JS: DOM

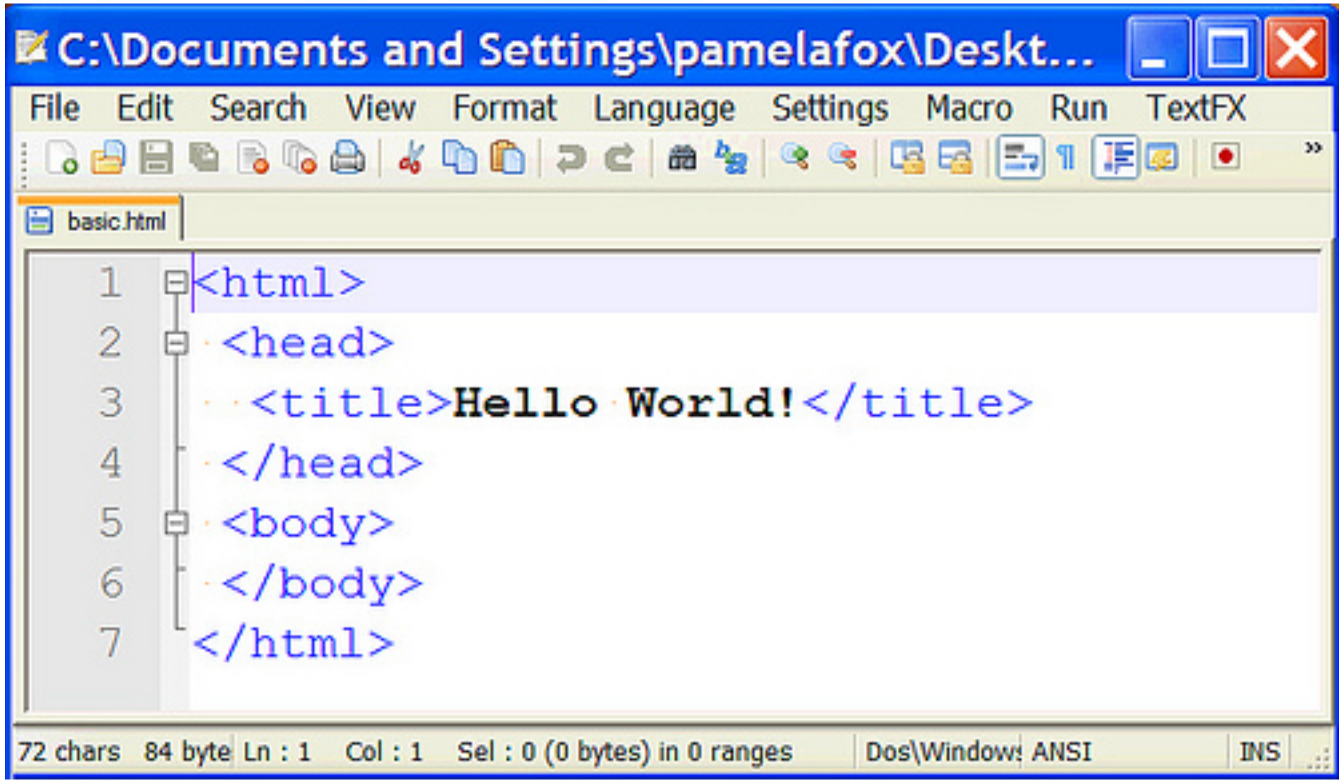
HTML & CSS: Review

```html
<!doctype html>
<html>
 <head>
  <meta charset="utf-8">
  <title>All About Cats</title>
  <style type="text/css">
   h1 {
     color: red;
   }
  #mainpicture {
    border: 1px solid black;
  }
  .catname {
    font-weight: bold;
  }
  </style>
 </head>
 <body>
  <h1>CATS!</h1>
  <img id="mainpicture" src="http://placekitten.com/200/300">
  <p>So cute!</p>
  <ul>
   <li class="catname">Lizzie</li>
   <li class="catname">Daemon</li>
  </ul>
 </body>
</html>
```

HTML Editors

Since HTML files are just text files, many programs can be used to create them. Some programs provide special assistance for handling HTML, like syntax-highlighting or autocompletion.

	Windows	**Mac**	**Online**
Free	Notepad++	Brackets, Atom,TextWrangler, Smultron	Cloud9 IDE,JSBin
$$		SublimeText, TextMate,Coda, Espresso	


C:\Documents and Settings\pamelafox\Deskt...
File Edit Search View Format Language Settings Macro Run TextFX
basic.html

```html
<html>
 <head>
  <title>Hello World!</title>
 </head>
 <body>
 </body>
</html>
```

72 chars 84 byte Ln : 1 Col : 1 Sel : 0 (0 bytes) in 0 ranges Dos\Windows ANSI INS

JS in HTML

You can put JS inside a `script` tag (commonly at bottom of the page):

```
...
<script>
```

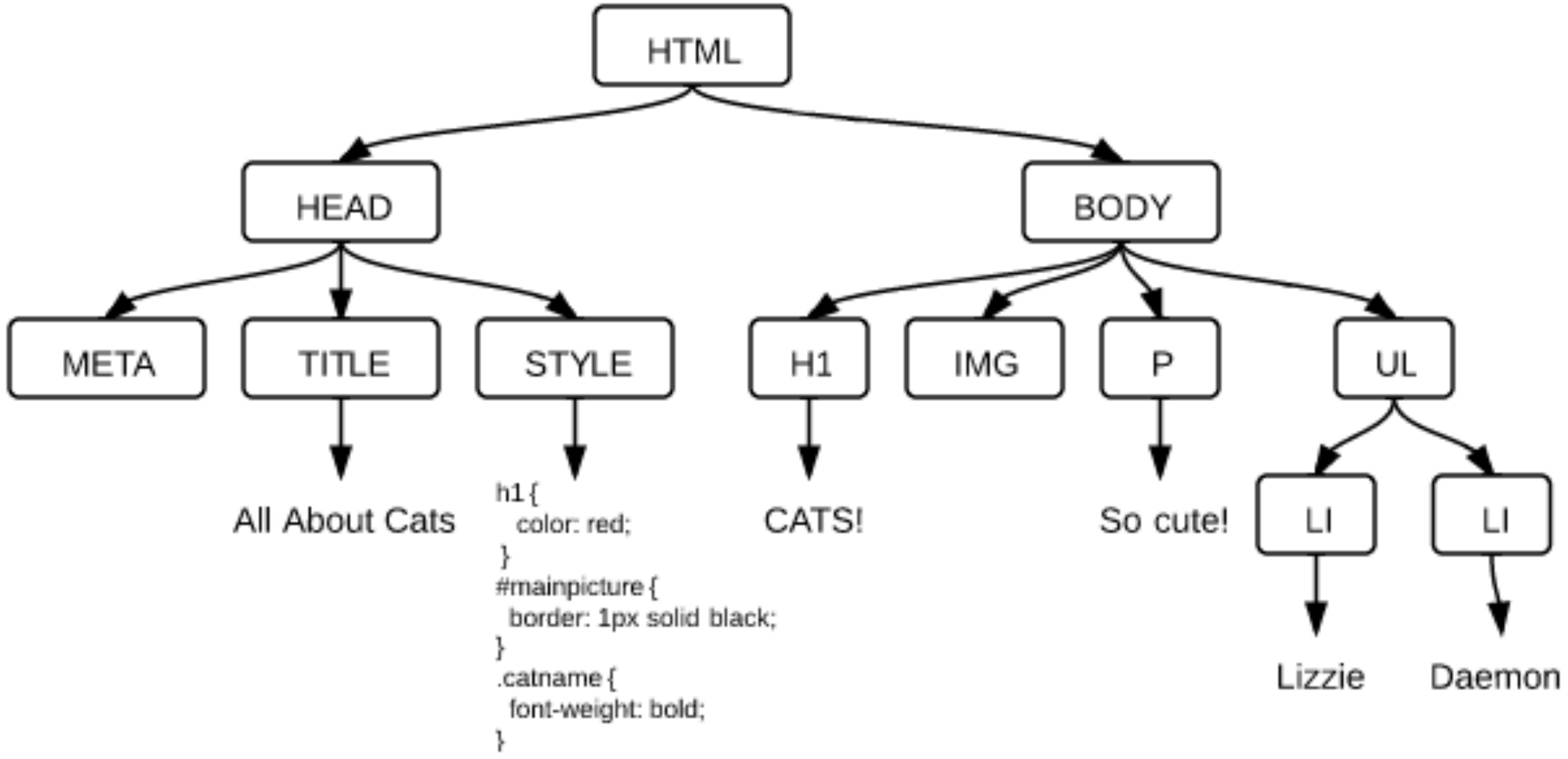
```
  console.log('IM ON A WEBPAGE!');
  </script>
 </body>
</html>
```

You can also put JS in an external file and reference it:

```
  ...
  <script src="app.js"></script>
 </body>
</html>
```

The DOM Tree

For this page:



DOM Inspecting

- Chrome: Right-click -> "Inspect Element"

- Firefox: Right-click -> "Inspect Element" -> "HTML"
- IE: Open Tools -> Developer Tools

Chrome:

DOM Access

The `document` object gives us ways of accessing and changing the DOM of the current webpage.

General strategy for DOM manipulation:

1. Find the DOM node using an access method and store it into a variable.
2. Manipulate the DOM node by changing its attributes, styles, inner HTML, or appending new nodes to it.

DOM Access: By Id

The method signature:

```
document.getElementById(id);
```

If the HTML had:

```
<img id="mainpicture" src="http://placekitten.com/200/300">
```

We'd access it this way:

```
var img = document.getElementById('mainpicture');
```

DOM Access: By Tag Name

The method signature:

```
document.getElementsByTagName(tagName);
```

If the HTML had:

```
<li class="catname">Lizzie</li>

<li class="catname">Daemon</li>
```

We'd access it this way:

```
var listItems = document.getElementsByTagName('li');

for (var i =0; i < listItems.length; i++) {

  var listItem = listItems[i];

}
```

DOM Access: HTML5

The HTML5 spec includes a few even more convenient methods.

Available in IE9+, FF3.6+, Chrome 17+, Safari 5+:

```
document.getElementsByClassName(className);
```

```
var catNames = document.getElementsByClassName('catname');

for (var i =0; i < catNames.length; i++) {

  var catName = catNames[i];

}
```

Available in IE8+, FF3.6+, Chrome 17+, Safari 5+:

```
document.querySelector(cssQuery);

document.querySelectorAll(cssQuery);
```

```
var catNames = document.querySelectorAll('ul li.catname');
```

DOM Access: Single Element vs. Array

Some access methods return a single element:

- `getElementById()`
- `querySelector()` *returns only the first of the matching elements

```
  var firstCatName = document.querySelector('ul li.catname');
```

Others return a collection of elements in an array:

- `getElementByClassName()`
- `getElementByTagName()`
- `querySelectorAll()`

```
  var catNames = document.querySelectorAll('ul li.catname');

  var firstCatName = catNames[0];
```

Exercise Time!

DOM Nodes: Attributes

You can access and change attributes of DOM nodes using dot notation.

If we had this HTML:

```
<img id="mainpicture" src="http://placekitten.com/200/300">
```

We can change the src attribute this way:

```


var oldSrc = img.src;

img.src = 'http://placekitten.com/100/500';
```

To set class, use the property `className`:

```
img.className = "picture";
```

DOM Nodes: Styles

You can change styles on DOM nodes via the `style` property.

If we had this CSS:

```
body {
  color: red;
}
```

We'd run this JS:

```
var pageNode = document.getElementsByTagName('body')[0];
pageNode.style.color = 'red';
```

CSS property names with a "-" must be camelCased and number properties must have a unit:

```
body {
  background-color: pink;
  padding-top: 10px;
}
pageNode.style.backgroundColor = 'pink';
pageNode.style.paddingTop = '10px';
```

DOM innerHTML

Each DOM node has an `innerHTML` property with the HTML of all its children:

```
var pageNode = document.getElementsByTagName('body')[0];
```

You can read out the HTML like this:

```
console.log(pageNode.innerHTML);
```

You can set `innerHTML` yourself to change the contents of the node:

```
pageNode.innerHTML = "<h1>Oh Noes!</h1> <p>I just changed the whole page!</p>"
```

You can also just add to the `innerHTML` instead of replace:

```
pageNode.innerHTML += "...just adding this bit at the end of the page.";
```

Exercise Time!

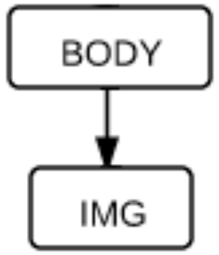
DOM Modifying

The `document` object also provides ways to create nodes from scratch:

```
document.createElement(tagName);

document.createTextNode(text);

document.appendChild();
```



```
var pageNode = document.getElementsByTagName('body')[0];


var newImg = document.createElement('img');

newImg.src = 'http://placekitten.com/400/300';

newImg.style.border = '1px solid black';

pageNode.appendChild(newImg);
```

```
var newParagraph = document.createElement('p');

var paragraphText = document.createTextNode('Squee!');

newParagraph.appendChild(paragraphText);

pageNode.appendChild(newParagraph);
```

Exercise Time!