Document Object Model (DOM)

Mendel Rosenblum
Browser JavaScript interface to HTML document

- HTML document exposed as a collection of JavaScript objects and methods

  The **Document Object Model** (DOM)

- JavaScript can query or modify the HTML document

- Accessible via the JavaScript global scope, aliases:
  
  `window`

  `this` (When not using 'use strict';)
DOM hierarchy

- Rooted at `window.document`
- Follows HTML document structure
  - `window.document.head`
  - `window.document.body`
- DOM objects have tons (~250) of properties, most private

  Objects (representing elements, raw text, etc.) have a common set of properties and methods called a DOM "Node"
DOM Node properties and methods

- Identification
  
  `nodeName` property is element type (uppercase: P, DIV, etc.) or #text

- Encode document's hierarchical structure
  
  `parentNode`, `nextSibling`, `previousSibling`, `firstChild`, `lastChild`.

- Provide accessor and mutator methods
  
  E.g. `getAttribute`, `setAttribute` methods, etc..
Sample <b>bold</b> display
Accessing DOM Nodes

- Walk DOM hierarchy (not recommended)
  
  ```javascript
  element = document.body.firstChild.nextSibling.firstChild;
  element.setAttribute(...)
  ```

- Use DOM lookup method. An example using ids:

  ```html
  HTML: <div id="div42">...</div>
  ```

  ```javascript
  element = document.getElementById("div42");
  element.setAttribute(...)
  ```

- Many: `getElementsByTagName()`, `getElementsByClassName()`, ...
  
  - Can start lookup at any element:
    
    ```javascript
    document.body.firstChild.getElementsByTagName()
    ```
More commonly used Node properties/methods

- `textContent` - text content of a node and its descendants
  Previous slide example: P Node textContext is "Sample bold display"

- `innerHTML` - HTML syntax describing the element's descendants.
  Previous slide example: P Node innerHTML is "Sample <b>bold</b> display"

- `outerHTML` - similar but includes element "<p>Sample <b>bold</b> display</p>

- `getAttribute()`/`setAttribute()` - Get or set the attribute of an element
Common DOM mutating operations

- Change the content of an element
  
  ```javascript
  element.innerHTML = "This text is <i>important</i>";
  ```
  
  Replaces content but retains attributes. DOM Node structure updated.

- Change an `<img` tag `src` attribute (e.g. toggle appearance on click)
  
  ```javascript
  img.src="newImage.jpg";
  ```

- Make element visible or invisible (e.g., for expandable sections, modals)
  
  ```javascript
  Invisible:  element.style.display = "none";
  Visible:    element.style.display = "";
  ```
DOM and CSS interactions

- Can update an element's class

  ```javascript
  element.className = "active";
  ```

- Can update element's style

  ```javascript
  element.style.color = "#ff0000";  // Not preferred way!
  ```

- Can also query DOM by CSS selector

  ```javascript
  document.querySelector() and document.querySelectorAll()
  ```
Changing the Node structure

- Create a new element (can also cloneNode() an existing one)
  
  ```javascript
  element = document.createElement("P");
  // or
  element = document.createTextNode("My Text");
  ```

- Add it to an existing one
  
  ```javascript
  parent.appendChild(element);
  // or
  parent.insertBefore(element, sibling);
  ```

- Can also remove an Nodes: `node.removeChild(oldNode)`;

- But, setting innerHTML can be simpler and more efficient.
More DOM operations

● Redirect to a new page

```javascript
window.location.href = "newPage.html";
```

Note: Can result in JavaScript script execution termination

● Communicating with the user

```javascript
console.log(" Reached point A" );  // Message to browser log
alert("Wow!"); confirm("OK?");  // Popup dialog
```
DOM's Coordinate System

- The screen origin is at the upper left; $y$ increases as you go down
- The position of an element is determined by the upper-left outside corner of its margin
- Read location with `element.offsetLeft, element.offsetTop`
- Coordinates are relative to `element.offsetParent`, which is not necessarily the same as `element.parentNode`
DOM Coordinates

```
<div class="div1"><div class="div2"><div class="div3"></div></div></div>
```
Positioning elements

- Normally elements are positioned automatically by the browser as part of the document.

- To pull an element out of the document flow and position it explicitly:

  ```javascript
  element.style.position = "absolute"; // anything but "static"
  element.style.left = "40px";
  element.style.top = "10px";
  
  "absolute" - the element no longer occupies space in the document flow.
  ```

- The origin inside an offsetParent (for positioning descendants) is just inside the upper-left corner of its border.
Positioning context

- Each element has an offsetParent (some ancestor element).
- When an element is positioned, coordinates such as element.style.left are relative to its offsetParent.
- Default offsetParent is the <body> element.
- Some elements define a new positioning context:
  - position CSS attribute is absolute (element is explicitly positioned)
  - position CSS attribute is relative (element is positioned automatically by the browser in the usual way)
  - This element will become the offsetParent for all its descendents (unless overridden by another positioning context)
Positioning Children

- Parent margin
- Parent border
- Parent padding

Child margin
Child border

top/offsetTop
left/offsetLeft
Element dimensions

- Reading dimensions: `element.offsetWidth` and `element.offsetHeight`
  Include contents, padding, border, but not margin

- Updating dimensions: `element.style.width` and `element.style.height`
Positioning

```html
<body>
    <div id="div1">
        <p>div1</p>
    </div>
</body>

#div1 {
    width: 50px;
    height: 200px;
    background: #ffe0e0;
}
```
Positioning

...<div id="div2">
   <p>div2</p>
   <div id="div2-1">
      <p>div2-1</p>
   </div>
</div>
#div2 {width: 300px; height: 200px; position: relative; background: #d0e0ff;}
#div2-1 {width: 150px; height: 80px; position: absolute; top: 50px; left: 100px; background: #d0e0ff;}
Positioning

...  
  <div id="div3">
    <p>div3</p>
    <div id="div3-1">
      <p>div3-1</p>
    </div>
  </div>
#div3 {width: 300px; height: 200px; background: #ffffe0;}

#div3-1 {width: 150px; height: 80px; position: absolute; top: 50px; left: 100px; background: #ffffe0;