Complex DOM manipulation problems

How would we do each of the following in JavaScript code? Each involves modifying each one of a group of elements ...

- When the Go button is clicked, reposition all the divs of class puzzle to random x/y locations.
- When the user hovers over the maze boundary, turn all maze walls red.
- Change every other item in the ul list with id of TAs to have a gray background.
The DOM tree (8.3)

- The elements of a page are nested into a tree-like structure of objects
  - the DOM has properties and methods for traversing this tree

Types of DOM nodes (8.3.1)

- **element nodes** (HTML tag)
  - can have children and/or attributes

- **text nodes** (text in a block element)

- **attribute nodes** (attribute/value pair)
  - text/attributes are children in an element node
  - cannot have children or attributes
  - not usually shown when drawing the DOM tree
Traversing the DOM tree (8.3.2 - 8.3.3)

every node's DOM object has the following properties:

<table>
<thead>
<tr>
<th>name(s)</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstChild, lastChild</td>
<td>start/end of this node's list of children</td>
</tr>
<tr>
<td>childNodes</td>
<td>array of all this node's children</td>
</tr>
<tr>
<td>nextSibling, previousSibling</td>
<td>neighboring nodes with the same parent</td>
</tr>
<tr>
<td>parentNode</td>
<td>the element that contains this node</td>
</tr>
</tbody>
</table>

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

DOM tree traversal example

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

![DOM tree diagram]
Element vs. text nodes

```html
<div>
  <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?
  - A: 3
    - an element node representing the `<p>`
    - two text nodes representing "
      \n      t"
      (before/after the paragraph)
- Q: How many children does the paragraph have? The `a` tag?

Prototype's **DOM element** methods (9.1.3)

<table>
<thead>
<tr>
<th>absolutize</th>
<th>addClassName</th>
<th>classNames</th>
<th>cleanWhitespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>cumulativeOffset</td>
<td>cumulativeScrollOffset</td>
<td>empty</td>
<td>extend</td>
</tr>
<tr>
<td>getDimensions</td>
<td>getHeight</td>
<td>getOffsetParent</td>
<td>getStyle</td>
</tr>
<tr>
<td>hasClassName</td>
<td>hide</td>
<td>identify</td>
<td>insert</td>
</tr>
<tr>
<td>makeClipping</td>
<td>makePositioned</td>
<td>match</td>
<td>positionedOffset</td>
</tr>
<tr>
<td>recursivelyCollect</td>
<td>relativize</td>
<td>remove</td>
<td>removeClassName</td>
</tr>
<tr>
<td>scrollTo</td>
<td>select</td>
<td>setOpacity</td>
<td>setStyle</td>
</tr>
<tr>
<td>toggle</td>
<td>toggleClassName</td>
<td>undoClipping</td>
<td>undoPositioned</td>
</tr>
<tr>
<td>viewportOffset</td>
<td>visible</td>
<td>wrap</td>
<td>writeAttribute</td>
</tr>
</tbody>
</table>

- categories: CSS classes, DOM tree traversal/manipulation, events, styles
## Prototype's DOM tree traversal methods (9.1.5)

<table>
<thead>
<tr>
<th>method(s)</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ancestors, up</td>
<td>elements above this one</td>
</tr>
<tr>
<td>childElements, descendants, down</td>
<td>elements below this one (not text nodes)</td>
</tr>
<tr>
<td>siblings, next, nextSiblings,</td>
<td>elements with same parent as this one (not text nodes)</td>
</tr>
<tr>
<td>previous, previousSiblings,</td>
<td></td>
</tr>
<tr>
<td>adjacent</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// alter siblings of "main" that do not contain "Sun"
var sibs = $('main').siblings();
for (var i = 0; i < sibs.length; i++) {
    if (sibs[i].innerHTML.indexOf("Sun") < 0) {
        sibs[i].innerHTML += " Sunshine";
    }
}
```

- Prototype strips out the unwanted text nodes
- notice that these are methods, so you need ()

## Selecting groups of DOM objects (8.3.5)

- methods in `document` and other DOM objects for accessing descendents:

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getElementsByTagName</td>
<td>returns array of descendents with the given tag, such as &quot;div&quot;</td>
</tr>
<tr>
<td>getElementsByName</td>
<td>returns array of descendents with the given <code>name</code> attribute (mostly useful for accessing form controls)</td>
</tr>
</tbody>
</table>
Getting all elements of a certain type

highlight all paragraphs in the document:

```javascript
var allParas = document.getElementsByTagName("p");
for (var i = 0; i < allParas.length; i++) {
  allParas[i].style.backgroundColor = "yellow";
}
```

```html
<body>
  <p>This is the first paragraph</p>
  <p>This is the second paragraph</p>
  <p>You get the idea...</p>
</body>
```

Combining with `getElementById`

highlight all paragraphs inside of the section with ID "address":

```javascript
var addrParas = $($("address")).getElementsByTagName("p");
for (var i = 0; i < addrParas.length; i++) {
  addrParas[i].style.backgroundColor = "yellow";
}
```

```html
<p>This won't be returned!</p>
<div id="address">
  <p>1234 Street</p>
  <p>Atlanta, GA</p>
</div>
```
Prototype's methods for selecting elements

Prototype adds methods to the `document` object (and all DOM element objects) for selecting groups of elements:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getElementsByClassName</code></td>
<td>array of elements that use given <code>class</code> attribute</td>
</tr>
<tr>
<td><code>select</code></td>
<td>array of descendants that match given CSS selector, such as &quot;div#sidebar ul.news &gt; li&quot;</td>
</tr>
</tbody>
</table>

```
var gameButtons = $("game").select("button.control");
for (var i = 0; i < gameButtons.length; i++) {
    gameButtons[i].style.color = "yellow";
}
```

The `$$` function (9.1.5)

```
var arrayName = $("CSS selector");

// hide all "announcement" paragraphs in the "news" section
var paragraphs = $$("div#news p.announcement");
for (var i = 0; i < paragraphs.length; i++) {
    paragraphs[i].hide();
}
```

- `$$` returns an array of DOM elements that match the given CSS selector
  - like `$` but returns an array instead of a single DOM object
  - a shorthand for `document.select`
- useful for applying an operation each one of a set of elements
Common $$ issues

- many students forget to write . or # in front of a class or id

```javascript
// get all buttons with a class of "control"
var gameButtons = $$("control");
var gameButtons = $$(".control");
```

- $$ returns an array, not a single element; must loop over the results

```javascript
// set all buttons with a class of "control" to have red text
$$(".control").style.color = "red";
var gameButtons = $$(".control");
for (var i = 0; i < gameButtons.length; i++) {
    gameButtons[i].style.color = "red";
}
```

- Q: Can I still select a group of elements using $$ even if my CSS file doesn't have any style rule for that same group? (A: Yes!)

Creating new nodes (8.3.5)

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>document.createElement(&quot;tag&quot;)</code></td>
<td>creates and returns a new empty DOM node representing an element of that type</td>
</tr>
<tr>
<td><code>document.createTextNode(&quot;text&quot;)</code></td>
<td>creates and returns a text node containing given text</td>
</tr>
</tbody>
</table>

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

- merely creating a node does not add it to the page
- you must add the new node as a child of an existing element on the page...
Modifying the DOM tree

Every DOM element object has these methods:

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appendChild(node)</td>
<td>places given node at end of this node's child list</td>
</tr>
<tr>
<td>insertBefore(new, old)</td>
<td>places the given new node in this node's child list just before old child</td>
</tr>
<tr>
<td>removeChild(node)</td>
<td>removes given node from this node's child list</td>
</tr>
<tr>
<td>replaceChild(new, old)</td>
<td>replaces given child with new node</td>
</tr>
</tbody>
</table>

var p = document.createElement("p");
p.innerHTML = "A paragraph!";
$("main").appendChild(p);

Removing a node from the page

function slideClick() {
    var bullets = document.getElementsByTagName("li");
    for (var i = 0; i < bullets.length; i++) {
        if (bullets[i].innerHTML.indexOf("children") >= 0) {
            bullets[i].remove();
        }
    }
}

- each DOM object has a removeChild method to remove its children from the page
- Prototype adds a remove method for a node to remove itself
DOM versus innerHTML hacking

Why not just code the previous example this way?

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

- Imagine that the new node is more complex:
  - ugly: bad style on many levels (e.g. JS code embedded within HTML)
  - error-prone: must carefully distinguish " and '
  - can only add at beginning or end, not in middle of child list

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

Problems with reading/changing styles

```html
<button id="clickme">Click Me</button>
```

```javascript
window.onload = function() {
  $('#clickme').onclick = biggerFont;
};

function biggerFont() {
  var size = parseInt($('#clickme').style.fontSize);
  size += 4;
  $('#clickMe').style.fontSize = size + "pt";
}
```

- `style` property lets you set any CSS style for an element
- problem: you cannot (usually) read existing styles with it
Accessing styles in Prototype (9.1.4)

```javascript
function biggerFont() {
    // turn text yellow and make it bigger
    var size = parseInt($("clickme").getStyle("font-size"));
    $"clickme".style.fontSize = (size + 4) + "pt";
}
```

- `getStyle` function added to DOM object allows accessing existing styles
- `addClassName`, `removeClassName`, `hasClassName` manipulate CSS classes

**Common bug: incorrect usage of existing styles**

```javascript
this.style.top = this.getStyle("top") + 100 + "px"; // bad!
```

- the above example computes e.g. "200px" + 100 + "px",
  which would evaluate to "200px100px"
- a corrected version:

```javascript
this.style.top = parseInt(this.getStyle("top")) + 100 + "px"; // correct
```
Setting CSS classes in Prototype (9.1.4)

```javascript
function highlightField() {
    // turn text yellow and make it bigger
    if (!$("text").hasClass("invalid")) {
        $\"text\".addClassName\"highlight\";
    }
}
```

- `addClassName`, `removeClassName`, `hasClassName` manipulate CSS classes
- similar to existing `className` DOM property, but don't have to manually split by spaces