FRESNO STATE
Discovery. Diversity. Distinction.

Web Programming (CSci 130)

Department of Computer Science
College of Science and Mathematics
California State University Fresno
H. Cecotti

Learning outcomes

- Warning: this class is very important
  - From coding "by hand" HTML, to generate HTML content dynamically
  - Bridging the gap between HTML & CSS and JavaScript
  - To be used until the end of the semester and beyond
- In this class, you will learn about the DOM
  - **D**ocument **O**bject **M**odel
- Web programming
  - Creation of webpages (HTML+CSS+Javascript)
  - Think about the browser
    - The HTML file must be parsed, decoded, and presented on the screen
    - How to decode this file so
      - It stays consistent across browsers
    - → Need of a data structure to contain the webpage downloaded on the server

Introduction

- JavaScript
  - Initially for web browsers
  - Evolution to other platforms
    - Browser
    - Web server
    - Another host…
    - → host environment
  - Host environment
    - Platform specific objects + functions
      - Web browsers → control web pages
      - Node.JS → server-side functions

Introduction

- Window (root object) : access methods, properties of the window object
  - let h=window.innerHeight; // in pixels
  - let w=window.innerWidth; // in pixels
- **DOM** (Document Object Model)
  - Document → access to the page content
  - document.body.style.background = ‘blue';
  - 2 standards: W3C and WhatWC ☺
- **BOM** (Browser Object Model)
  - Objects provided by the browser
    - Host environment
    - Example of objects: navigator, location
  - Navigator, screen, frames, history,
- JS (Javascript)
  - Object, array, functions, …

DOM

- Document Object Model (**DOM**)
  - Programming interface for HTML and XML documents
  - It represents the page
    - Programs can change the document structure, style, and content
  - It represents the document as nodes and objects
  - Object Oriented representation of the web page
    - To be modified with a language such as JavaScript (JS)
- Web page document
  - To be displayed in the browser window
  - As the HTML source

DOM

- Different versions
  - DOM level 1, DOM level 2, DOM level 3, … → DOM
- DOM specification
  - Tells the structure of a document
  - Gives objects to manipulate the document
- DOM: every HTML tag is an object
  - Nested tags: children
  - Text inside a tag: an object
  - → All the objects can be accessed through Javascript

D, O, and M of DOM

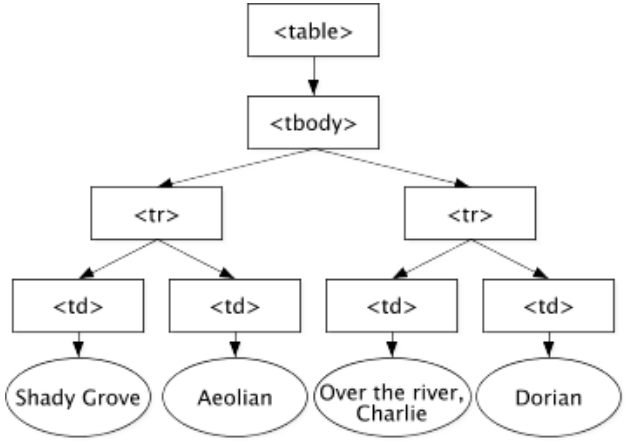
- **D**: Document
  - We go back to the notion of document
    - Content + Structure (hierarchy/organization) + Semantic (meaning of blocks)
  - What happens when you load an HTML file?
    - That's the job of the browser (it is also web programming)
  - Conversion of the file that is loaded
    - From an HTML file to an Object
- **O**: Objects
  - With properties
  - With methods
- **M**: Model
  - Definition of what is a document
    - See week 1 (XML)

HTML DOM

- Representation of the document as a tree
  - Notion of node
    - The document is a document: a node
    - All the different HTML elements: nodes
    - All the attributes: attribute nodes
    - Piece of text in the HTML: text nodes
    - Comments: comment nodes

DOM Tree structure

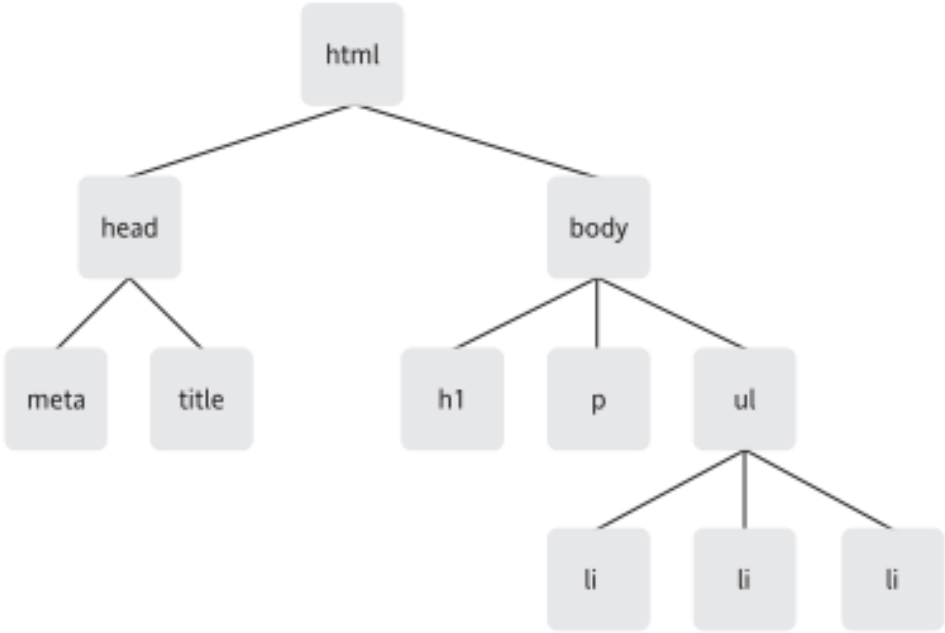
- Tags
  - Element nodes (elements)
- Tree of elements
  - Html = root
    - (always the top tag, even if absent from the HTML file)
  - Head and Body = two children of the root …
  - Text inside elements
    - Text nodes: #text
  - Comments in HTML
    - #comment → part of the DOM!
  - Tables have always a <tbody>



DOM Tree structure

- Example
  - ➢ Elements tree of a basic webpage

- Objects
  - ➢ A paragraph in an HTML page
    - ○ You can see it as an object instance of the class paragraph
      - Attributes = properties of the object
      - Values = values assigned to the properties of the object
      - + additional functions and properties
    - ○ Class parent of the paragraph
      - ‘’An HTML block’’ = node in the tree/document
        - Functions & methods for all the HTML blocks



DOM

- Once you access the document object
  - ➢Access to MANY properties/methods
    - Use the specifications of the DOM to know what you can access or not
    - Typical functions that you can obtain when you browse in a tree
    - Direct access to the main HTML blocks
      - You can modify
        - Direct update in the HTML page ☺
    - See the list of the methods (Canvas + many websites)
  - ➢Some elements will be seen during the next classes for special functions

Exploring the tree

- DOM
  - Allows you to reach the elements in the ``tree'' of HTML blocks
  - Start: document object
  - <html> = document.documentElement
  - <body> = document.body
  - <head> = document.head
- Children
  - childNodes
  - firstChild
  - lastChild
- Example

```
for (let i = 0; i < document.body.childNodes.length; i++) {
    alert( document.body.childNodes[i] ); // Text, DIV, Text, UL, ..., SCRIPT
  }
```

InnerHTML vs. textContent

- innerHTML
  - The Element property innerHTML gets or sets the HTML or XML markup contained within the element.
  - https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML
- textContent
  - The textContent property of the Node interface represents the text content of the node and its descendants.
  - https://developer.mozilla.org/en-US/docs/Web/API/Node/textContent
- Differences
  - **innerHTML** it parses content as HTML → it takes longer.
  - **nodeValue** it uses straight text, does not parse HTML, and is faster.
  - **textContent** it uses straight text, does not parse HTML, and is faster.
  - **innerText** it takes styles into consideration. It won't get hidden text for instance.

Different ways to create code dynamically

- Remember we deal with objects !!
  - Let o=document.getElementById(“myparagraph");
    - o is an object corresponding to the block with the id=“myparagraph“ in the HTML code
  - o.innerHTML= ‘<table><tr><td>A</td><td>B</td></tr></table>’;
    - Put whatever you need in the string using concatenation and the values from JS variables

Different ways to create code dynamically

```
table = document.createElement('TABLE');
o.appendChild(table);
let tableBody = document.createElement('TBODY');
table.appendChild(tableBody);
let tr = document.createElement('TR');
tableBody.appendChild(tr);
let td = document.createElement('TD');
// Solution 1:
// td.appendChild(document.createTextNode("blabla"));
// Solution 2:
// td.textContent="blabla";
// Solution 3:
td.innerHTML="blabla";
tr.appendChild(td);
```

Examples

- See files on Canvas
  - class_javascript_dom_01.html
  - class_javascript_dom_02.html
  - class_javascript_dom_03.html

- **Very important example**
  - **Midterm 1 – Final: You must able to create tables dynamically**
    - Final: retrieve data from a table in the database, then display the content of the table on the client side in an HTML page.
  - class_javascript_dom_dynatable.html + js file
    - Dynamic table + events

Conclusion

- **DOM**
  - ➢ Well … everything is a node ☺
    - Document, HTML elements, attributes, text, comments
  - ➢ Allows to get the document as an Object
    - To access its properties
    - To access its methods
- Programming
  - ➢ → We can do everything from JS to modify the tree
    - Creation of functions to automatize the creation of HTML pages generations