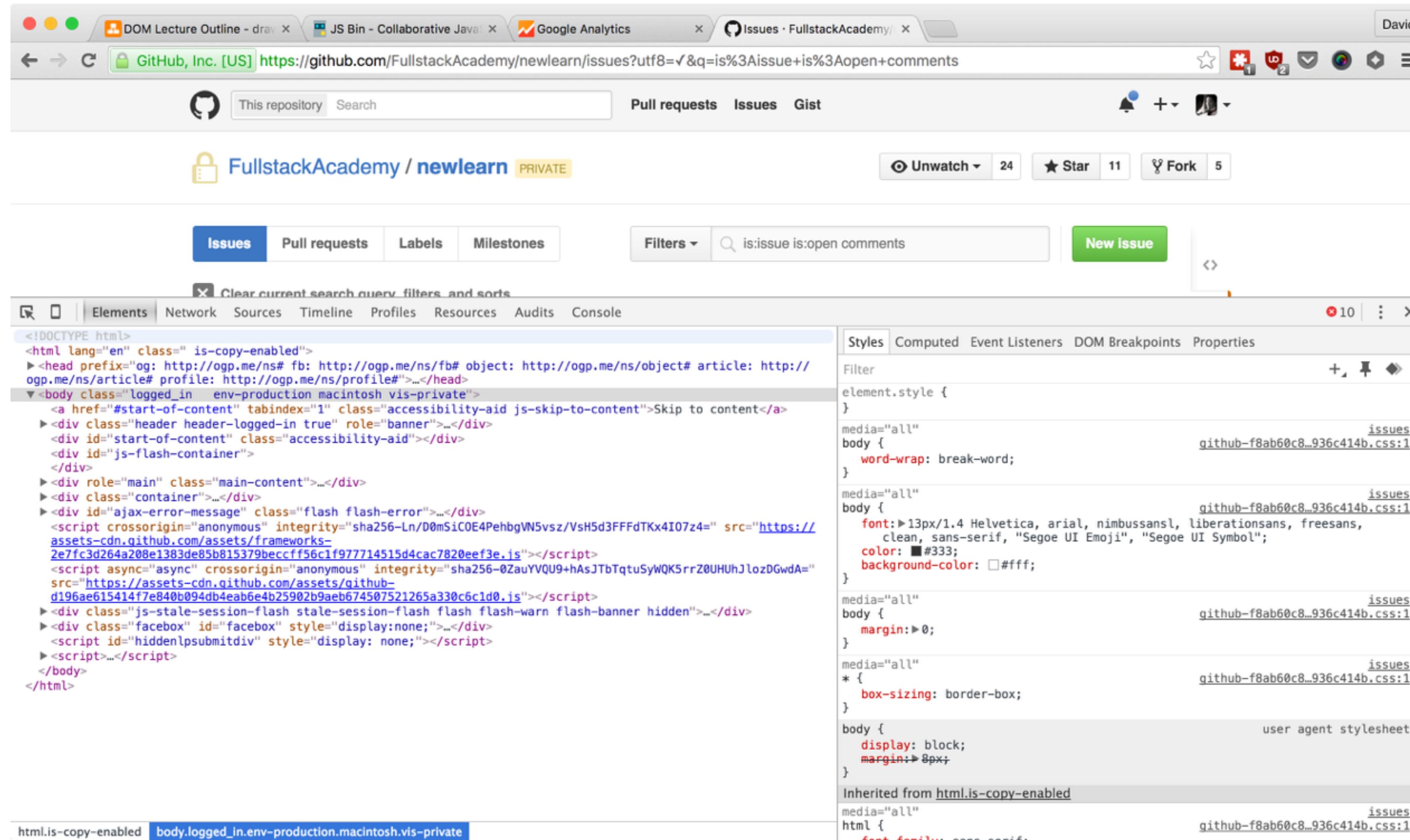
- `element.parentElement`

Navigating the DOM

○ Searching the DOM

- **getElementById** (find a node with a certain ID attribute)
 - `document.getElementById("will");`
- **getElementsByClassName** (find nodes with a certain CLASS ATTRIBUTE)
 - `document.getElementsByClassName("will");`
- **getElementsByTagName** (find nodes with a certain HTML tag)
 - `document.getElementsByTagName("div");`
- **querySelector, querySelectorAll** (search using CSS selectors)
 - `document.querySelector("#will .will:first-child");`

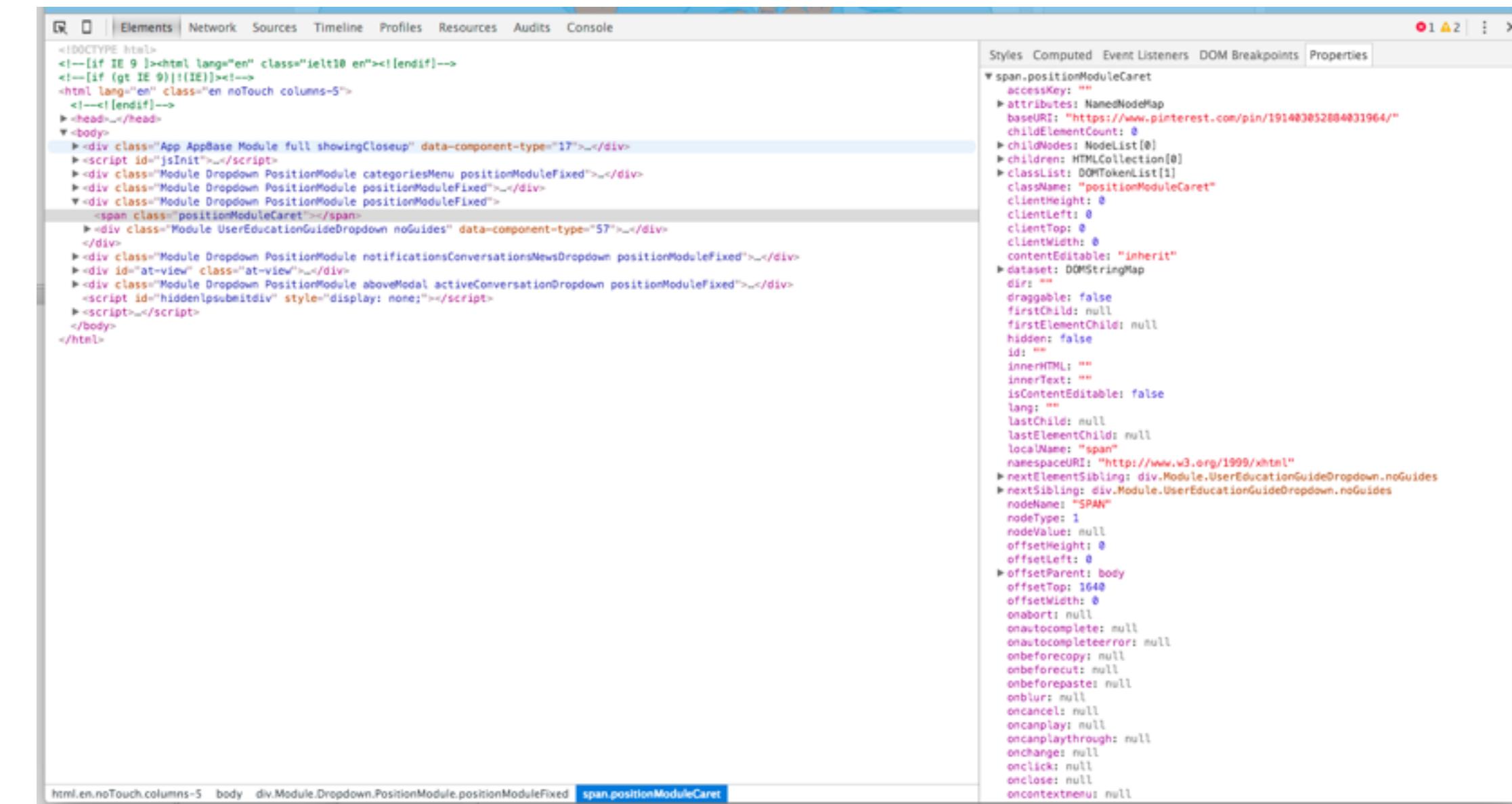
Chrome Developer Tools



Elements have lots of Attributes

- Elements are JavaScript Objects
- Elements have Attributes that are JavaScript properties
- Attributes define how the Element looks and responds to User activity

One node



hundreds of properties!

Let's Check That Out!

WORKSHOP