Walking the DOM Tree

CSE 190 M (Web Programming), Spring 2008
University of Washington

References: Forbes/Steele, Chipman (much of this content was stolen from them)

Lecture Outline

- DOM nodes
  - element nodes
  - text nodes
- DOM tree traversals
- Prototype additions to the DOM
- getting and setting text inside an element
- adding and removing content from a page

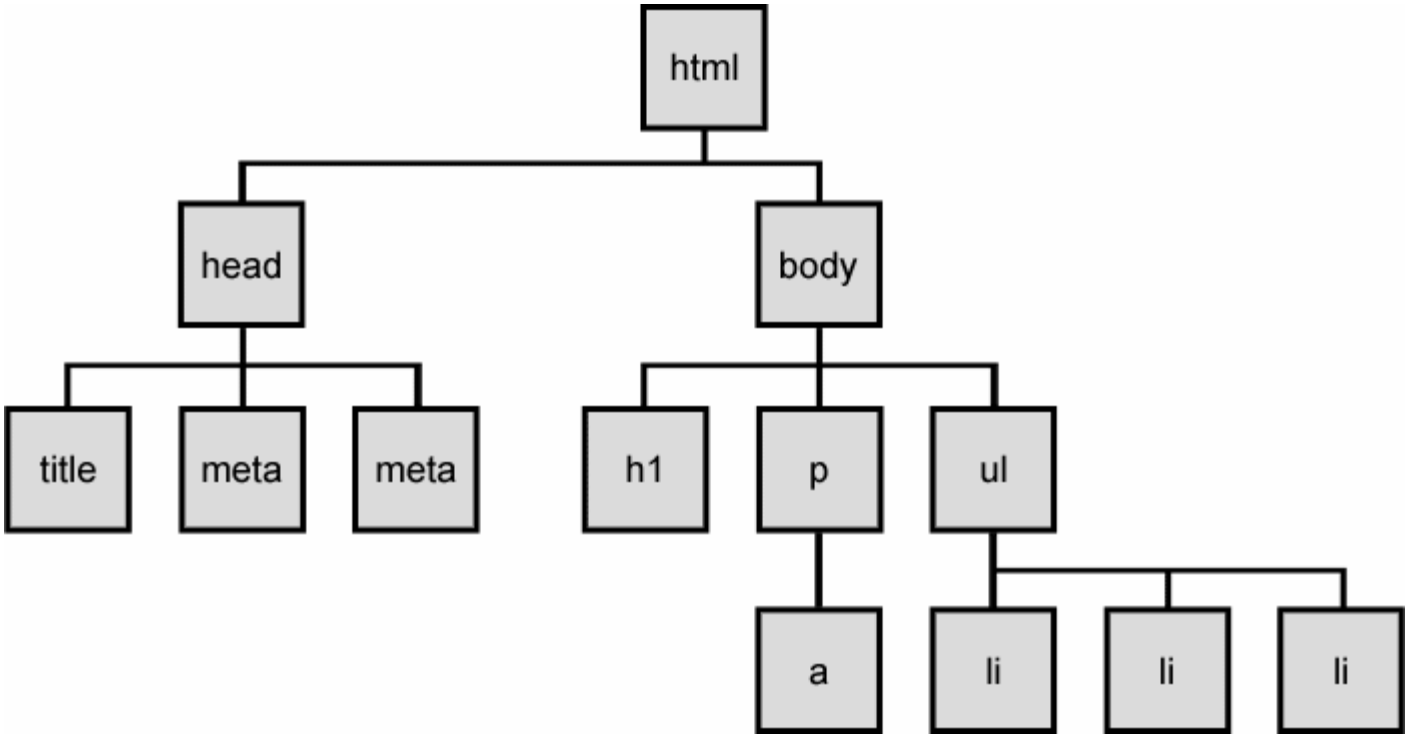
An example XHTML page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/tr/xhtml11/dtd/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/XHTML">
  <head>
    <title>Page Title</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  </head>

  <body>
    <h1>This is a heading</h1>
    <p>A paragraph with a <a href="http://www.google.com/">link</a>.</p>
    <ul>
      <li>a list item</li>
      <li>another list item</li>
      <li>a third list item</li>
    </ul>
  </body>
</html>
```

HTML

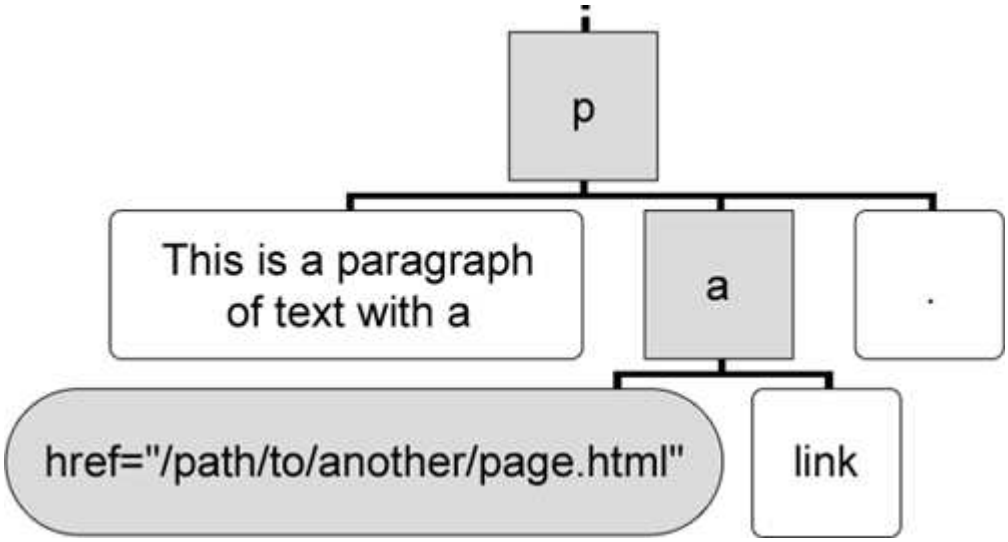
The resulting DOM tree



Types of nodes

```
<p>This is a paragraph of text with a
  <a href="/path/to/another/page.html">link</a> inside.</p>
```
HTML

- **element nodes** (HTML tag)
  - can have children and/or attributes
- **text nodes** (text in a block element)
  - a child within an element node
  - cannot have children or attributes
- **attribute nodes** (attribute/value pair inside the start of a tag)
  - a child within an element node
  - cannot have children or attributes



Traversing the DOM tree

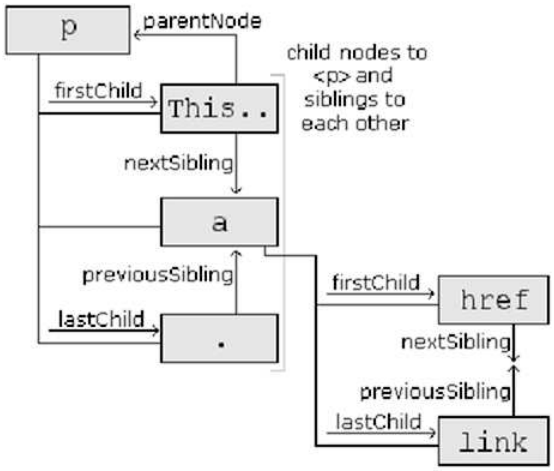
every node's DOM object has the following properties:

- `firstChild`, `lastChild` : start/end of this node's list of children
- `childNodes` : array of all this node's children
- `nextSibling`, `previousSibling` : neighboring nodes that have the same parent
- `parentNode` : the element that contains this node

- complete list of DOM node properties
- browser incompatiblity information (IE6 sucks)

DOM tree traversal example

```
<p id="foo">This is a paragraph of text with a
  <a href="/path/to/another/page.html">link</a> inside.</p>
```
HTML



- How would we change the word "link" in the above HTML to be "bunny"?

Annoying text nodes

```HTML
<div id="foo">
  <p>
    This is a paragraph of text with a
    <a href="page.html">link</a>
  </p>
</div>
```

- Q: How many children does the `div` above have?
- 3!
  - a *text node* representing `"\n\t"` (before the paragraph)
  - an element node representing the <p>
  - a *text node* representing `"\n\t"` (after the paragraph)
- these text nodes are annoying when traversing siblings, children, etc.

Prototype DOM tree traversal methods

All of the following methods return only element nodes (not text nodes):

- `ancestors` : array of elements above this one
- `childElements` : array of children (*elements only, not text nodes*)
- `descendants`, `firstDescendant`, `descendantOf` : array of all this element's children, grandchildren, etc.
- `next`, `previous`, `siblings`, `previousSiblings`, `nextSiblings`, `adjacent` : methods to access node's siblings

Example

set all siblings of the element with `id` of `main` to be bold:

```JS
var siblings = $("main").siblings();
for (var i = 0; i < siblings.length; i++) {
  siblings[i].style.fontWeight = "bold";
}
```

- note that these are *methods*, not properties (must use `()`)

Modifying the text inside a node

```
$("foo").textContent = "New paragraph text";  // change text on the page
```
JS

- `textContent` : text (no HTML tags) inside a node
  - for Firefox/Opera/Safari; **a web standard** (preferred for this course)
- `innerText` : text (no HTML tags) inside a node
  - for IE, which does not support the standard `textContent` property
  - cross-browser version:

```
$("foo").innerText = $("foo").textContent = "Some text";
```

- `innerHTML` : text and/or HTML tags inside a node
  - powerful, but non-standard; discouraged

Abuse of `innerHTML`

```
// bad style!
$("foo").innerHTML = "<p>text and <a href="page.html">link</a>";
```
JS

- `innerHTML` can inject arbitrary XHTML content into the page
- however, this is prone to bugs and errors and is considered poor style
- in this course, we forbid using `innerHTML` and instead encourage using `textContent` only to set plain text contents inside an element
  - so then, how do we add content with XHTML tags in it to the page?

Creating new nodes

```
// create a new <h2> node
var newHeading = document.createElement("h2");
newHeading.style.color = "green";
newHeading.textContent = "This is a heading";
```
JS

- `document.createElement("`*tag*`")` : creates and returns a new empty DOM node representing an element of that type
  - this node's properties can be set just like any other DOM node's
  - IE note: If you want to use Prototype methods on a newly created element, you must call `$` or `Element.extend` on it
- `document.createTextNode("`*text*`")` : creates and returns a new text node containing the given text

Adding a node to the page

```JS
window.onload = function() {
  $("thisslide").onclick = slideClick;
}

function slideClick() {
  var p = document.createElement("p");
  p.textContent = "A paragraph!";
  $("thisslide").appendChild(p);
}
```

- merely creating a node does not add it to the page
- you must add the new node to the children list of an existing node on the page

Modifying the DOM tree

Every DOM node object has these methods:

- `appendChild(`*node*`)` : places the given node at the end of this node's child list
- `insertBefore(`*newChild*`, `*oldChild*`)` : places the given new node in this node's child list just before `oldChild`
- `removeChild(`*node*`)` : removes given node from this node's child list
- `replaceChild(`*newChild*`, `*oldChild*`)` : replaces given child with new node
- `up` *, `down` *, `remove` * : moves this node up/down a level in the tree, or deletes it (Prototype)

DOM versus `innerHTML` revisited

Why not just code the previous example this way?

```JS
function slideClick() {
  $("thisslide").innerHTML += "<p>A paragraph!</p>";
}
```

- it's fewer lines of code, and debatably easier to read... what's wrong with it?

Ugly `innerHTML` code

Imagine that the new node is more complex:

```JS
function slideClick() {
  this.innerHTML += "<p style='color: red; " +
      "margin-left: 50px;' " +
      "onclick='myOnClick();'>" +
      "A paragraph!</p>";
}
```

- ugly
- must carefully distinguish " and '
- bad style on many levels (e.g. JS code embedded within HTML)
- can only add at beginning or end, not in middle of child list

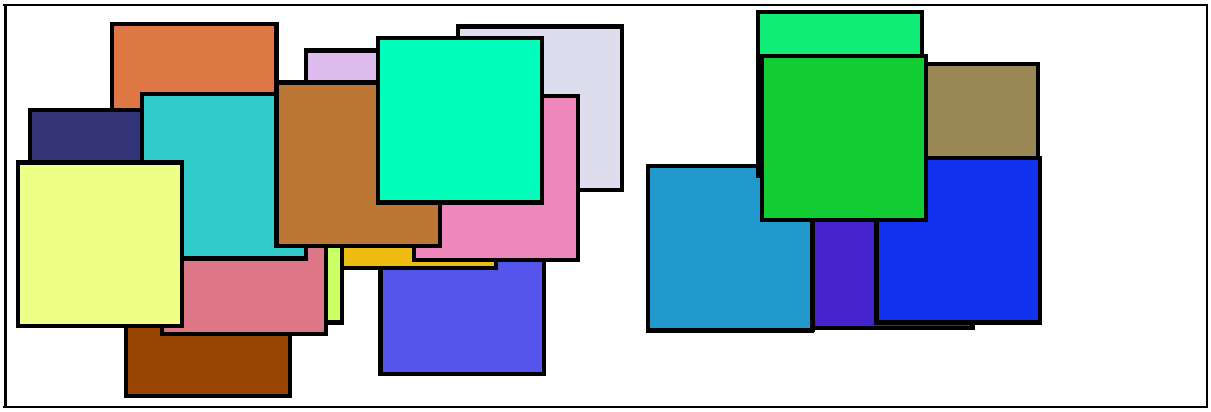
Benefits of DOM over `innerHTML`

```JS
function slideClick() {
  var p = document.createElement("p");
  p.style.color = "red";
  p.style.marginLeft = "50px";
  p.onclick = myOnClick;
  p.textContent = "A paragraph!";
  $("thisslide").appendChild(p);
}
```

- cleaner to attach event handlers to DOM object
- cleaner to set styles on DOM object

Practice problem: Rectangles

Click a rectangle to move it to the front. Shift-click a rectangle to delete it.



- write JavaScript code to create and manipulate random rectangles
- Hint: see absolute and relative positioning from the layout slides.
- Hint: use `z-index` property to adjust which rectangles are on top