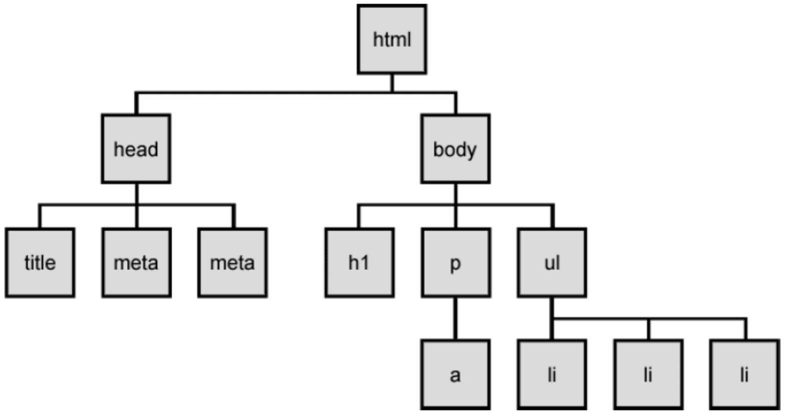
The DOM tree

The DOM tree



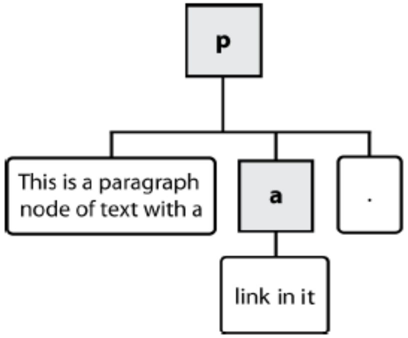
Types of DOM nodes

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

- 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

Types of DOM nodes

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



Traversing the DOM tree

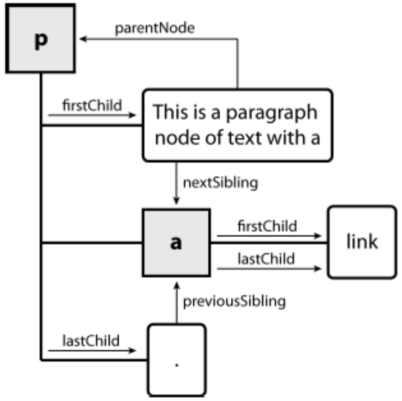
name(s)	description
firstChild, lastChild	start/end of this node's list of children
childNodes	array of all this node's children
nextSibling, previousSibling	neighboring nodes with the same parent
parentNode	the element that contains this node

- complete list of DOM node properties
- browser incompatiblity information

DOM tree traversal example

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

HTML



Elements vs text nodes

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

HTML

- 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

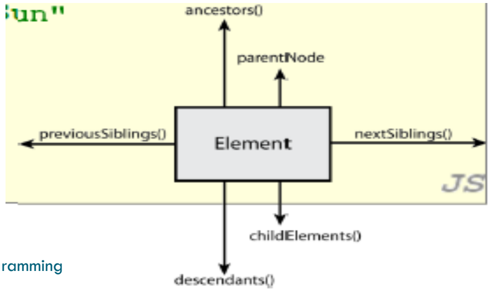
absolutize	addClassName	classNames	cleanWhitespace	clonePosition
cumulativeOffset	cumulativeScrollOffset	empty	extend	firstDescendant
getDimensions	getHeight	getOffsetParent	getStyle	getWidth
hasClassName	hide	identify	insert	inspect
makeClipping	makePositioned	match	positionedOffset	readAttribute
recursivelyCollect	relativize	remove	removeClassName	replace
scrollTo	select	setOpacity	setStyle	show
toggle	toggleClassName	undoClipping	undoPositioned	update
viewportOffset	visible	wrap	writeAttribute	

Prototype's DOM tree traversal methods

method(s)	description
ancestors, up	elements above this one
childElements, descendants, down	elements below this one (not text nodes)
siblings, next, nextSiblings, previous, previousSiblings, adjacent	elements with same parent as this one (not text nodes)

Prototype's DOM tree traversal methods

```
// 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";
      }
}                                                         JS
```



Selecting groups of DOM objects

- methods in document and other DOM objects for accessing descendants:

name	description
getElementsByTagName	returns array of descendents with the given tag, such as "div"
getElementsByName	returns array of descendants with the given name attribute (mostly useful for accessing form controls)

Getting all elements of a certain type

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

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

Combining with getElementById

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

```
<p>This won't be returned!</p>
<div id="address">
       <p>1234 Street</p>
       <p>Atlanta, GA</p>
</div>                                              HTML
```

Prototype's methods for selecting elements

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

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

getElementsByClassName	array of elements that use given class attribute
select	array of descendants that match given CSS selector, such as "div#sidebar ul.news > li"

```
<ul id="fruits">
  <li id="apples">apples
    <ul>
      <li id="golden-delicious">Golden Delicious</li>
      <li id="mutsu" class="yummy">Mutsu</li>
      <li id="mcintosh" class="yummy">McIntosh</li>
      <li id="ida-red">Ida Red</li>
    </ul>
  </li>
  <li id="exotic" class="yummy">exotic fruits
    <ul>
      <li id="kiwi">kiwi</li>
      <li id="granadilla">granadilla</li>
    </ul>
  </li>
</ul>HTML
```

```
$('fruits').getElementsByClassName('yummy');
// -> [li#mutsu, …]

$('exotic').getElementsByClassName('yummy');
// ->                                          JS
```

```
<ul id="fruits">
  <li id="apples">
    <h3 title="yummy!">Apples</h3>
    <ul id="list-of-apples">
      <li id="golden-delicious" title="yummy!" >Golden
Delicious</li>
      <li id="mutsu" title="yummy!">Mutsu</li>
      <li id="mcintosh">McIntosh</li>
      <li id="ida-red">Ida Red</li>
    </ul>
    <p id="saying">An apple a day keeps the doctor
away.</p>
  </li>
</ul>
```

```
$('apples').select('[title="yummy!"]');
// -> [h3, li#golden-delicious, li#mutsu]

$('apples').select( 'p#saying', 'li[title="yummy!"]');
//
$('apples').select('[title="disgusting!"]');
//                                                 JS
```

The $$ function


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

```
// 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();
}                                                            JS
```

- $$ 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 with $$


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

```
// 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";
}                                                            JS
```

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

name	description
document.createElement("tag")	creates and returns a new empty DOM node representing an element of that type
document.createTextNode("text")	creates and returns a text node containing given text

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

- 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

name	description
appendChild(node)	places given node at end of this node's child list
insertBefore(new, old)	places the given new node in this node's child list just before old child
removeChild(node)	removes given node from this node's child list
replaceChild(new, old)	replaces given child with new node

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

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();
                }
        }
}                                                        JS
```

- □ 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?

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

- □ 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

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

Problems with reading/changing styles

```
window.onload = function() {
        $("clickme").onclick = biggerFont;
};
function biggerFont() {
        var size = parseInt($("clickme").style.fontSize);
        size += 4;
        $("clickMe").style.fontSize = size + "pt";
}                                                        JS
```

- □ style property lets you set any CSS style for an element
- □ problem: you cannot (usually) read existing styles with it

Accessing styles in Prototype

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

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

Common bug: incorrect usage of existing styles

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

JS

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

```
this.style.top = parseInt(this.getStyle("top")) + 100 +
"px"; // correct
```

JS

Setting CSS classes in Prototype

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

JS

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

Example: createElements

```
<html>
<head>
<script src="
https://ajax.googleapis.com/ajax/libs/prototype/1.7.0.0/pr
ototype.js " type="text/javascript"></script>
<script src="paragraph.js "
type="text/javascript"></script>
</head>

<body>
<div id="paragrapharea">
       <button id="add">Add a paragraph</button>
</div>
</body>
</html>
```

Example: createElements

```
window.onload = function(){
       var button = $("add");
       button.onclick = addParagraphClick;
}

function addParagraphClick(){
       var paragraph = document.createElement("p");
       paragraph.innerHTML = "All work and no play makes
Jack a dull boy";
       var area = $("paragrapharea");
       area.appendChild(paragraph);
}

function addListClick(){
       }
```

JS