HTML

- HTML can be thought of as an extension to SGML, but more applicable for browsers
- HTML documents are a tree of nodes, containing text and possibly attributes for each node
    - ```
      <html>
          <head>
          </head>
          <body>
              <p aligned="right> Hello </p>
          </body>
      </html>
      ```
        - The `<p>` node (or element) has an attribute "aligned" with text content "Hello"
- Elements are enclosed in angled brackets and a tag specifying the type of element <>
    - Most (but not all) HTML elements also requiring a closing tag
        - i.e. `<html> ... </html>`
    - Elements that don't require a closing tag are known as void elements
        - i.e. `<meta charset="UTF-8">`
- A raw text element is an element that can contain only text
    - i.e. `<verbatim> ... \</verbatim>`
    - Everything else is considered a normal element (it can have subelements)
- The syntax for a HTML document is supplied by a Document Type Definition (DTD)
    - At the top of a typical HTML file, you will encounter a tag such as `<!DOCTYPE html>`, which specifies a Document Type Definition of HTML5
    - As browsers evolved, newer Document Type Definitions emerged for new needs (like for videos)
    - Document Type Definitions act as protocols that allow for portability such that your HTML document can work on Chrome, Mozilla, etc.
        - Though, as time has passed, the bureaucracy behind standardizing HTML via Document Type Definitions has faded and instead has been replaced by HTML5, which is effectively "evolving" via a Github repository containing the standards for HTML5
- HTML5 not only covers the text and syntax of an HTML document, but it also covers the intended meaning of document elements
    - i.e. HTML5 will interpret `<p> ... </p>` as a paragraph so that each web browser (Chrome, Mozilla) can format a paragraph how they want to
        - This allows for different browsers to have more "freedom" when interpreting and displaying an HTML document
- HTML5 also standardizes the Document Object Model (DOM), which acts as an interface for programs that wish to manipulate an HTML document (Javascript)
    - The DOM acts as a way to access the tree representing an HTML document in an object-oriented program
        - It effectly treats each tree node as an "object" such that the programming language can manipulate it accordingly (change its text content, add child nodes to it, etc.)
      ```

CSS

- Javascript has a DOM binding, but it can a bit difficult to work with due to the nature of Javascript - debugging can be a nightmare. To easily work with the presentation and layout of a HTML page, you should use CSS (Cascading Stylesheets)
- Like HTML, CSS involves declarative (no programming constructs) specifications for how to present HTML elements
- Via CSS, you specify how to style elements for certain subtrees
- CSS styles are first adapted to via the web page, then by the user's preferences, and then finally by the browser configuration

Javascript

- Javascript is a simple, easily interpreted language. It is most commonly used to "hook" into HTML, as with DOM manipulation.
    - i.e. `<script src ="myscript.js"></script>` will load the Javascript file "myscript.js" (which is relative to the location of the HTML file in the directory)
        - Scripts can also be put inline (though avoid doing this for big scripts)
- DOM manipulation with pure Javascript can require a lot of code, which leads to a more error-prone process, which is where libraries such as Node and React come into play
    - React makes use of JSX (Javascript Extension) to simplify DOM generation (though you may still need traditional Javascript DOM manipulation to manipulate existing DOM elements)
        - JSX is translated into plain Javascript during the compilation phase using transpilers (i.e. Babel)

    ```
    const language = "en";
    const class = "CS35L";
    const n = 3;
    const header = <h1 lang={language}> {class} assignment {n + 1} </h1>
    ```

        - This simplifies a bunch of DOM calls in Javascript to create an h1 element with the appropriate attributes and text content

JSON

- Javascript Object Notation is a standard way to serialize data via client-server communications

    ```
    { "menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem" : [
                {"value": "New", "onClick": "CreateNewDoc()"},
    ```

```
                {"value": "Close", "onClick": "CloseDoc()"}
            ]
        }
    }
```

- JSON is effectively another way to represent a HTML node, but it is preferred because it is easier to parse (many languages have built-in parsers for JSON)
    - You may encounter the alternate of XML instead of JSON

Browser Rendering Pipeline

- The browser will receive an HTML document and then actually *render* the contents of this HTML document in a way that is presentable
    - This used to involve first parsing the HTML, and then parsing the CSS, and then parsing the Javascript, but this sequential approach is slow
    - To optimize this process, browsers start rendering before it has all of the information about the webpage (typically the `<head>` element arrives first, allowing for the browser to setup important meta information)
    - Optimizations:
        - Guess whether content is rendered on the screen or not (needing to scroll) and if it is not, then skip and save rendering it for later
            - Don't assume that all the Javascript code will be executed right away (order of execution is not necessarily what you think it is)
        - Decide if an element is low priority and defer their rendering for later
        - Guess the overall layout of the HTML document and render based on the guess; if the guess is wrong, re-render