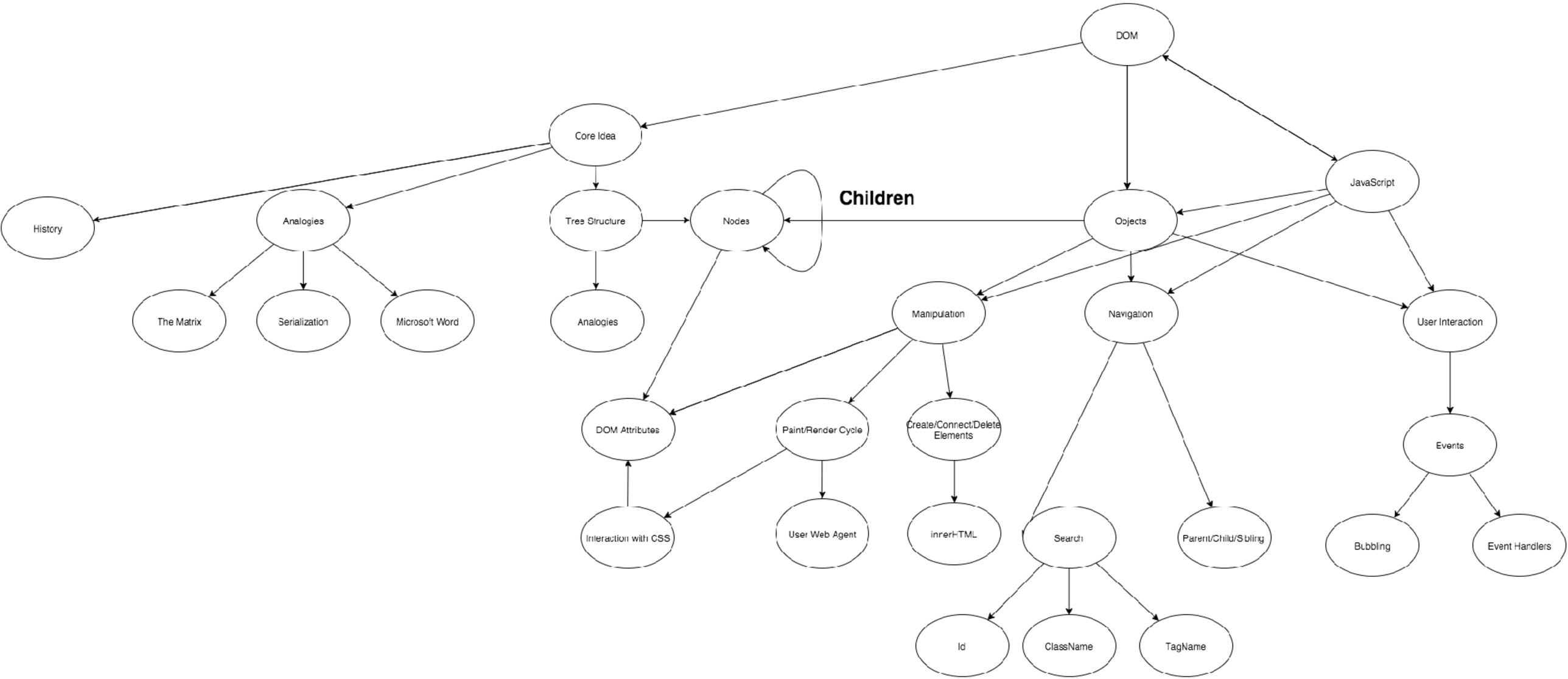
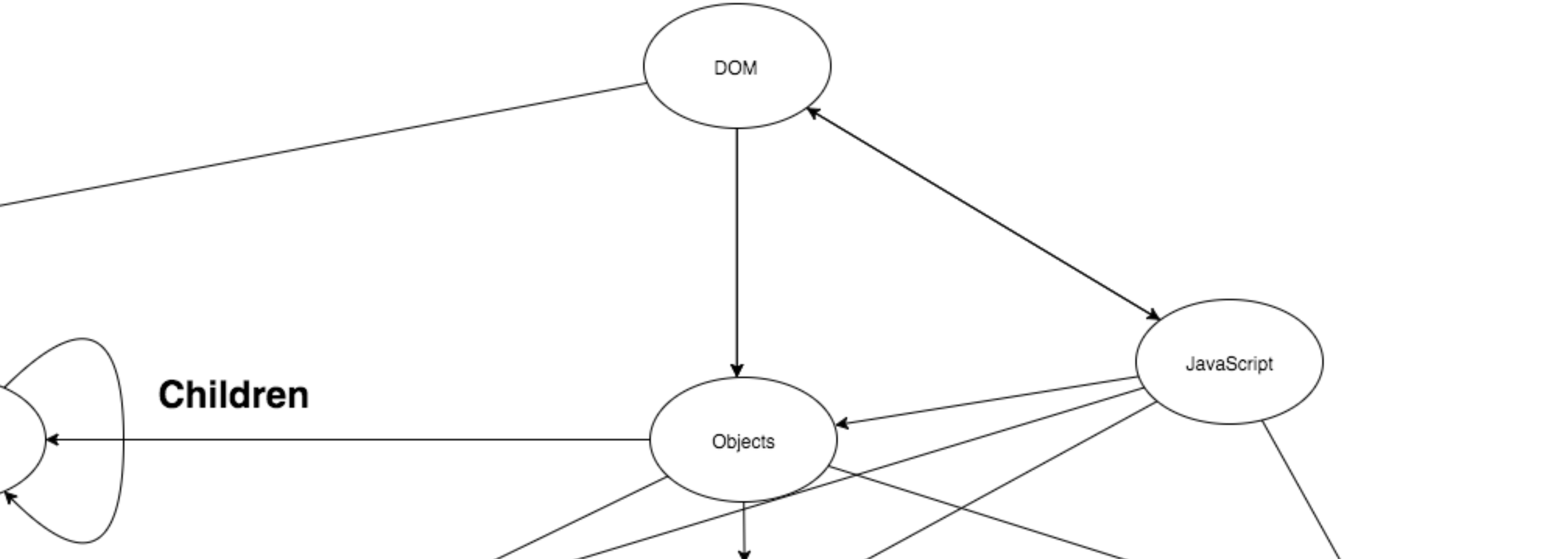
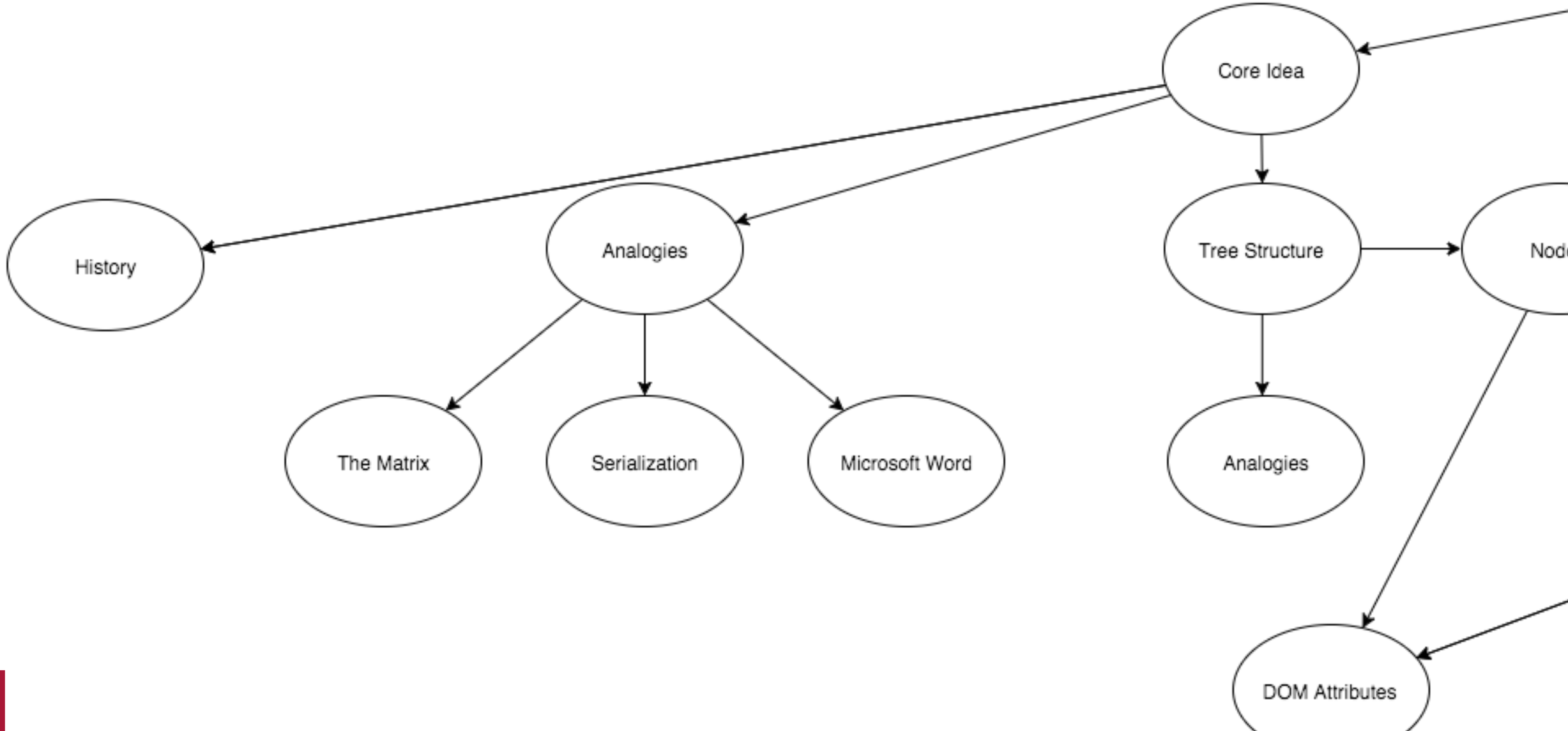
Introduction to the Document Object Model

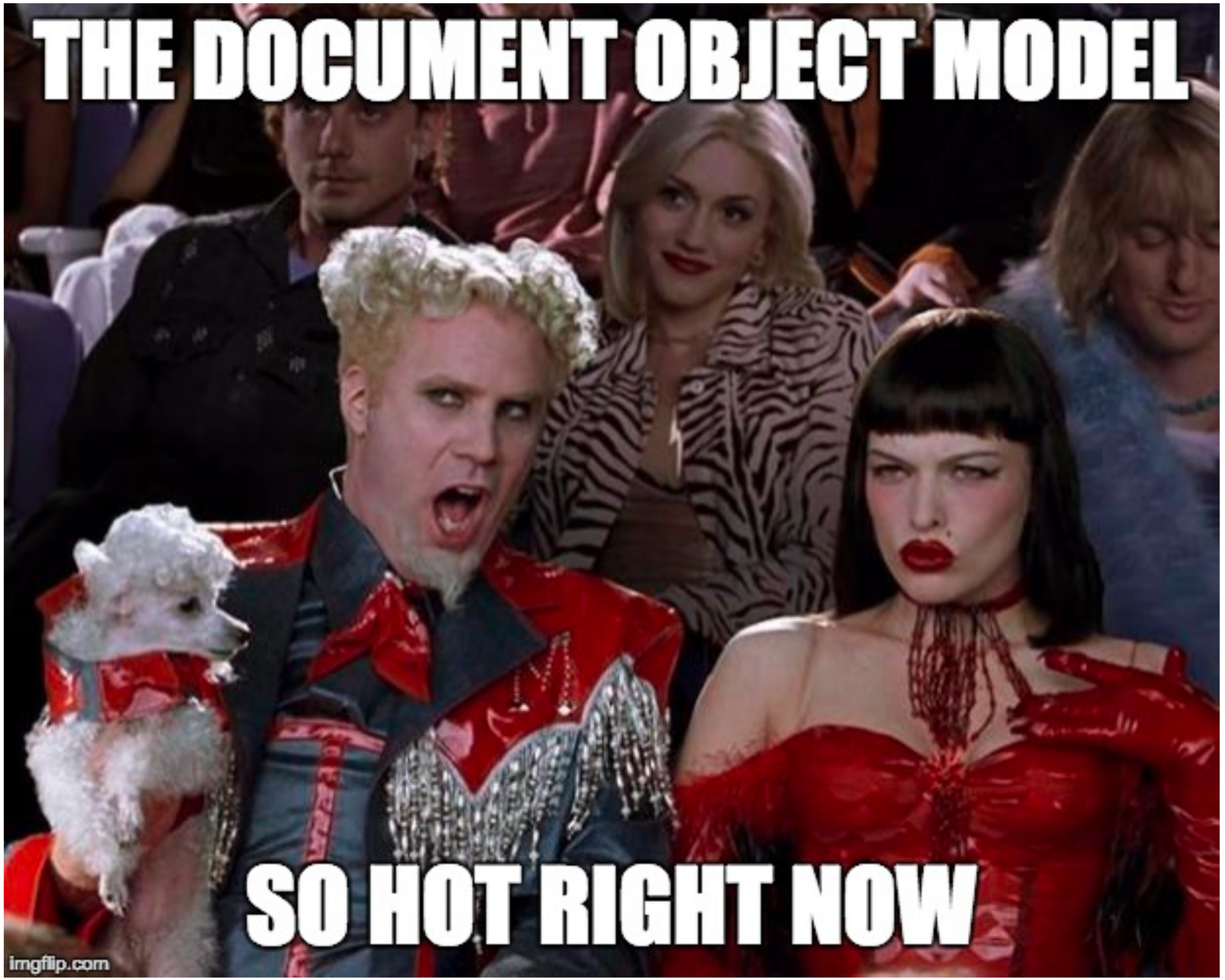

DOM
Core Idea
History
Analogies
The Matrix
Serialization
Microsoft Word
Tree Structure
Nodes
Children
Analogies
JavaScript
Objects
Manipulation
Navigation
User Interaction
DOM Attributes
Paint/Render Cycle
Create/Connect/Delete Elements
Interaction with CSS
User Web Agent
innerHTML
Search
Parent/Child/Sibling
Id
ClassName
TagName
Events
Bubbling
Event Handlers


DOM
JavaScript
Children
Objects

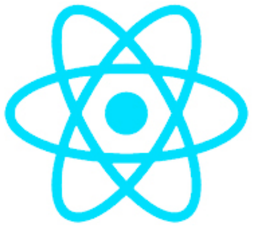

Core Idea
History
Analogies
Tree Structure
The Matrix
Serialization
Microsoft Word
Analogies
DOM Attributes

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


THE DOCUMENT OBJECT MODEL
SO HOT RIGHT NOW
imgflip.com

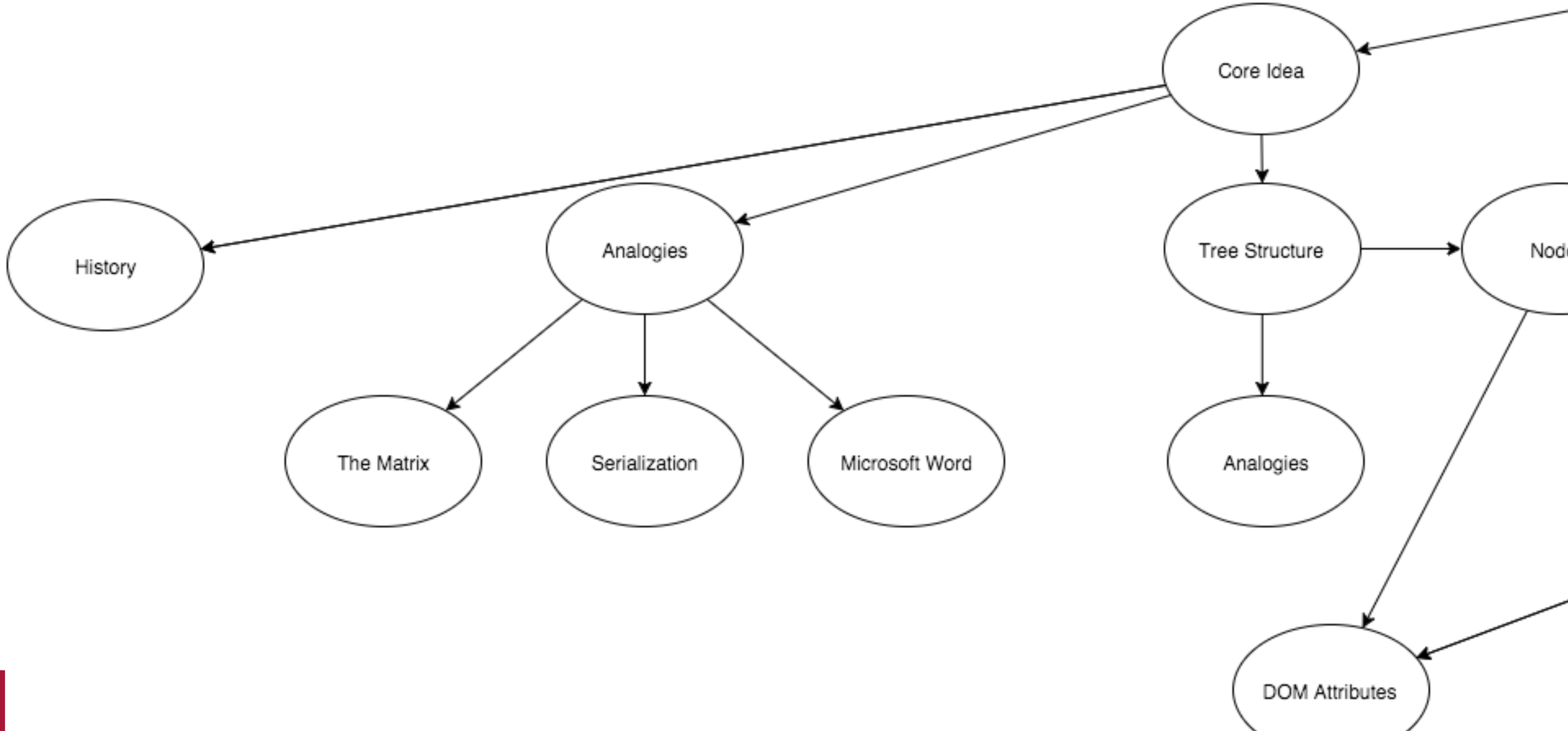
The DOM is Hot



React

polymer

WEB COMPONENTS


Core Idea
History
Analogies
Tree Structure
The Matrix
Serialization
Microsoft Word
Analogies
DOM Attributes

The History of the DOM

- The original World Wide Web was a simple idea: document retrieval through hyperlinks between documents
- No concepts of:
  - User Interactivity
  - Sessions
  - Presentation (no CSS!)

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.

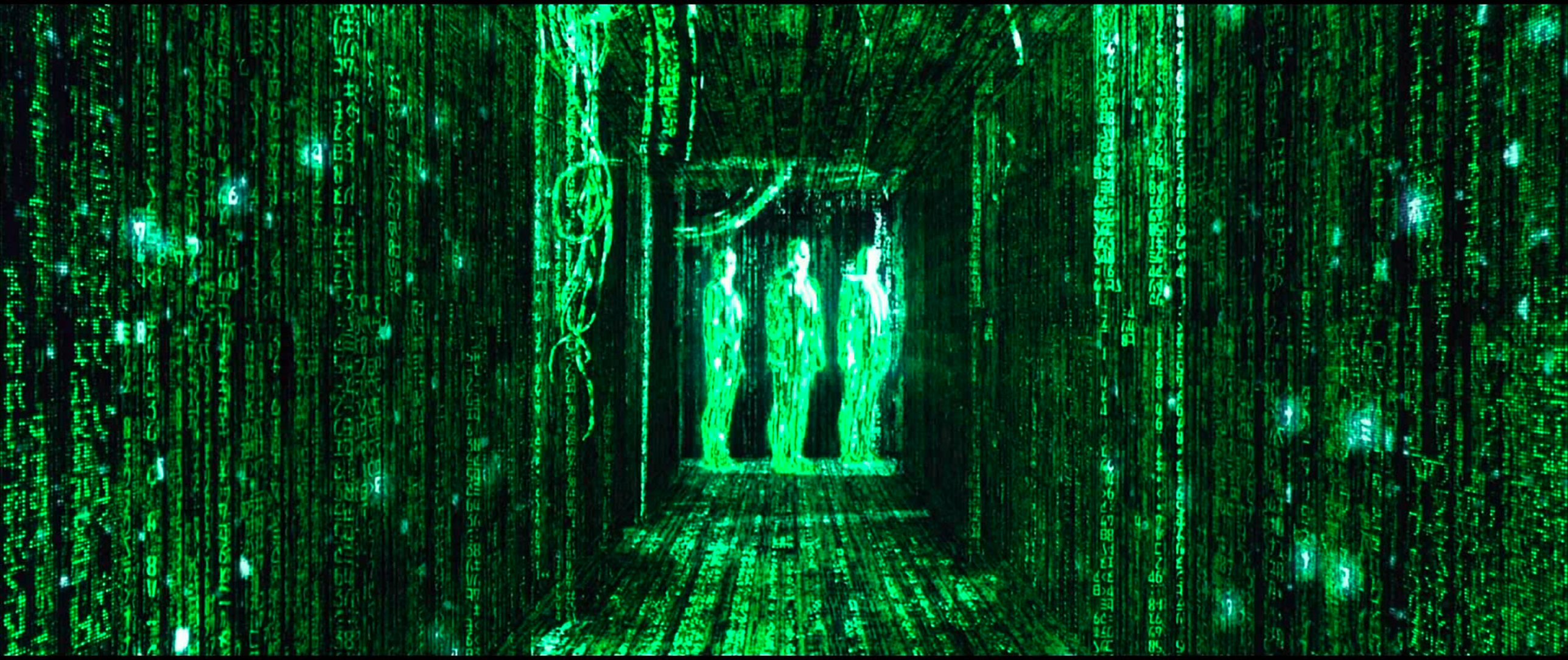
A technically correct definition of the DOM

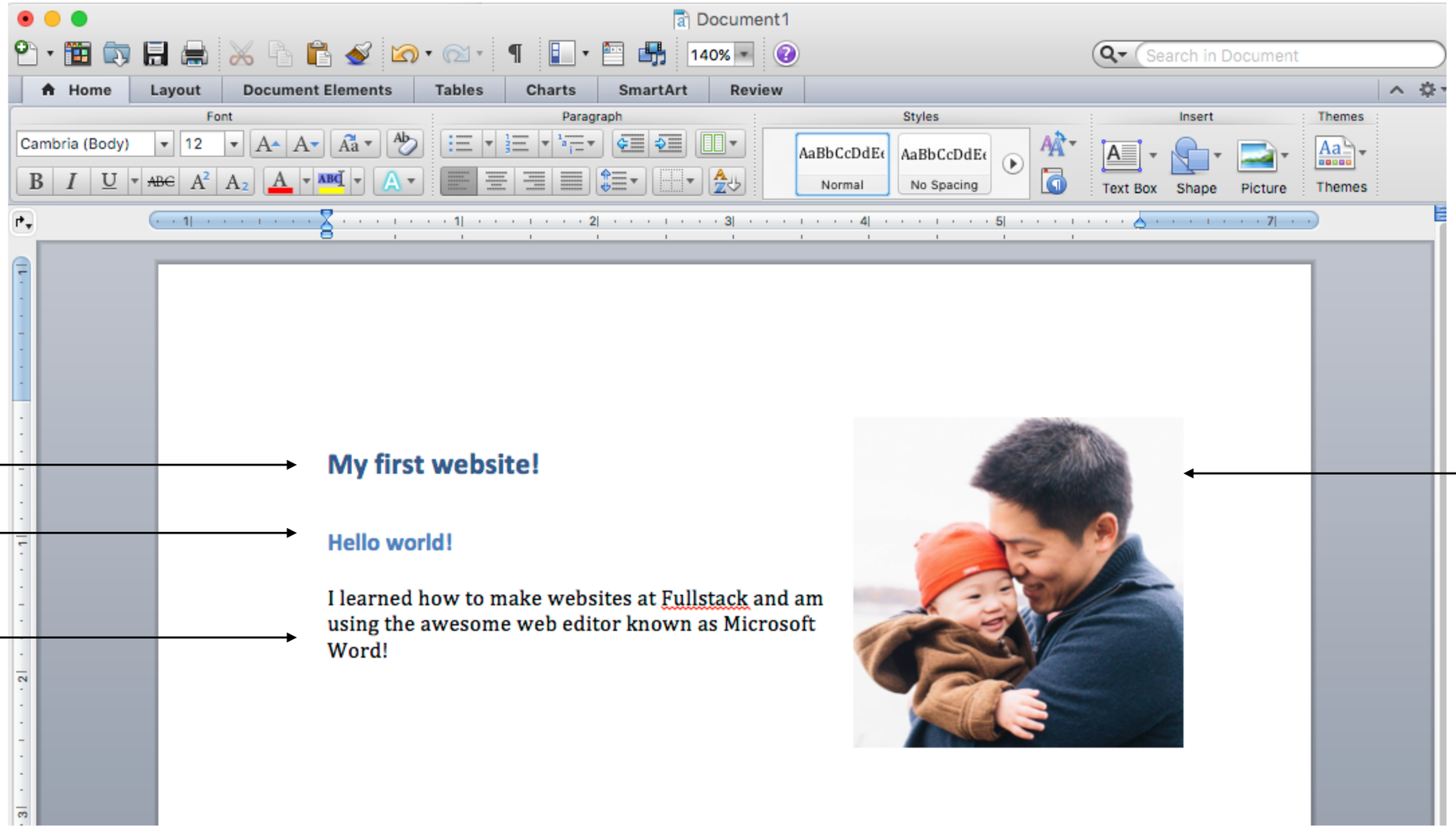
The Document Object Model (DOM) is a **cross-platform** and **language-independent** convention for representing and interacting with objects in **HTML, XHTML, and XML documents**.

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





<h1> → My first website!

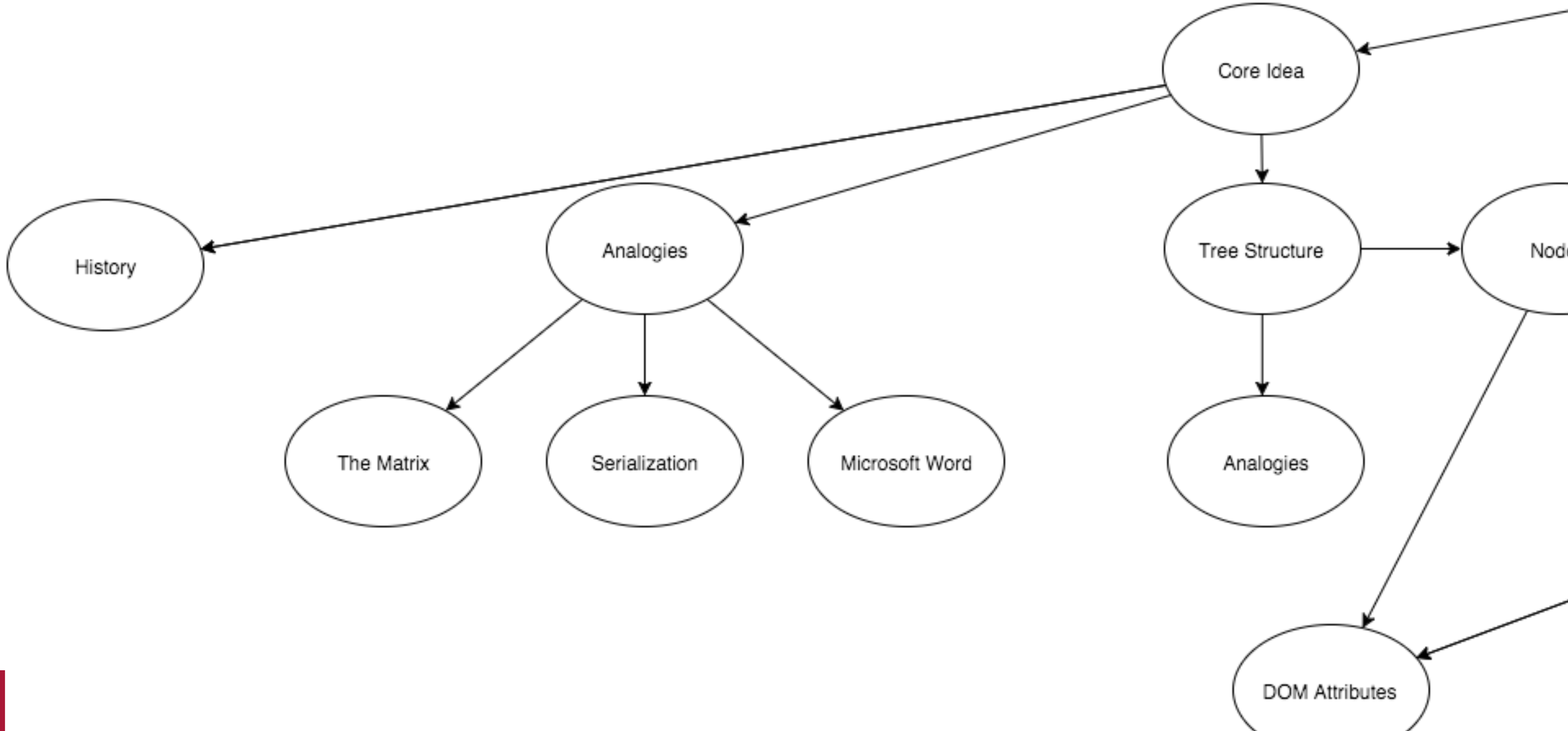
<h2> → Hello world!

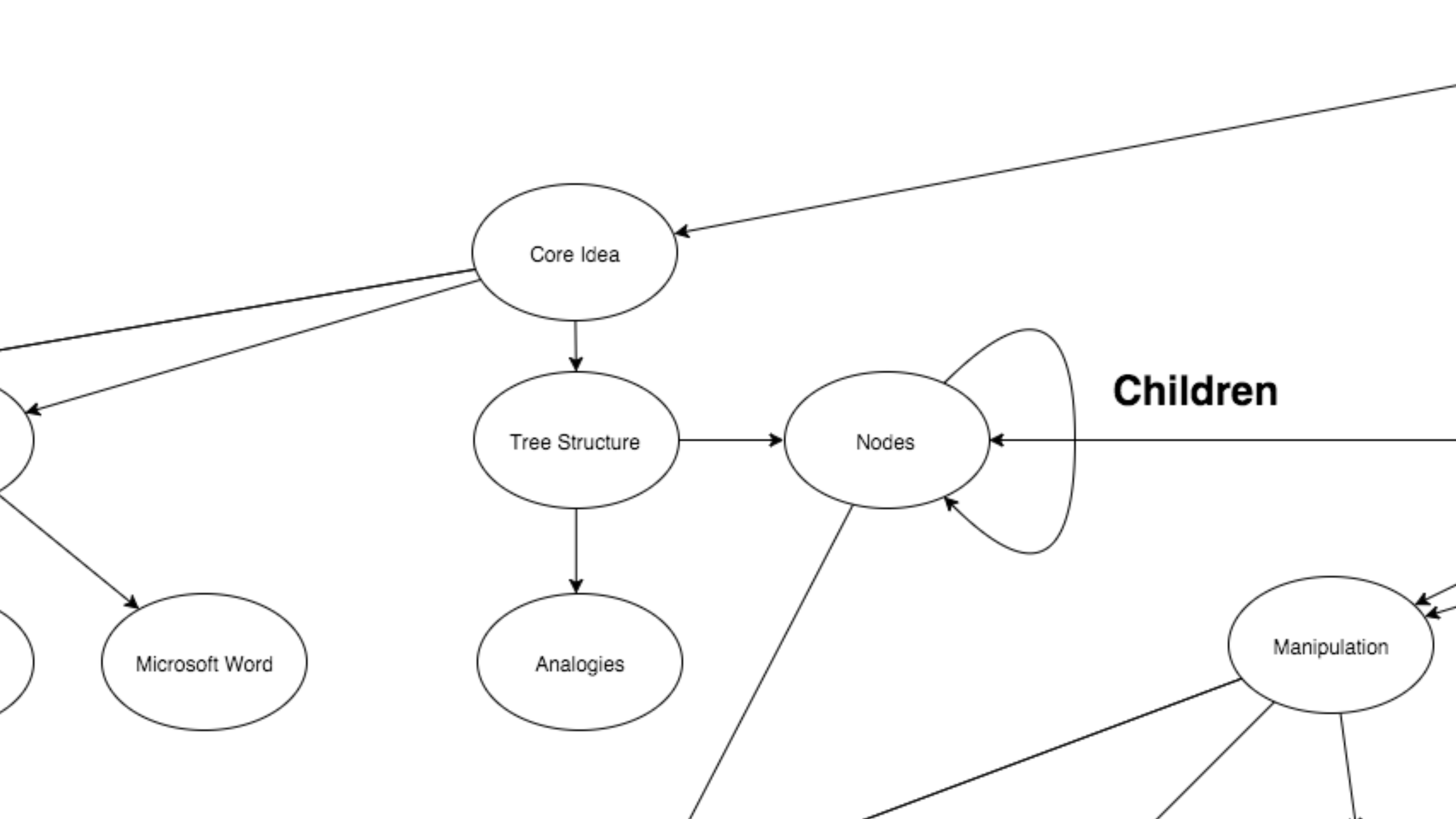
<p> → I learned how to make websites at Fullstack and am using the awesome web editor known as Microsoft Word!

 ← <img>

Serialization

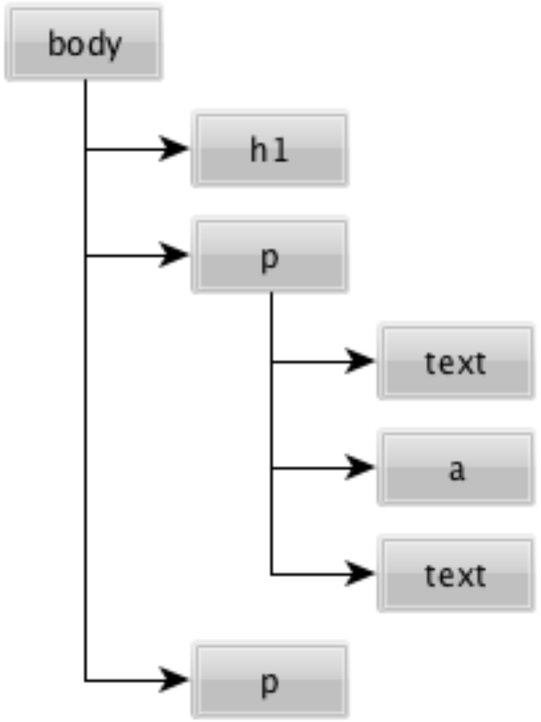
- In programming we often have a "thing" and a way to represent that "thing"
- We refer to that representation as the Serialization
- Human Example: Thoughts into words
- HTML: the serialized version of the DOM
- The DOM is the "deserialized" version of HTML
- Deserialization: Reading a book and visualizing the ideas in your mind


Core Idea
History
Analogies
Tree Structure
The Matrix
Serialization
Microsoft Word
Analogies
DOM Attributes


Core Idea
Tree Structure
Nodes
Children
Microsoft Word
Analogies
Manipulation

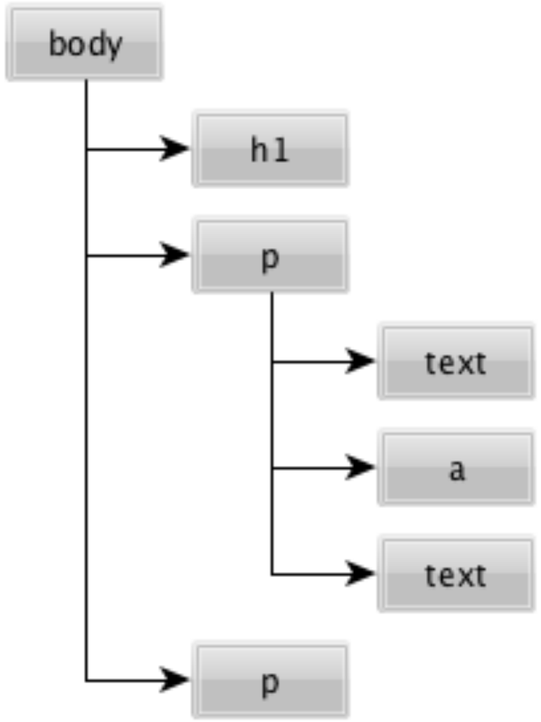
The DOM is a Tree

- 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>
```

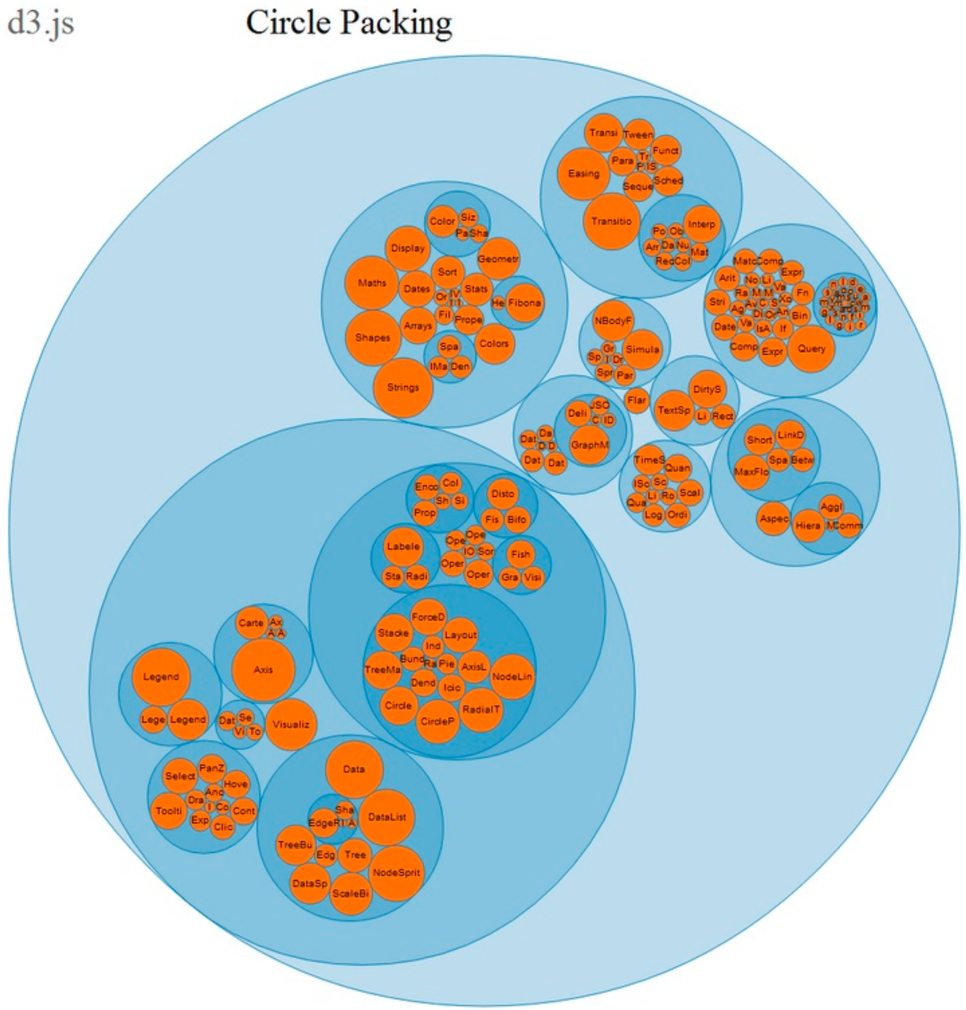


Indentation Is Important!

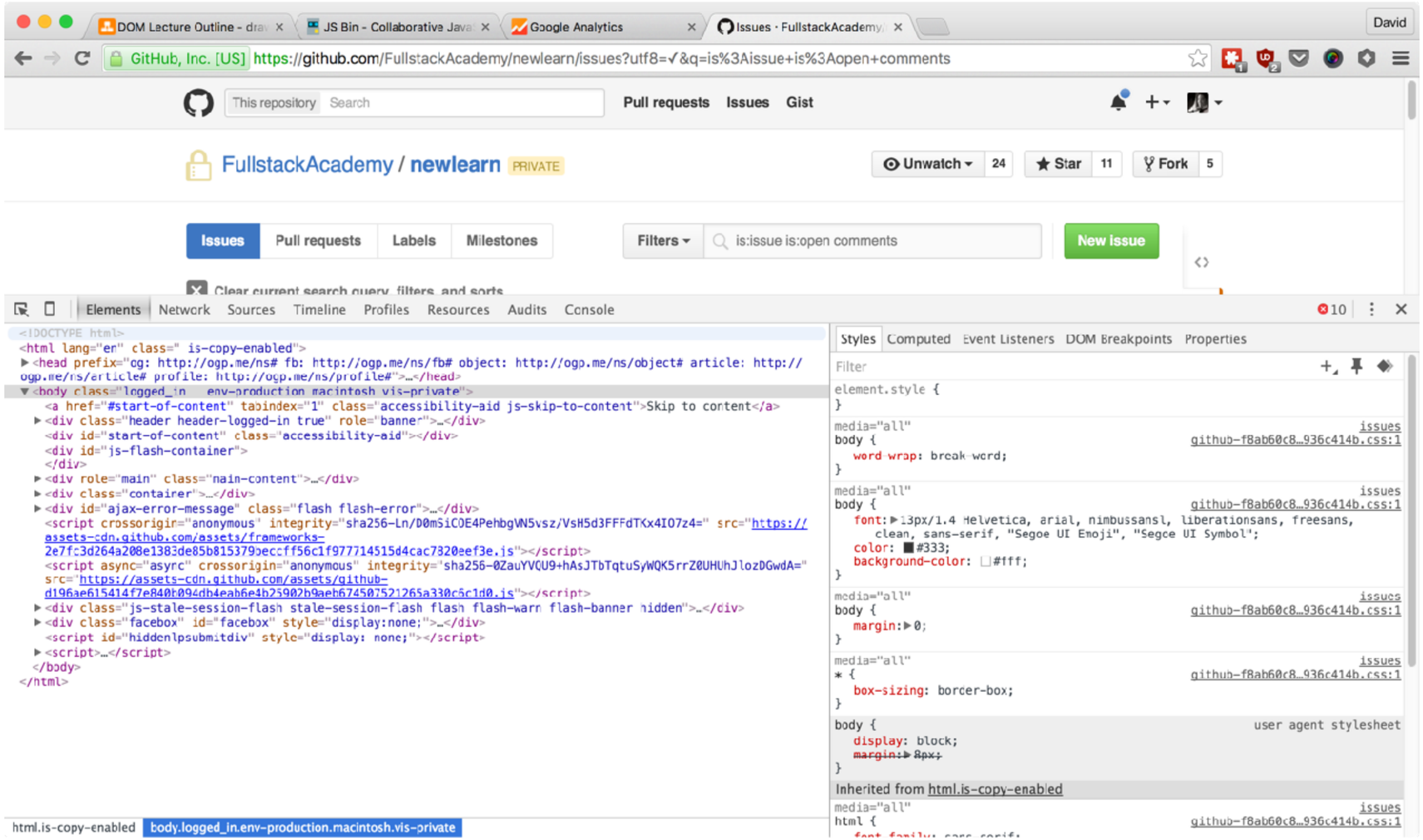
- No indentation makes it hard to see the tree structure:

```
<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>
```

Circle Packing

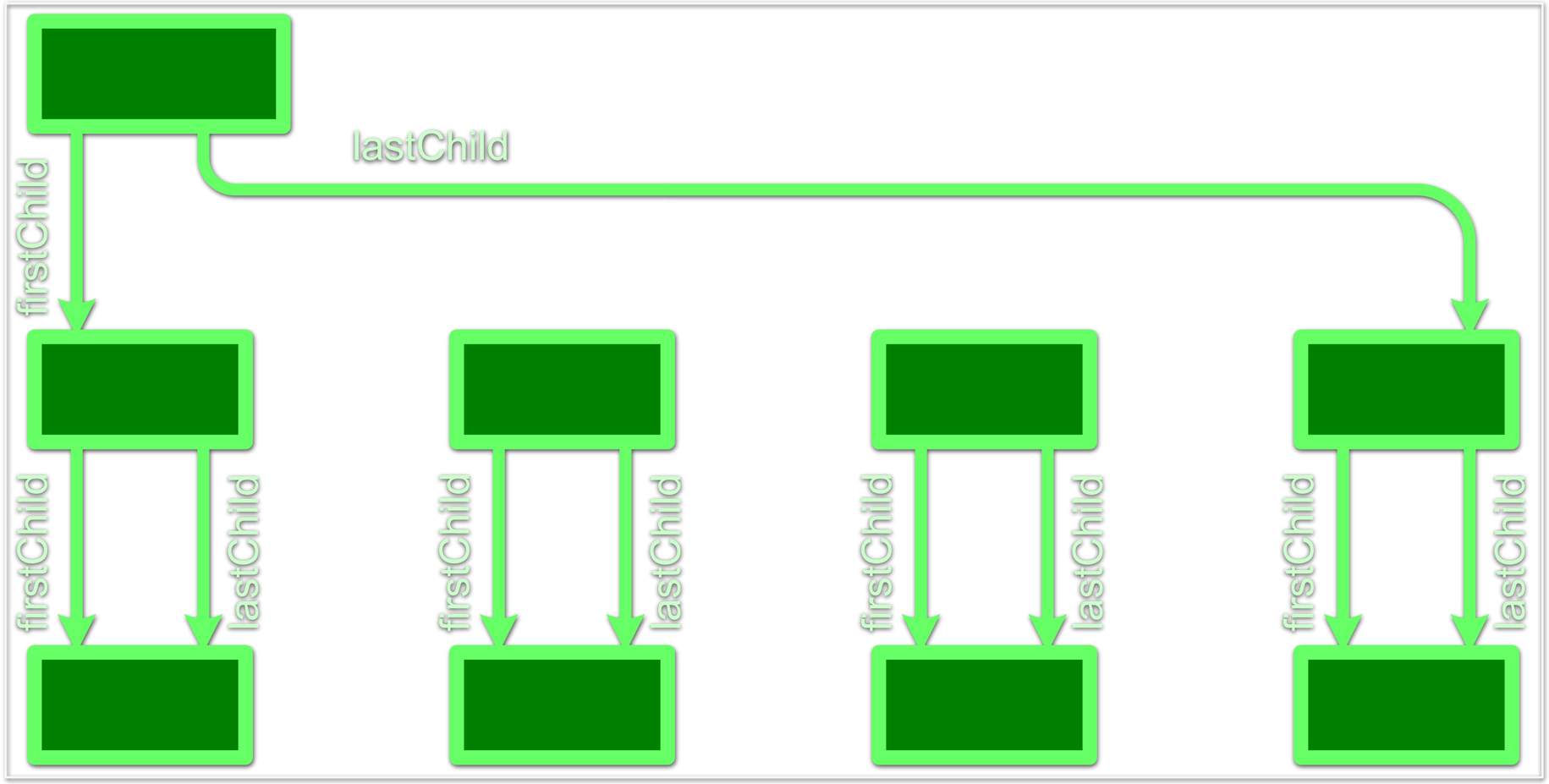


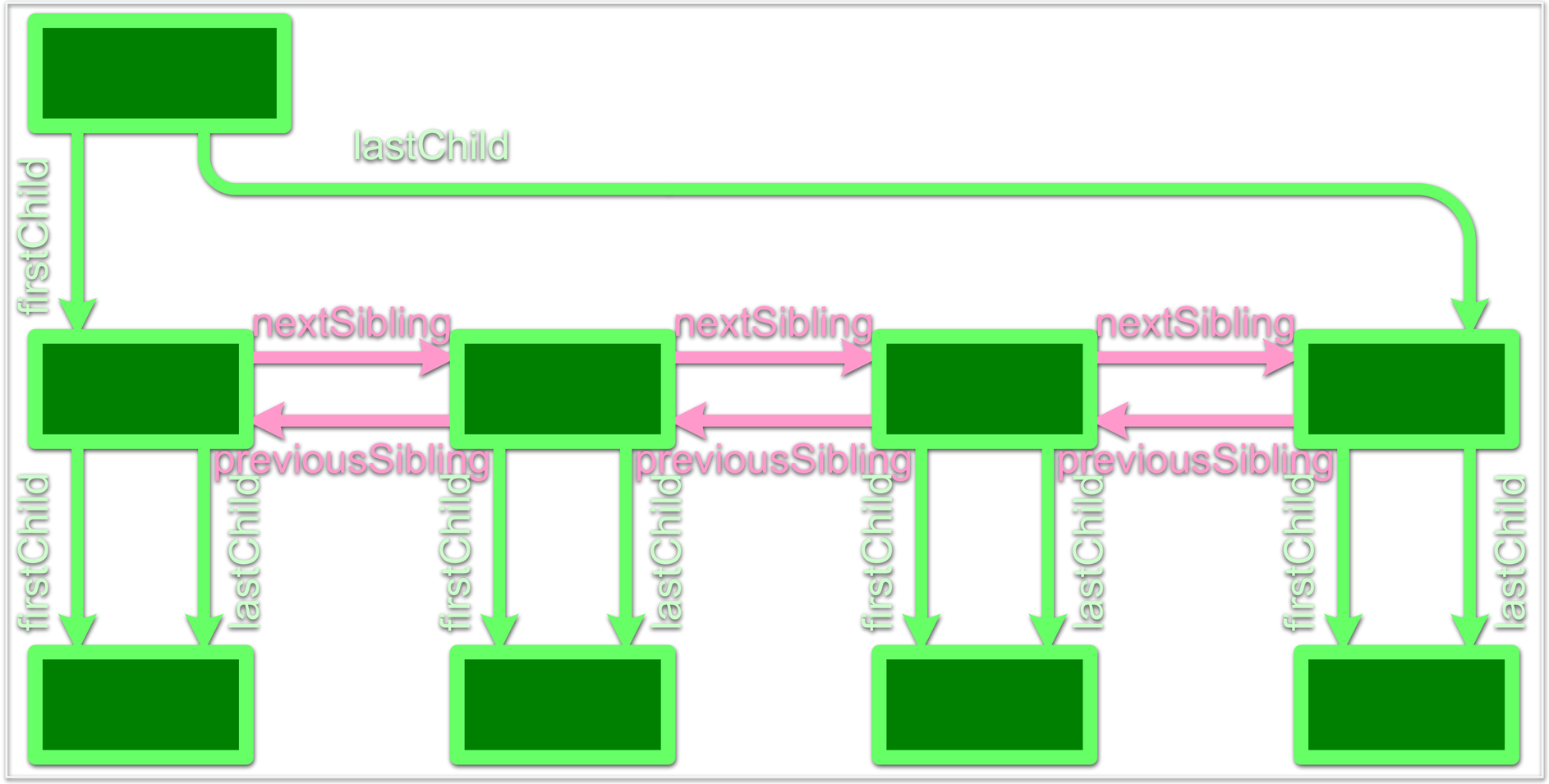
Chrome Developer Tools

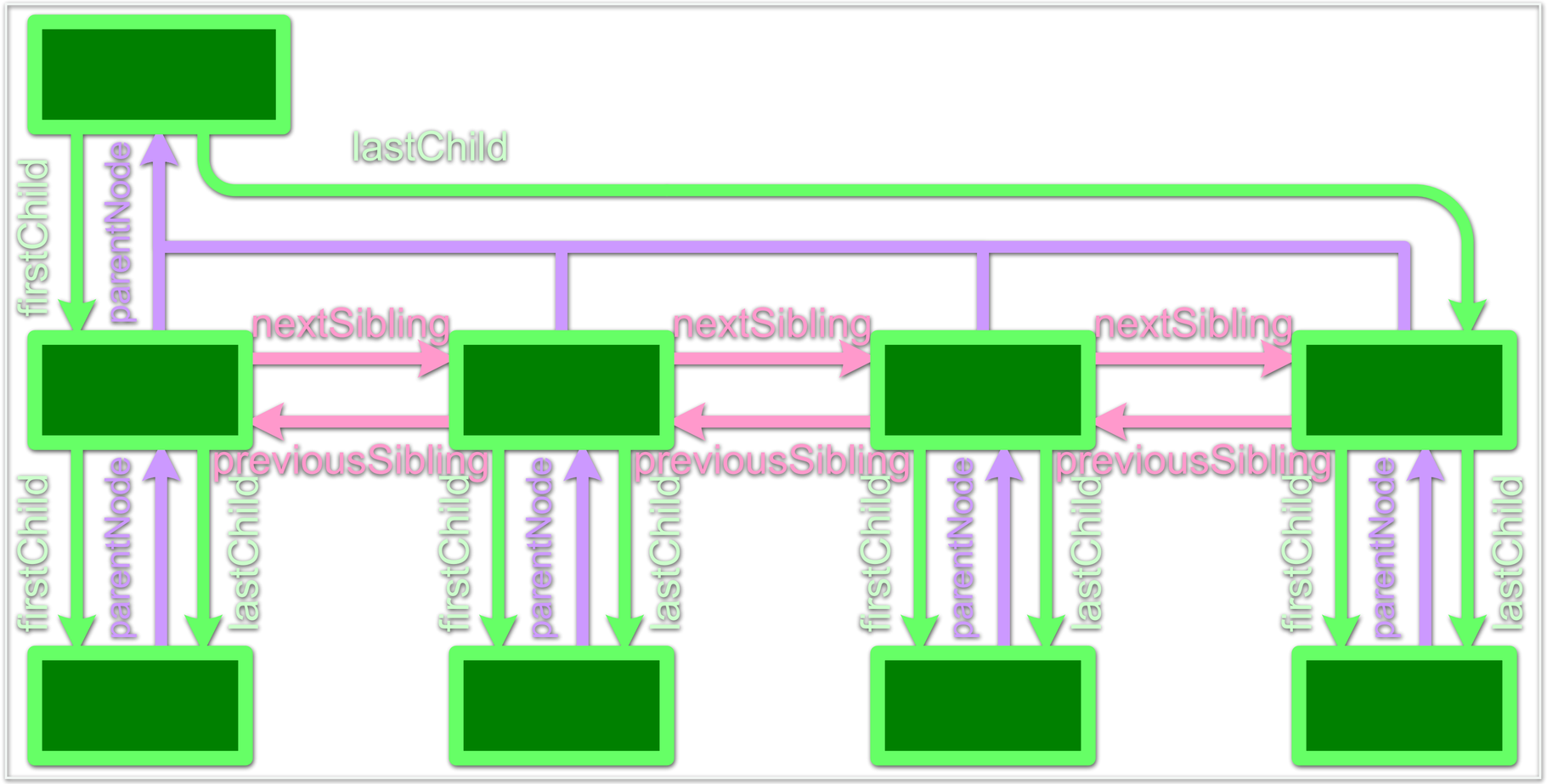


Tree Structures are easy to navigate

- 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

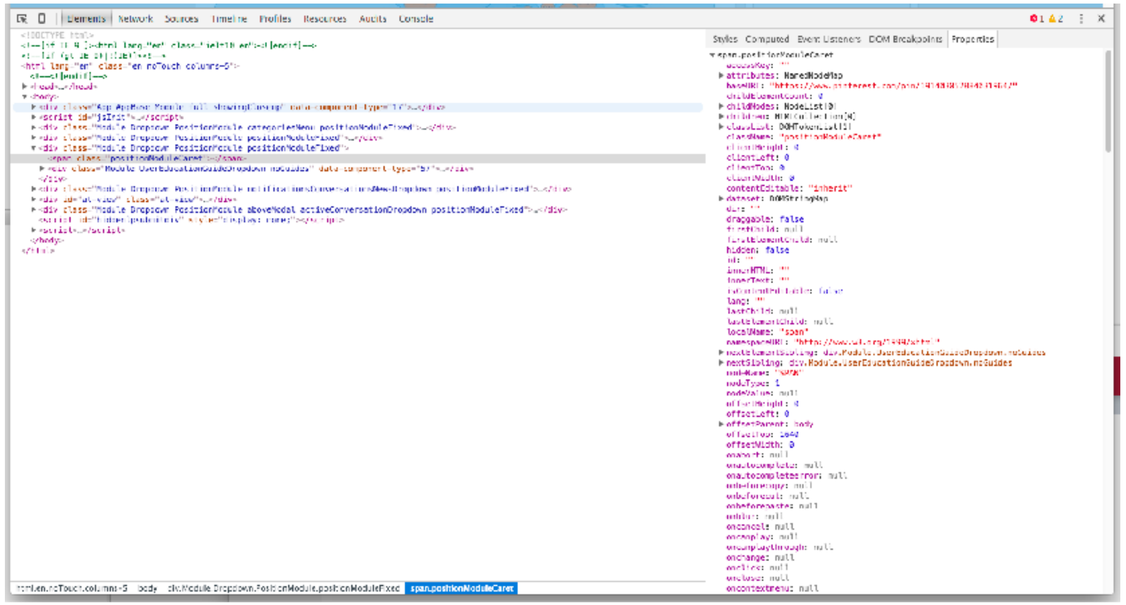

lastChild
firstChild
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild


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

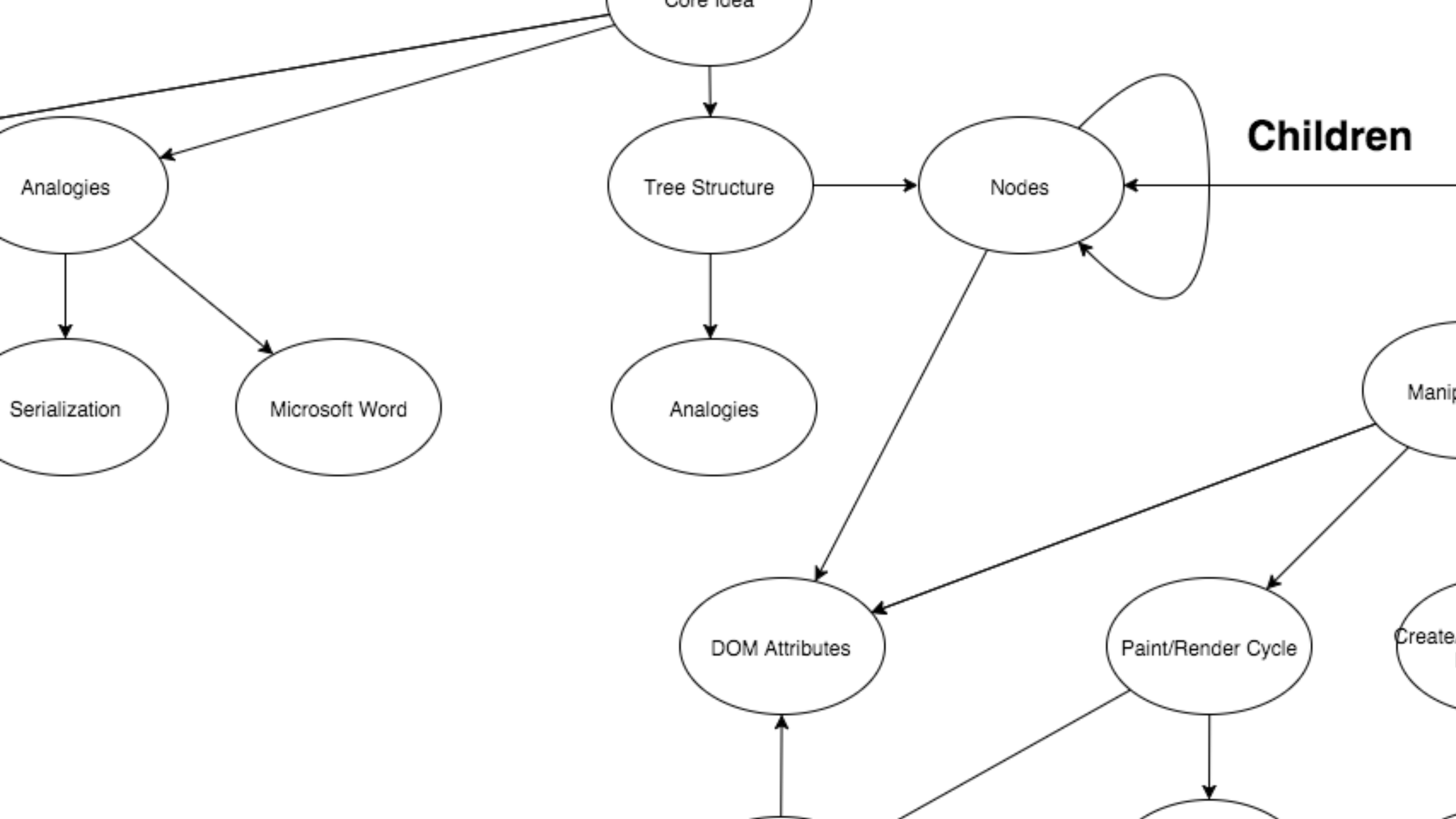
Nodes have lots of Attributes

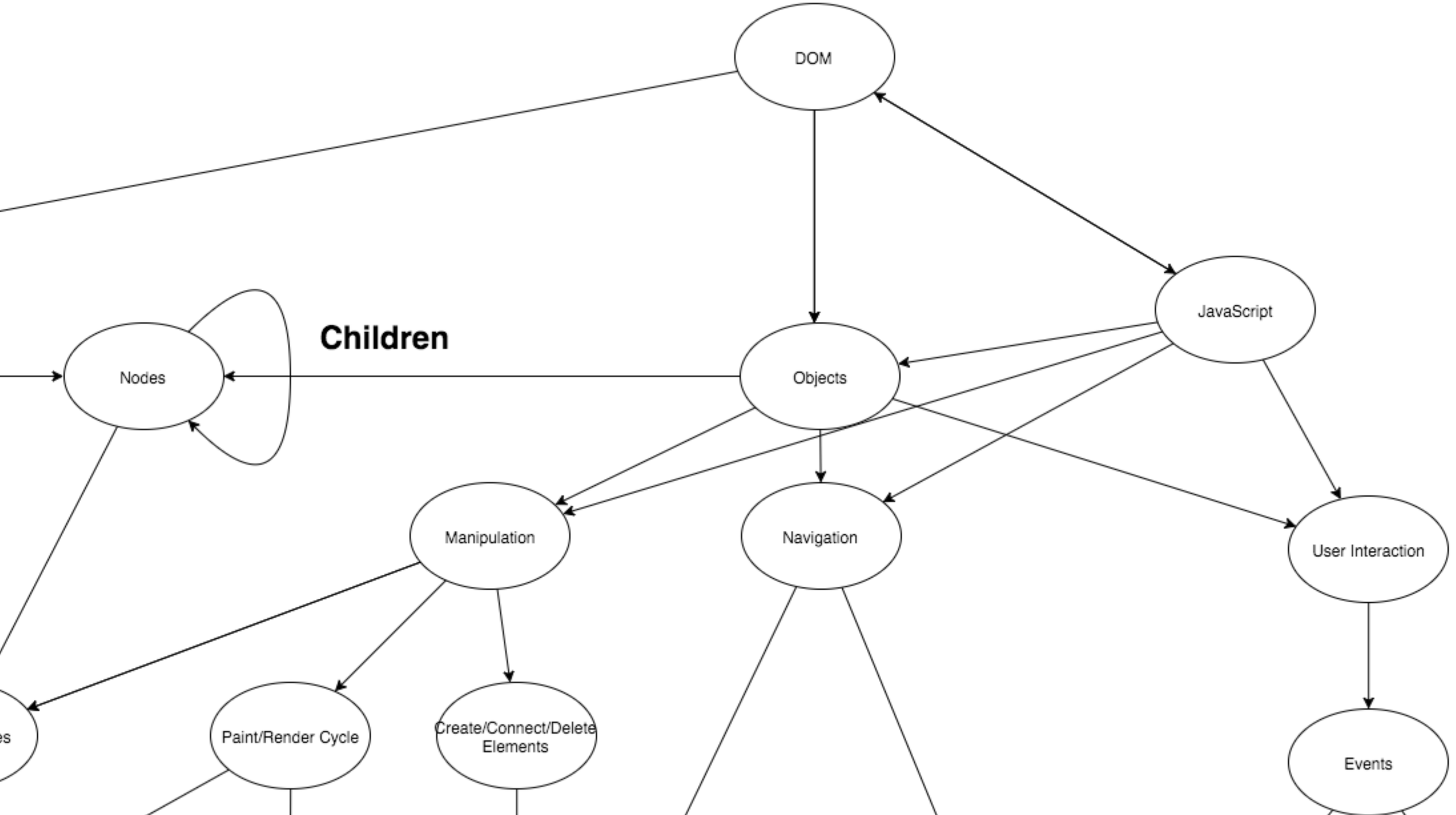
- Nodes are JavaScript Objects
- Nodes have Attributes that are JavaScript properties
- Attributes define how the Node looks and responds to User activity



one node

hundreds of properties!


Core Idea
Analogies
Tree Structure
Nodes
Children
Serialization
Microsoft Word
Analogies
DOM Attributes
Paint/Render Cycle

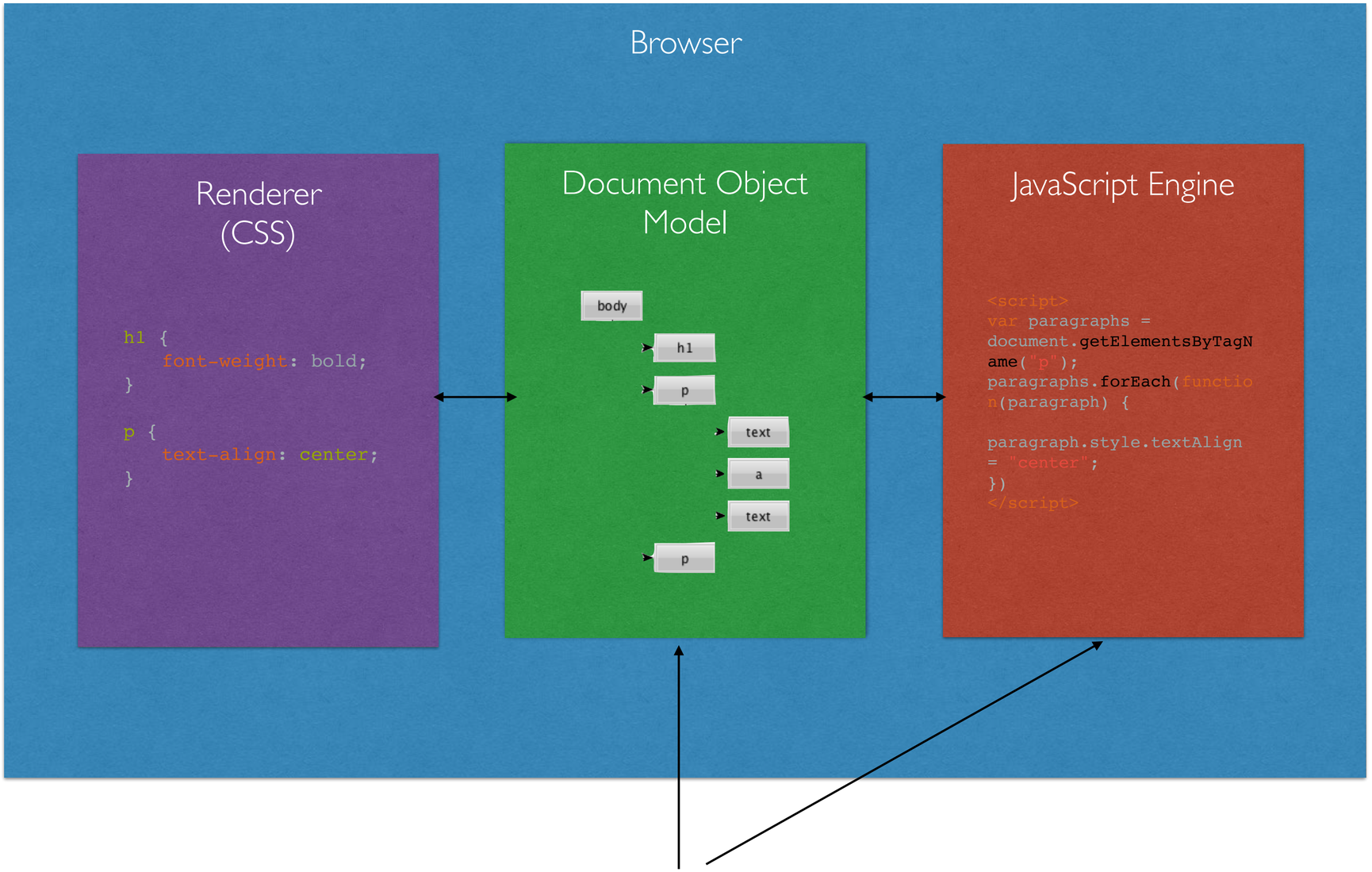

DOM
JavaScript
Children
Nodes
Objects
Manipulation
Navigation
User Interaction
Paint/Render Cycle
Create/Connect/Delete Elements
Events

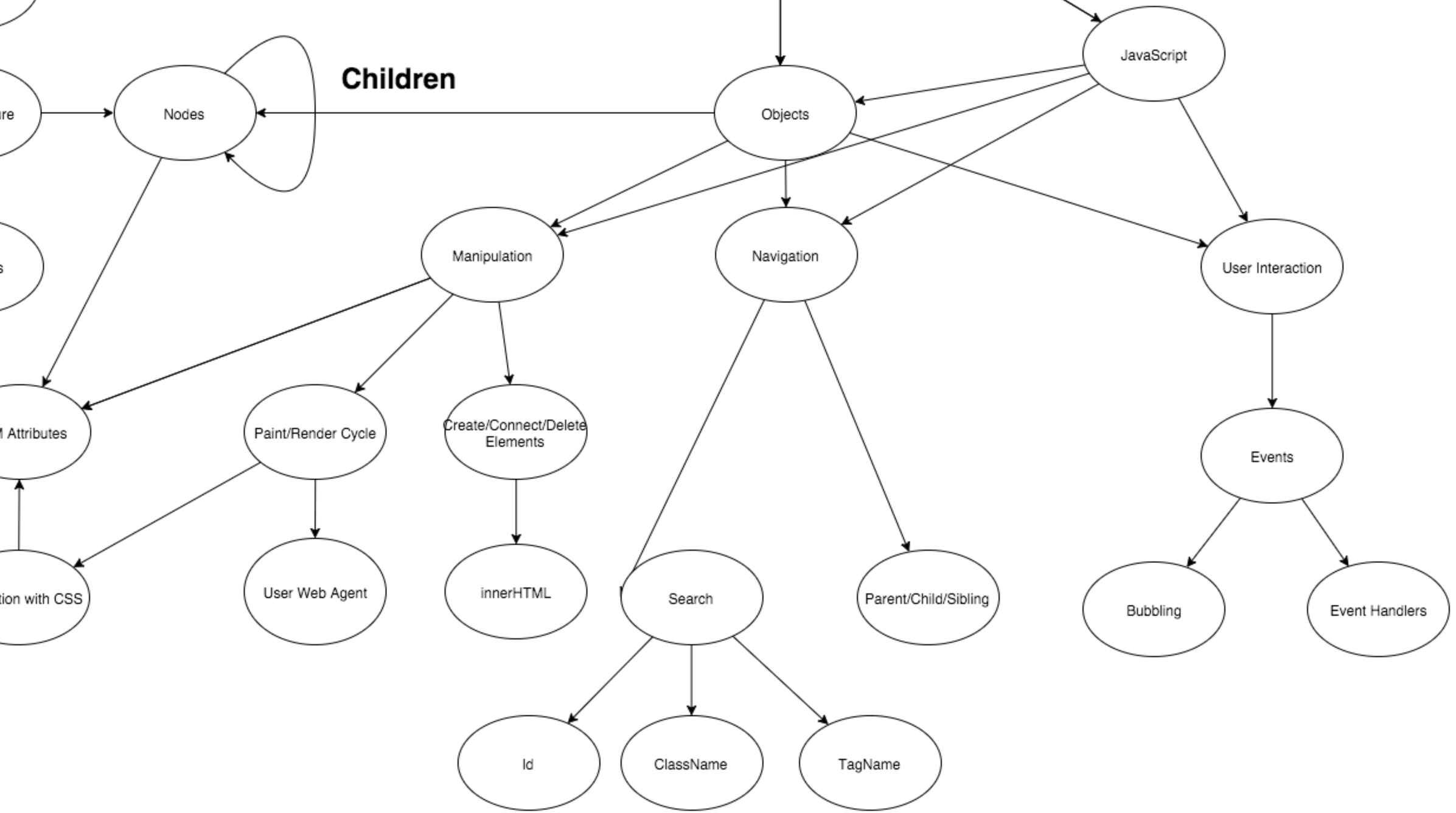
JavaScript 💚 DOM

- **We have this Object Model of an HTML document, how do we manipulate 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


Browser
Renderer
(CSS)
h1 {
font-weight: bold;
}
p {
text-align: center;
}
Document Object
Model
body
h1
p
text
a
text
p
JavaScript Engine
<script>
var paragraphs =
document.getElementsByTagN
ame("p");
paragraphs.forEach(functio
n(paragraph) {
paragraph.style.textAlign
= "center";
})
</script>
the document object is what connects these two things


Children
Nodes
Objects
JavaScript
Manipulation
Navigation
User Interaction
Paint/Render Cycle
Create/Connect/Delete Elements
User Web Agent
innerHTML
Search
Parent/Child/Sibling
Events
Bubbling
Event Handlers
Id
ClassName
TagName

Navigating the DOM

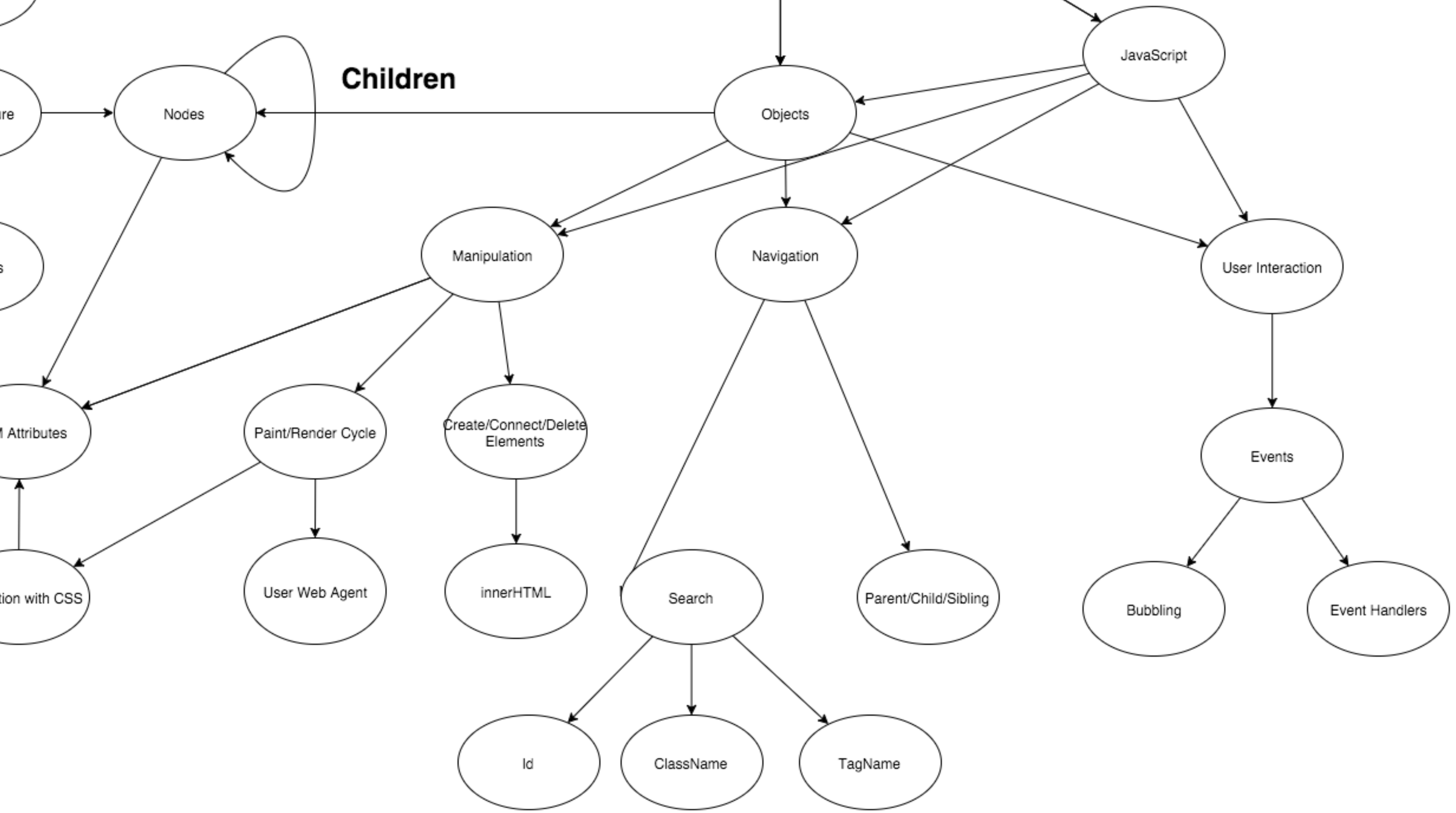
- https://jsbin.com/jijuweh/edit?html,js,output

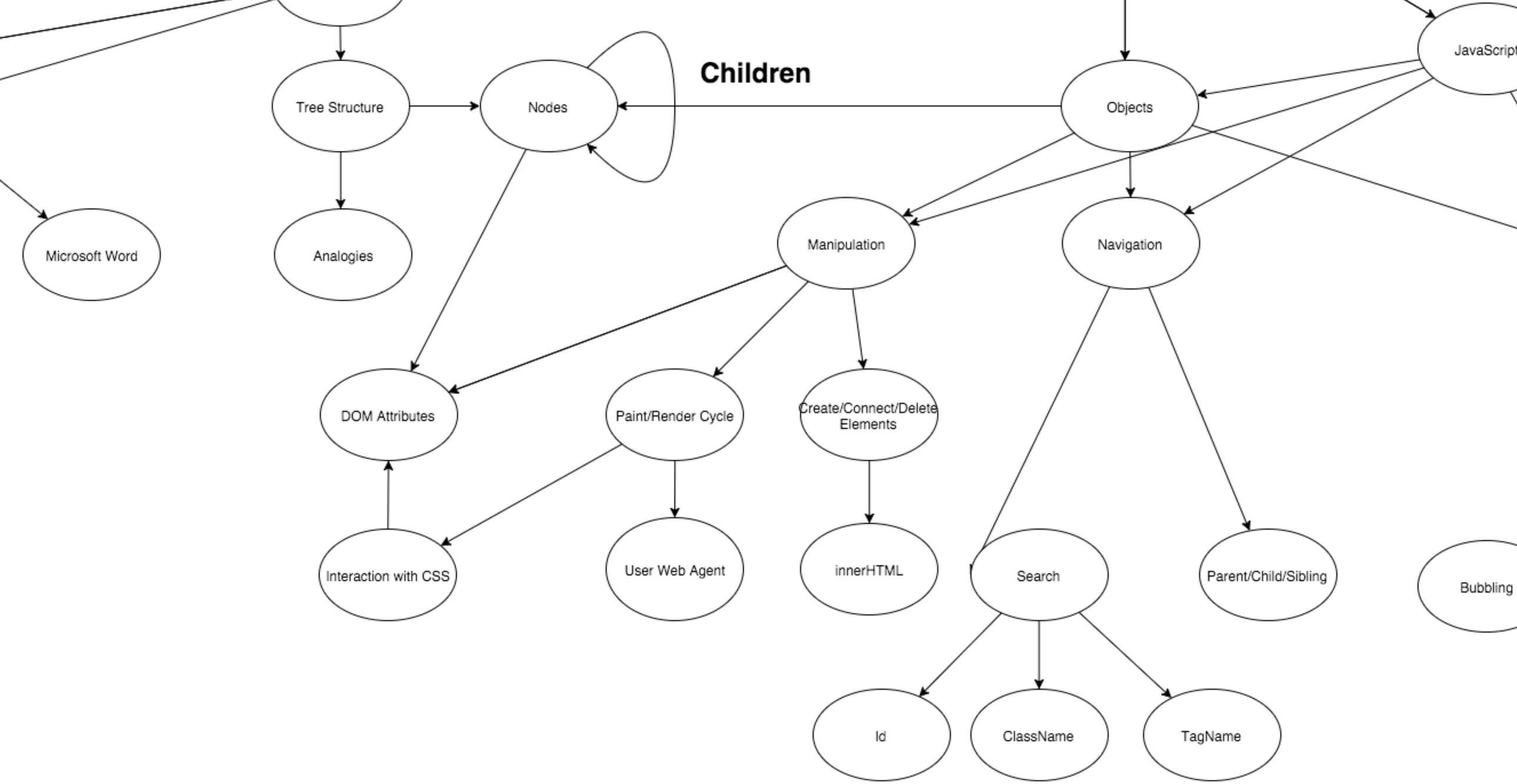
Navigating the DOM

- **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.querySelectorAll("#will");`

Traversing the DOM

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


Children
JavaScript
Nodes
Objects
Manipulation
Navigation
User Interaction
Attributes
Paint/Render Cycle
Create/Connect/Delete Elements
Events
with CSS
User Web Agent
innerHTML
Search
Parent/Child/Sibling
Bubbling
Event Handlers
Id
ClassName
TagName


Children
Tree Structure
Nodes
Objects
JavaScript
Microsoft Word
Analogies
Manipulation
Navigation
DOM Attributes
Paint/Render Cycle
Create/Connect/Delete Elements
Interaction with CSS
User Web Agent
innerHTML
Search
Parent/Child/Sibling
Bubbling
Id
ClassName
TagName

Manipulating the DOM

- Changing Attributes for Style
- Making Elements
- Putting them into the DOM
- Remove Elements
- innerHTML and the DOM HTML Reader

Changing style attributes

```
element.style.backgroundColor = “blue”;
```

- **CSS**
  - background-color
  - border-radius
  - font-size
  - list-style-type
  - word-spacing
  - z-index
- **JavaScript**
  - backgroundColor
  - borderRadius
  - fontSize
  - listStyleType
  - wordSpacing
  - zIndex

Changing CSS Classes

- *classList* is HTML5 way to modify which classes are on a Node

```
document.getElementById("MyElement").classList.add('class');

document.getElementById("MyElement").classList.remove('class');

if ( document.getElementById("MyElement").classList.contains('class') )

document.getElementById("MyElement").classList.toggle('class');
```

Creating Elements

- Create an element
  - `document.createElement(tagName)`
- Duplicate an existing node
  - `node.cloneNode()`

- Nodes are just free floating, not connected to the document itself, until you *link* them to the DOM.

Adding elements to the DOM

- Insert newNode at end of current node
  - `node.appendChild(newNode);`
- Insert newNode at end of current node
  - `node.prependChild(newNode);`
- Insert newNode before a certain childNode
  - `node.insertBefore(newNode, sibling);`

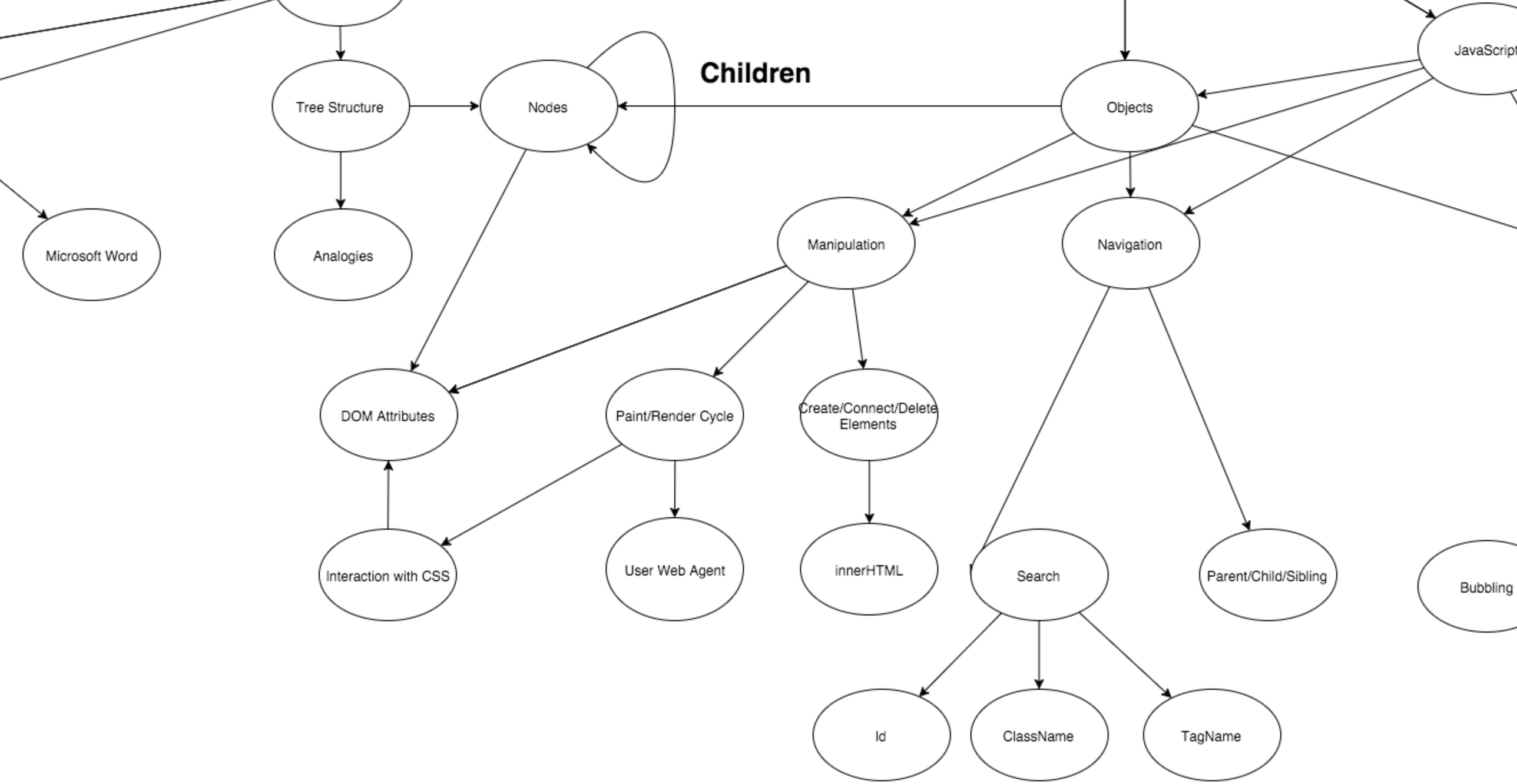
Removing Elements

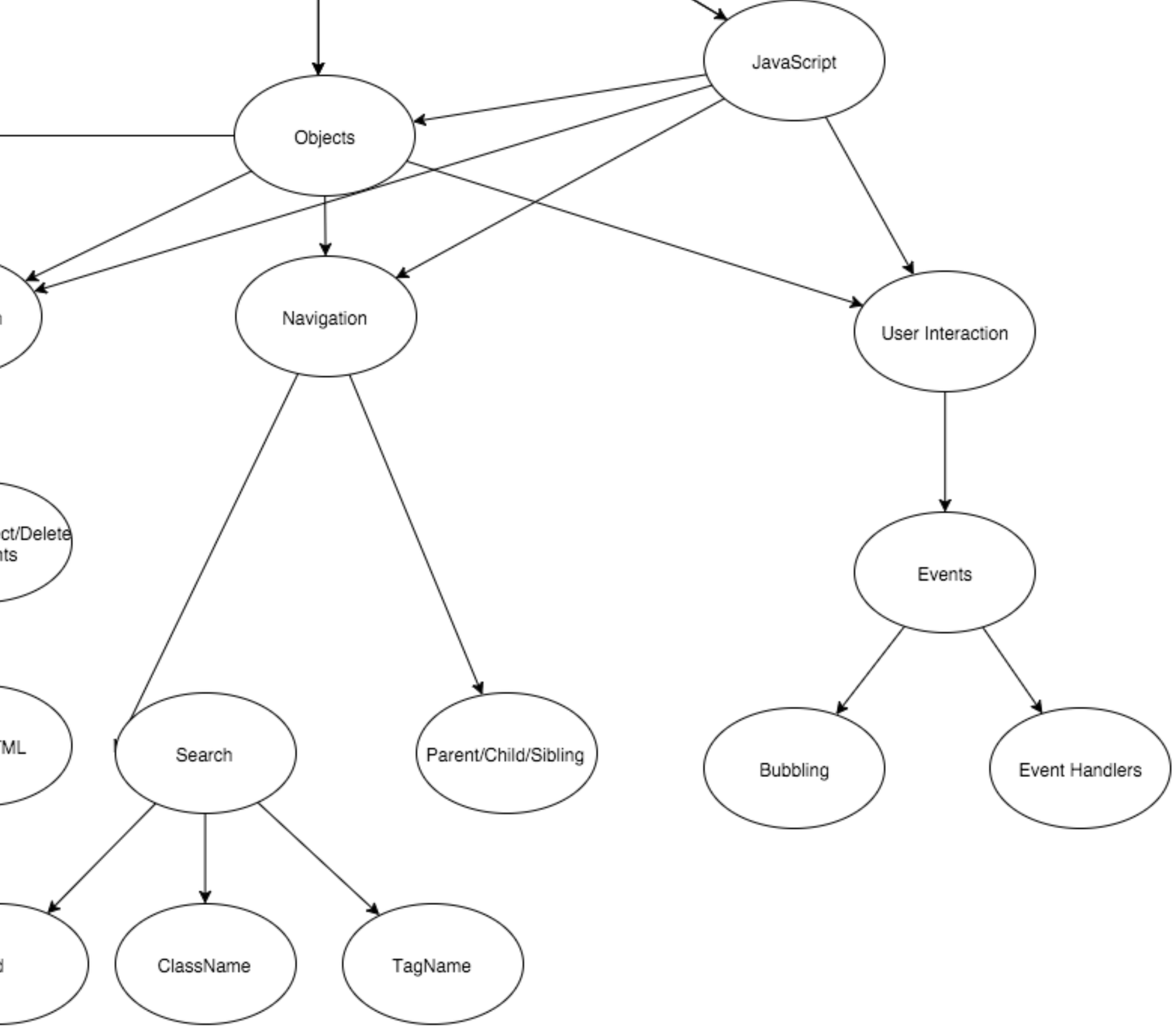
- **Removes the oldNode child.**
  - `node.removeChild(oldNode);`

- **Quick hack:**
  - oldNode.parentNode.removeChild(oldNode);

Using innerHTML

- What if you have a string of HTML and want to use the browser's HTML renderer instead of creating all the nodes by hand?
- You can use innerHTML
  - `element.innerHTML = '<ul><li>First</li><li>Second</li></ul>';`
- Reads a string of HTML and replaces the children of current Node with the parsed HTML


Children
Tree Structure
Nodes
Objects
JavaScript
Microsoft Word
Analogies
Manipulation
Navigation
DOM Attributes
Paint/Render Cycle
Create/Connect/Delete Elements
Interaction with CSS
User Web Agent
innerHTML
Search
Parent/Child/Sibling
Bubbling
Id
ClassName
TagName


JavaScript
Objects
Navigation
User Interaction
Events
Bubbling
Event Handlers
Search
Parent/Child/Sibling
ClassName
TagName

Responding to User Activity

- Event Handlers
- Default Events
- Bubbling and Propagation of Events

https://jsbin.com/sarohuyivu/1/edit?html,js,output

Event Handlers

```
element.addEventListener('click', function(event) {
    // Run this code on click
});
```

- JS that handles things that happen in the DOM
- Event examples:
  - click
  - (form) submit
  - hover
  - mouseover

Preventing Default Events

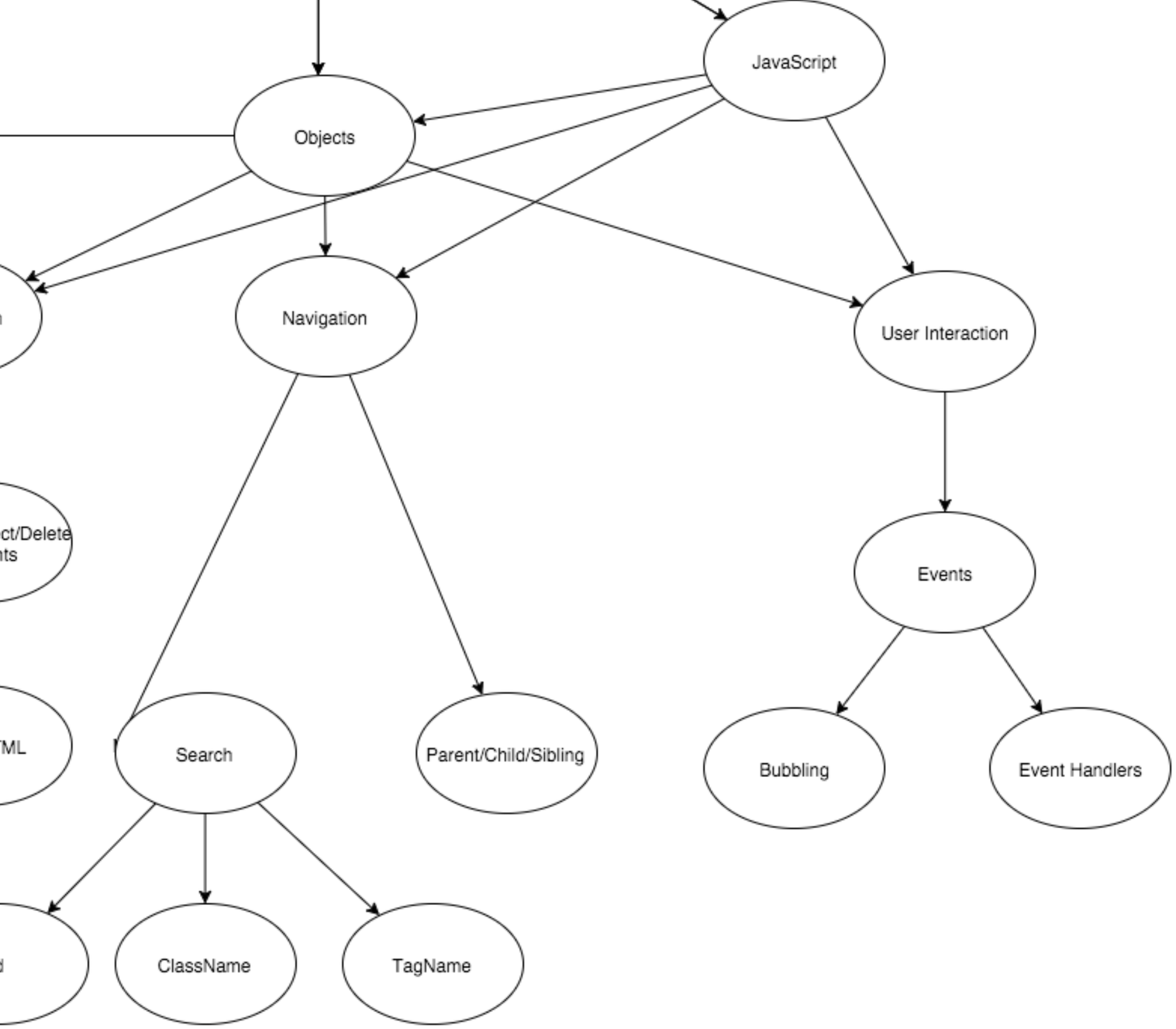
- **Just like the DOM has default styling, it also has default events**
  - Forms submit to the server
  - Clicks take you to new URLs
- **You can prevent these default events from occurring:**

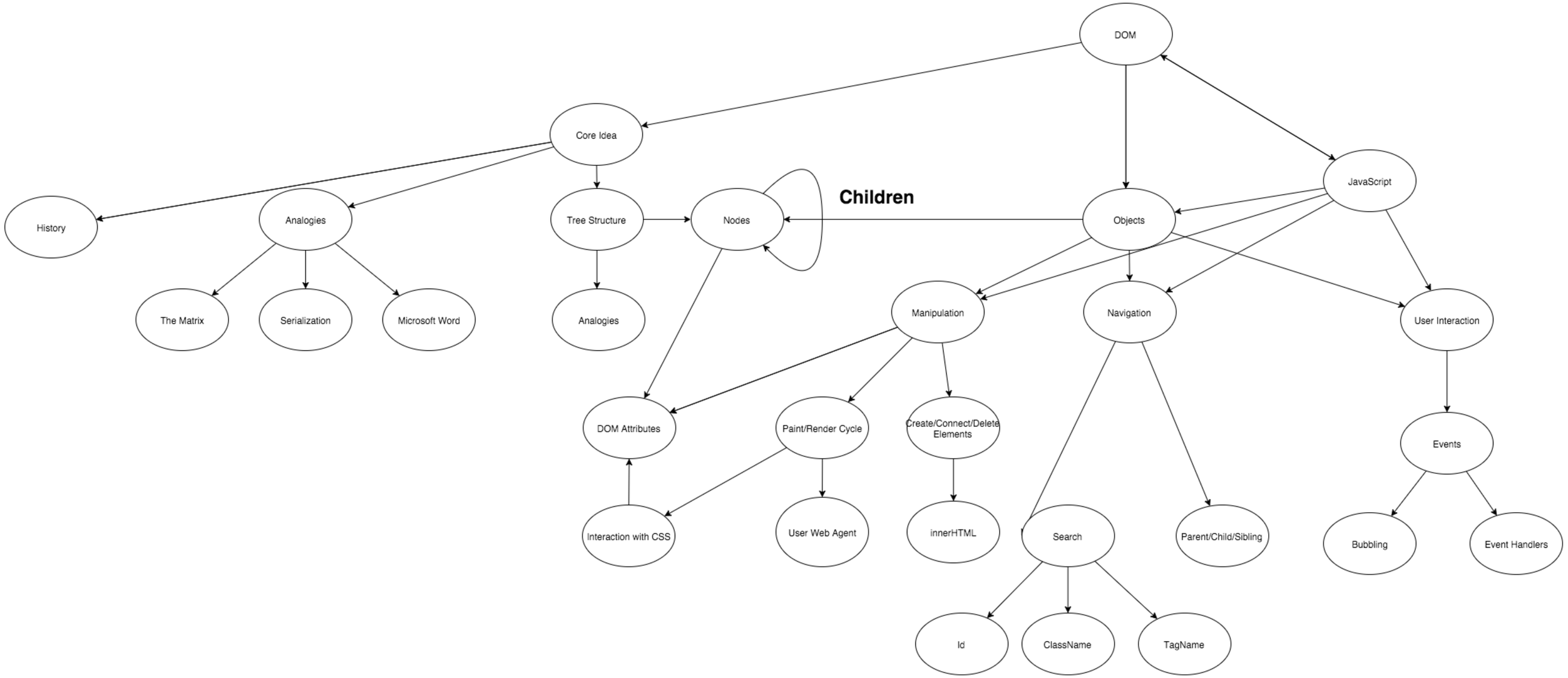
```
element.addEventListener('click', function(event) {
    event.preventDefault();
    // the rest of your code
});
```

Preventing Event Bubbling

- Events also fire at all the parent nodes of an element
- Why?
  - Performance: Don't have to create 100 event listeners, just create one at the parent Node
- To prevent events from “bubbling” up:

```
element.addEventListener('click', function(event) {
    event.stopPropagation();
    // the rest of your code
});
```


JavaScript
Objects
Navigation
User Interaction
Events
Bubbling
Event Handlers
Search
Parent/Child/Sibling
ClassName
TagName


DOM
Core Idea
JavaScript
History
Analogies
Tree Structure
Nodes
Children
Objects
The Matrix
Serialization
Microsoft Word
Analogies
Manipulation
Navigation
User Interaction
DOM Attributes
Paint/Render Cycle
Create/Connect/Delete Elements
Events
Interaction with CSS
User Web Agent
innerHTML
Search
Parent/Child/Sibling
Bubbling
Event Handlers
Id
ClassName
TagName