What is Dynamic HTML?
- Generally defined as web pages reacting to user actions without server interaction
- There is no single standard
  - Cascading Style Sheets (CSS) allow fine control over how html elements are positioned on page
  - JavaScript code runs inside the browser
    - Triggered by events generated by the browser
    - Document Object Model (DOM) describes how the document object should be manipulated
- Widespread support, incompatibilities remain

Overview of lecture
- CSS
- DOM
- Dynamic web pages
How stylesheets work

- A style sheet is a collection of rules
- Rules consist of two parts
  - References to one or more HTML elements (selectors)
  - Rules also apply to elements inside those referenced
  - One or more style sheet attributes applied to them
  - For each attribute, the value assigned to it
- Multiple rules can refer to the same element, the cascading preference gives their priority

```html
p {color:#0000ff;font-size:14pt}
a,h3 {color:green}
```

Further CSS selectors

<table>
<thead>
<tr>
<th>Name</th>
<th>HTML</th>
<th>CSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class selector</td>
<td>&lt;p class=&quot;big&quot;&gt; &lt;h2 class=&quot;hot&quot;&gt;</td>
<td>p.big{font-size:large} .hot{color:red}</td>
</tr>
<tr>
<td>ID selector</td>
<td>&lt;p id=&quot;special&quot;&gt;</td>
<td>#special{border:ridge red} p#special{…}</td>
</tr>
<tr>
<td>Descendant selector</td>
<td>&lt;p&gt;&lt;em&gt;This&lt;/em&gt;</td>
<td>p em{background-color:blue}</td>
</tr>
<tr>
<td>Pseudoclasses</td>
<td>&lt;a href=”…”&gt;</td>
<td>:link,:active,:visited :hover</td>
</tr>
</tbody>
</table>

- Many HTML elements can use the same class, but the value of the id attribute must be unique
- Pseudoclasses identify various states of elements: visited links, the mouse hovering over a paragraph

Adding style sheets to pages

- Rules inside the `<style>` tag
  - Place it inside `<head>`
  - For safety use `<!-- -->`
  - Can read external files
    - `@import url("coolstyle.css");`
  - Must precede other rules
- The `style` attribute of HTML tags
  - Selectors not needed, applies to HTML element
    - `<p style="color:yellow">`
  - Don’t use it, mixes content and presentation
Some CSS attributes

- **font-color** specifies the color of text
  - Six digit hexadecimal RGB representation
  - Names such as red, green, yellow, darkred, etc
- **font-size** specifies absolute or relative font size
  - In various units, percentages, symbolically
  - 12pt, x-small, medium, larger, 150%
- **font** can specify font size, color and other characteristics with a single attribute
- Other attributes: `text-align`, `vertical-align`, `background-color`, `background-image`

Size units used in CSS

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>em</td>
<td>Element’s font’s height</td>
</tr>
<tr>
<td>ex</td>
<td>The height of a lower case x</td>
</tr>
<tr>
<td>px</td>
<td>Pixel (exact size depends on display)</td>
</tr>
<tr>
<td>in</td>
<td>Inch</td>
</tr>
<tr>
<td>cm</td>
<td>Centimeter</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeter</td>
</tr>
<tr>
<td>pt</td>
<td>Point = inch/72</td>
</tr>
<tr>
<td>pc</td>
<td>Pica = 12 points</td>
</tr>
</tbody>
</table>

Borders, margins, padding

- `margin` space (transparent)
- `border` space
  - The content of the HTML element goes inside the content space. For, say a `<p>` tag, the content is the actual text of the paragraph.
More CSS attributes

- **height** and **width** can specify the absolute or relative size for element (only content space)
- **border-style** can be solid, dashed, dotted, double, groove, ridge, inset, outset, none
- **border-width**, **border-color**
- **border** gives width, style, and color (in this order)
- **margin** and **padding** specify width only
- Other attributes: margin-bottom, border-top, border-right-color, border-left-style

Two useful HTML tags

- **<div></div>** and **<span></span>** can enclose other HTML elements
  - Used as containers that pass on their CSS attributes to elements inside
  - **<div>** defines a paragraph-like rectangular area
  - **<span>** can apply to any small piece of text

```html
<head>
  <style type="text/css">/
  p{color:#0000ff;font-size:14pt}
  a,h2{color:green}
  .big{font-size:x-large}
  .halfpage{width:50%;border:double}
  #special{border:ridge red;width:40%}
  p em{background-color:red}
  p:hover{background-color:yellow}
  .halfpage p[id="special"]
  </style>
</head>
<body>
  Normal text with a `<a href="bucky.gif">link</a>`.
  Going crazy with `<em>colors and fonts</em>`
  This text is `<em>big</em>`
  This is a special paragraph.
  Another paragraph with a lot of text that shows us where text wraps.
</body>
```
Some CSS positioning

- `position` identifies rule as giving position
  - `absolute` – position is absolute
  - `fixed` – similar but ignores scrolling
  - Other values possible
- `top`, `left`, `right`, `bottom` specify offset
distance between element’s positioning context and element’s box (includes margins)

Overview of lecture

- CSS
- DOM
- Dynamic web pages

Document Object Model

- Describes how the `document` object can be traversed and modified
  - Represented as tree structure
- Two approaches in use
  - IE-specific more convenient for HTML
  - W3C more verbose, but also applies to XML
- DOM has levels 0-3 and many sub-standards
- The DOM interface used in other contexts with other languages (C++, Java, python, etc.)
The document as a tree

```html
<html>
<head>
<title>A Document</title>
</head>
<body>
<h1>A web page</h1>
<p>A simple paragraph</p>
</body>
</html>
```

Manipulating nodes

- Traversing the element tree
  - Each node has `childNodes` array
  - Can use properties `firstChild`, `lastChild`, `nextSibling`, `previousSibling`
  - Firefox’s DOMInspector visualizes the DOM tree
- `nodeType` property can be 1 (element), 2 (attribute), 3 (text), 8 (comment), 9 (document)
- Can change structure using `appendChild()`, `removeChild()`, `replaceChild()`, `insertBefore()`

Tag attributes

- Attribute nodes are ignored during traversal
- Elements have properties for attributes
  - Words capitalized – e.g. the body element has a `bgColor` property corresponding to the HTML `bgcolor` attribute
  - Can assign strings to these properties
  - Can also treat `style` attribute as an object with properties of its own
- Elements have methods `getAttribute()`, `setAttribute()`, `removeAttribute()`
More DOM manipulation

- The document object (and element objects) have methods for finding specific elements:
  - `getElementsByTagName()` returns an array with all elements with the given tag name
  - `getElementsByName()` returns an array with all elements with given name
  - `getElementById()` returns element with given ID
- To build new nodes, use the document object’s methods `createElement(tagName)` and `createTextNode(text)`
- Text node have `appendData()`, `insertData()`, `deleteData()`, `replaceData()` methods

Overview of lecture

- CSS
- DOM
- Dynamic web pages

JavaScript timers

- Used extensively in dynamic pages
- `setTimeout(code, delay)` tells browser to execute `code` in `delay` milliseconds
- If you save the return value, you can cancel using `clearTimeout(timeoutID)`
- `setInterval()` and `clearInterval()` work similarly, but code is run periodically instead of just once
Dynamic Colors

```javascript
function changeBGColor(color) {
    var p = document.getElementById("para1");
    p.style.backgroundColor = color;
}

function checkColor() {
    var s = document.getElementById("textfield1").value;
    if (s.length != 6) {
        alert("Must enter six hex digits");
        return;
    }
    for (var i = 0; i < 6; i++) {
        if (!(s[i] >= 'A' && s[i] <= 'F') ||
            (s[i] >= 'a' && s[i] <= 'f') ||
            (s[i] >= '0' && s[i] <= '9')) {
            alert("Character " + s[i] + " is not valid");
            return;
        }
    }
    changeBGColor("#" + s);
}
```

Animations Example

```javascript
var pos = 0;
function runAway() {
    var image = document.getElementById("bucky");
    if (pos == 0) {
        image.style.left = "250px";
        image.style.top = "50px";
        pos = 1; vpos = 50;
    } else {
        image.style.left = "50px";
        image.style.top = "50px";
        pos = 0; vpos = 50;
    }
    setTimeout("shiftImage()", 50);
}

var vpos = 0;
function shiftImage() {
    var image = document.getElementById("bucky");
    if (vpos < 250) {
        vpos += 2;
        image.style.top = vpos + "px";
        setTimeout("shiftImage()", 50);
    }
}
```

Changing the structure

```javascript
function addBold() {
    var b = document.createElement("b");
    b.appendChild(document.createTextNode("bold"));
    addParagraph(b);
}

function addParagraph(node) {
    var p = document.createElement("p");
    p.appendChild(document.createTextNode("Some "));
    p.appendChild(node);
    p.appendChild(document.createTextNode(" text.")));
    document.getElementById("playground").appendChild(p);
}

function clearAll() {
    var d = document.getElementById("playground");
    while (d.childNodes.length > 0) {
        d.removeChild(d.childNodes[0]);
    }
}
```