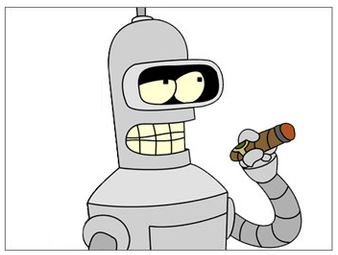
Introduction to the Document Object Model

WHO ARE YOU,



AND WHY SHOULD I CARE?

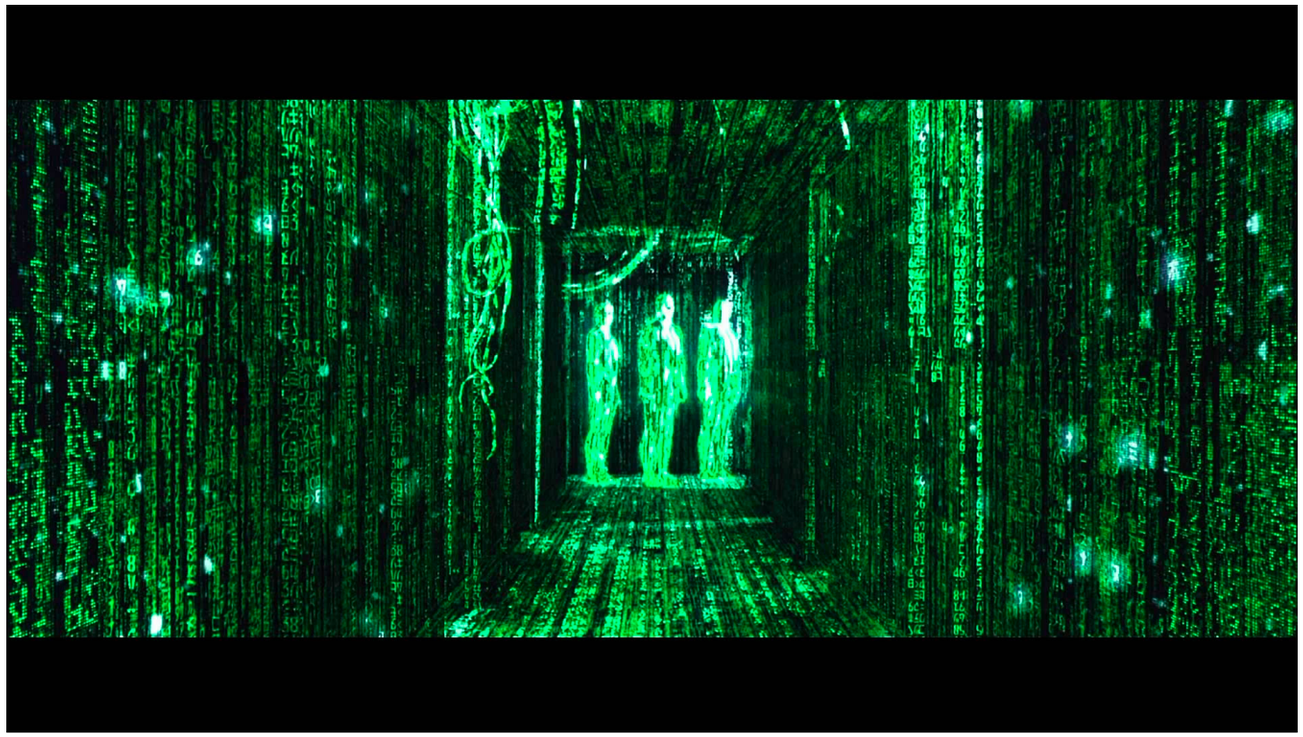
The Document Object Model is what allows web pages to render, respond to user events and change

Serialization

- Turning something abstract into a format that can be stored and transmitted
- Human Example: Thoughts into words
- Deserialization: Reading a book and visualizing the ideas in your mind
- The DOM is the "deserialized" version of HTML

Serialization: turning an abstract idea into a format that can be stored in memory and/or transmitted (for example, across a network) to be reconstructed later (possibly in a different computer environment).

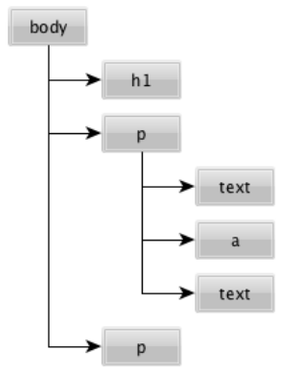
This is how most of us see the web. We experience it as users. But underneath every web page there is a complicated life of objects interacting. Just like in the Matrix...



This is the DOM. It's not what we see, it's how the things under the hood are behaving.

HTML vs DOM

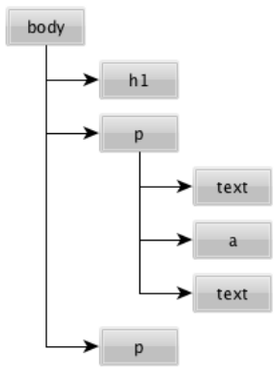
```
<body>

  <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>
```



http://software.hixie.ch/utilities/js/live-dom-viewer/?%3C!DOCTYPE%20html%3E%0A%3Chtml%3E%0A%09%3Chead%3E%0A%09%20%20%20%20%3Ctitle%3EMy%20first%20web%20page%3C%2Ftitle%3E%0A%09%3C%2Fhead%3E%0A%09%3Cbody%3E%0A%09%20%20%20%20%3Ch1%3EHello%20world!%3C%2Fh1%3E%0A%09%20%20%20%20%3Cp%3E%3Cb%3EI%27m%20very%20excited%3C%2Fb%3E%20to%20be%20exploring%20the%20Document%20Object%20Model.%20Here%27s%20the%20%3Ca%20href%3D%22wikipedia.org%2FDOM%22%3EWikipedia%3C%2Fa%3E%20page%20on%20the%20topic.%3C%2Fp%3E%0A%09%3C%2Fbody%3E%0A%3C%2Fhtml%3E

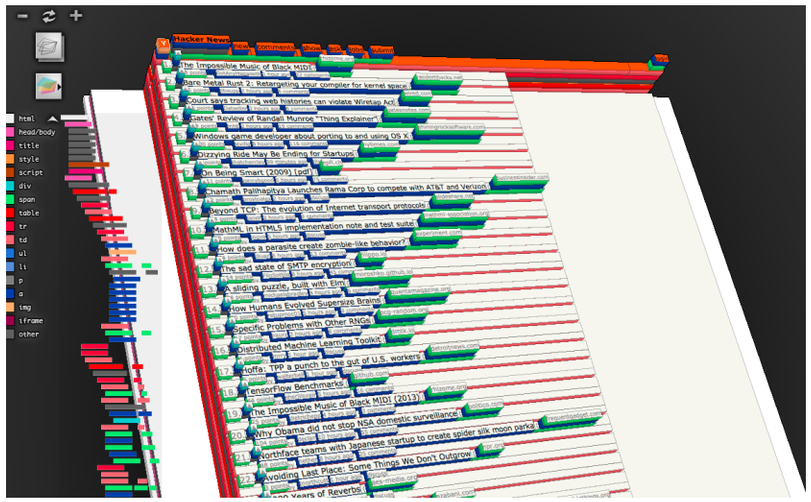
The DOM is a Tree

- 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



- whats the parent, whats the child, what's the sibling.
- similar to folder structure (show sublime text)

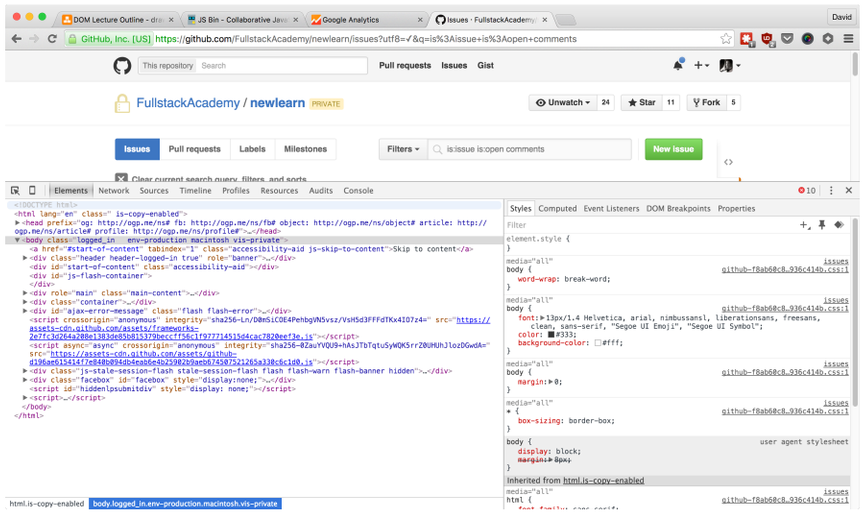
Firefox 3D Demo



Command Shift M in Firefox with Tilt plugin

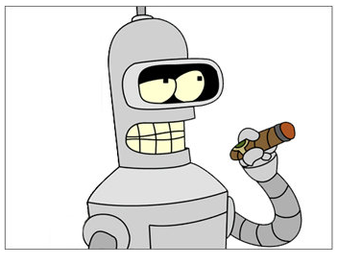
Show the stuff inside the Nodes
hacker news is a nice simple site

Chrome Developer Tools



source = serialized stuff
developer tools = shows the actual dom representation

WHO ARE YOU,



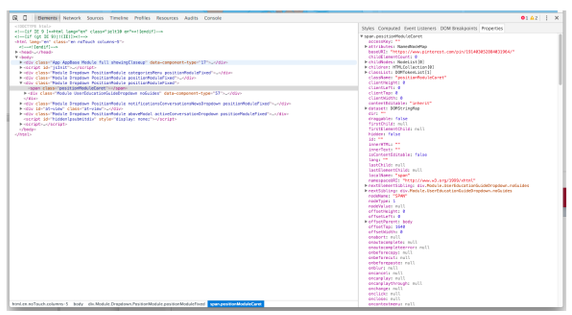
AND WHY SHOULD I CARE?

The DOM makes possible to use JavaScript to manipulate the document content and structure

In other words, it allows us to write code that dynamically changes what the user is seeing.

Nodes have lots of Attributes

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



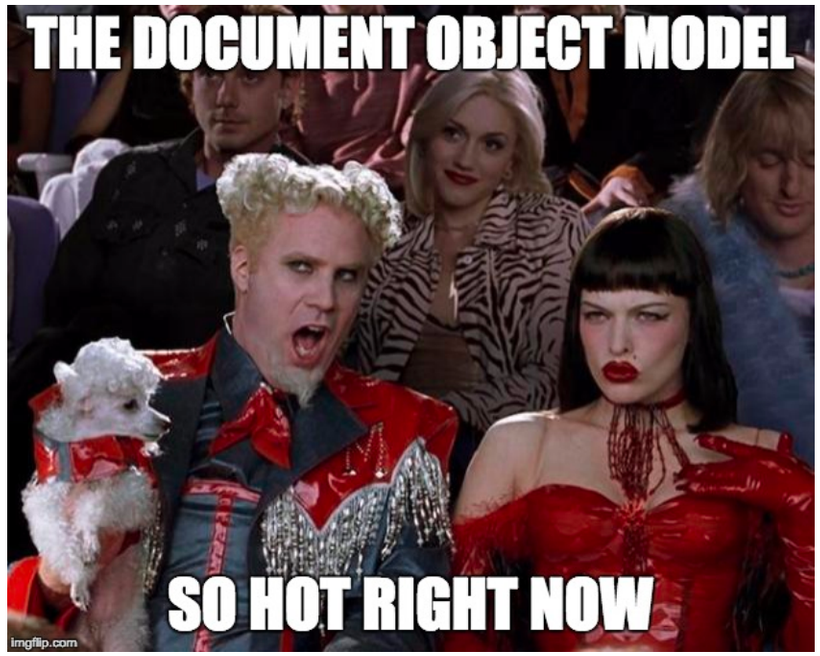
one node

hundreds of properties!

Node are just JavaScript ObjectsL they're bags of attributes.
These objects contain 100s of attributes that let you change the node in two main ways:

- how it looks and is drawn by the browser
- how it responds to user input

Now we'll get into how to use JS to manipulate these Nodes.

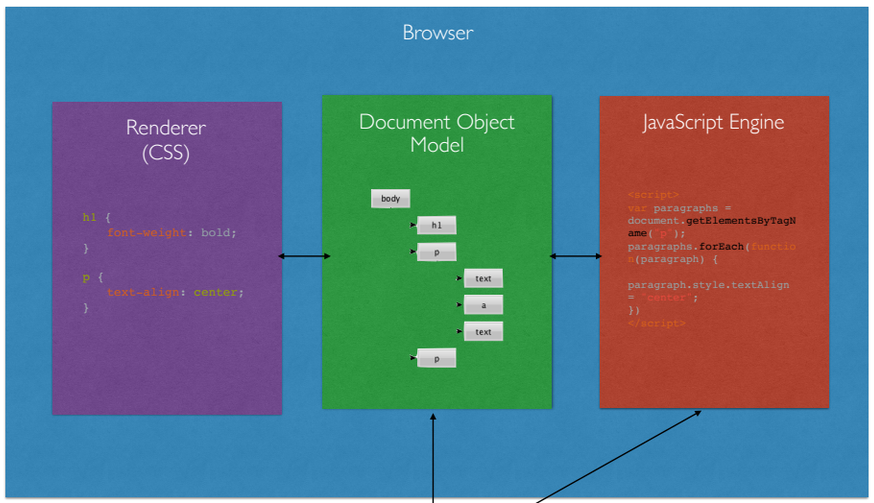

THE DOCUMENT OBJECT MODEL
SO HOT RIGHT NOW
imgflip.com

The *document* Object

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

Without the document object, we could write JavaScript but we wouldn't be able to manipulate the DOM. The browser gives us access to the object so that we can do these things and that's the power behind HTML and JS.

CSS of course is involved as well.



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

Navigating the DOM

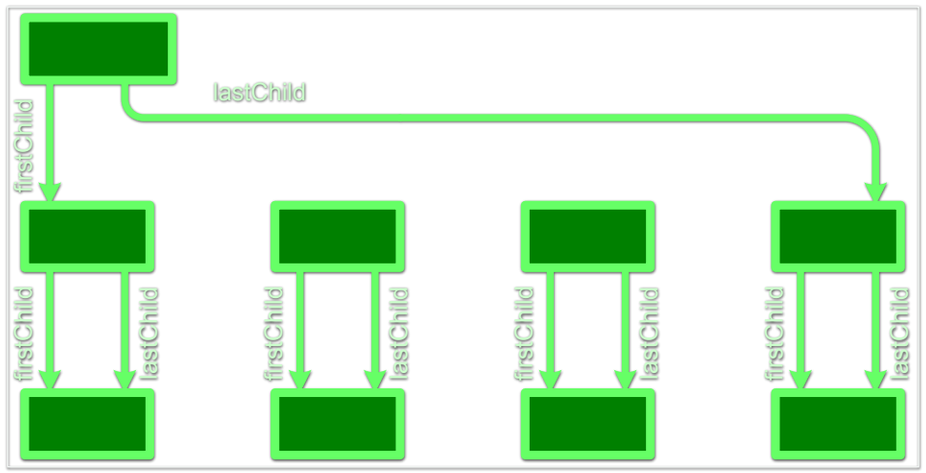
- **Searching the DOM**
  - getElementById (find nodes with a certain ID attribute)
    - `document.getElementById("will");`
  - getElementsByClassName (find nodes with a certain CLASS ATTRIBUTE)
    - `document.getElementsByClassName("will");`
  - getElementsByTagName (find nodes with a certain HTML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

One thing to be careful about is that getElementById returns the first Element it finds. That's why you don't want to have multiple IDs on a page.

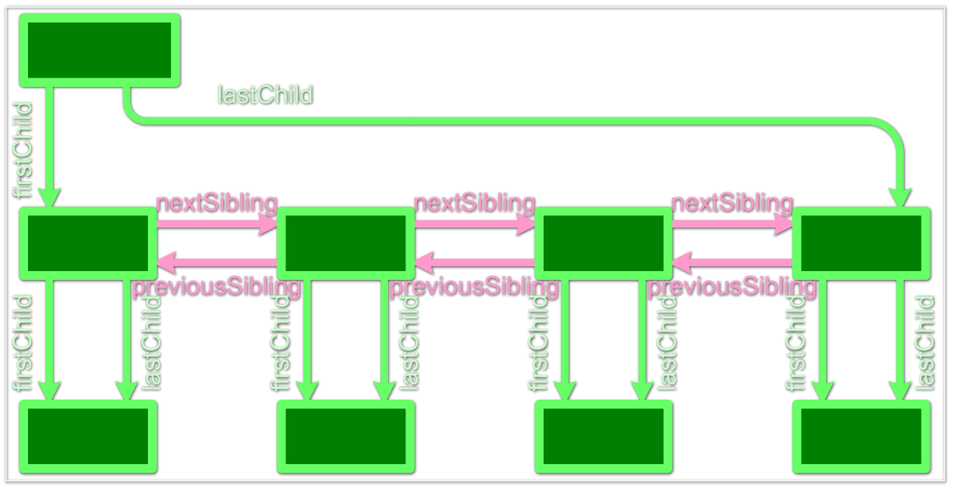
getElements... returns an HTMLCollection (array-like object) of Elements, even if there is only one element with that class or tag. Then you can use your JavaScript to manipulate all of the elements.

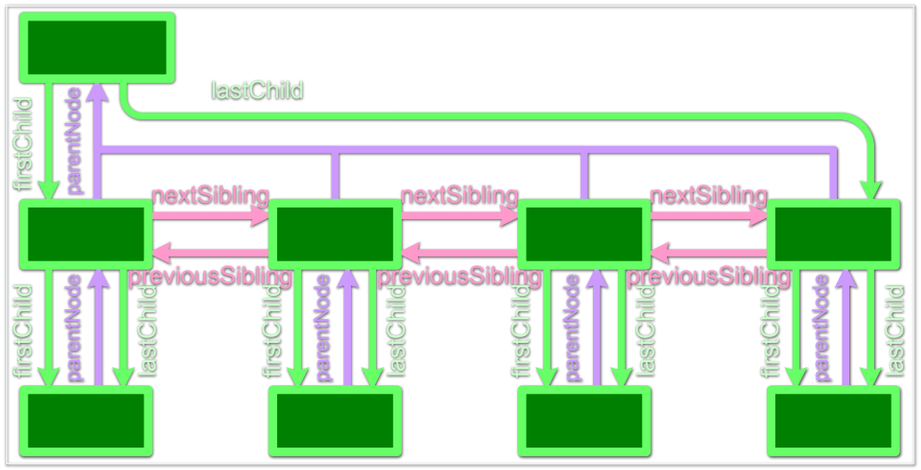
Traversing the DOM

- **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 Nodes share similar DOM navigation methods



where have we seen firstChild and lastChild? (answer = css)


lastChild
firstChild
nextSibling
nextSibling
nextSibling
previousSibling
previousSibling
previousSibling
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild


lastChild
firstChild
parentNode
nextSibling
nextSibling
nextSibling
previousSibling
previousSibling
previousSibling
firstChild
parentNode
lastChild
firstChild
parentNode
lastChild
firstChild
parentNode
lastChild
firstChild
parentNode
lastChild

Traversing the DOM

- Access children
  - `element.children, element.lastChild, element.firstChild`
- Access siblings
  - `element.nextElementSibling, element.previousElementSibling`
- Access parent
  - `element.parentElement`


WORKSHOP