# Chapter 6 – The DOM

- JavaScript is object based
- The browser is object based
  - We can access the browser's objects in the same way we did JavaScript's objects
- Two Object-Models
  - DOM (Document Object Model)
  - BOM (Browser Object Model)

    e.g. location.href = "http://cnn.com";

# Section 6-10

## The DOM
## (Document Object Model)

# How the DOM is Loaded

- When Web page is loaded into the browser
  - Every element on the page gets a "reference"
  - JavaScript code can use these references to change elements on a page

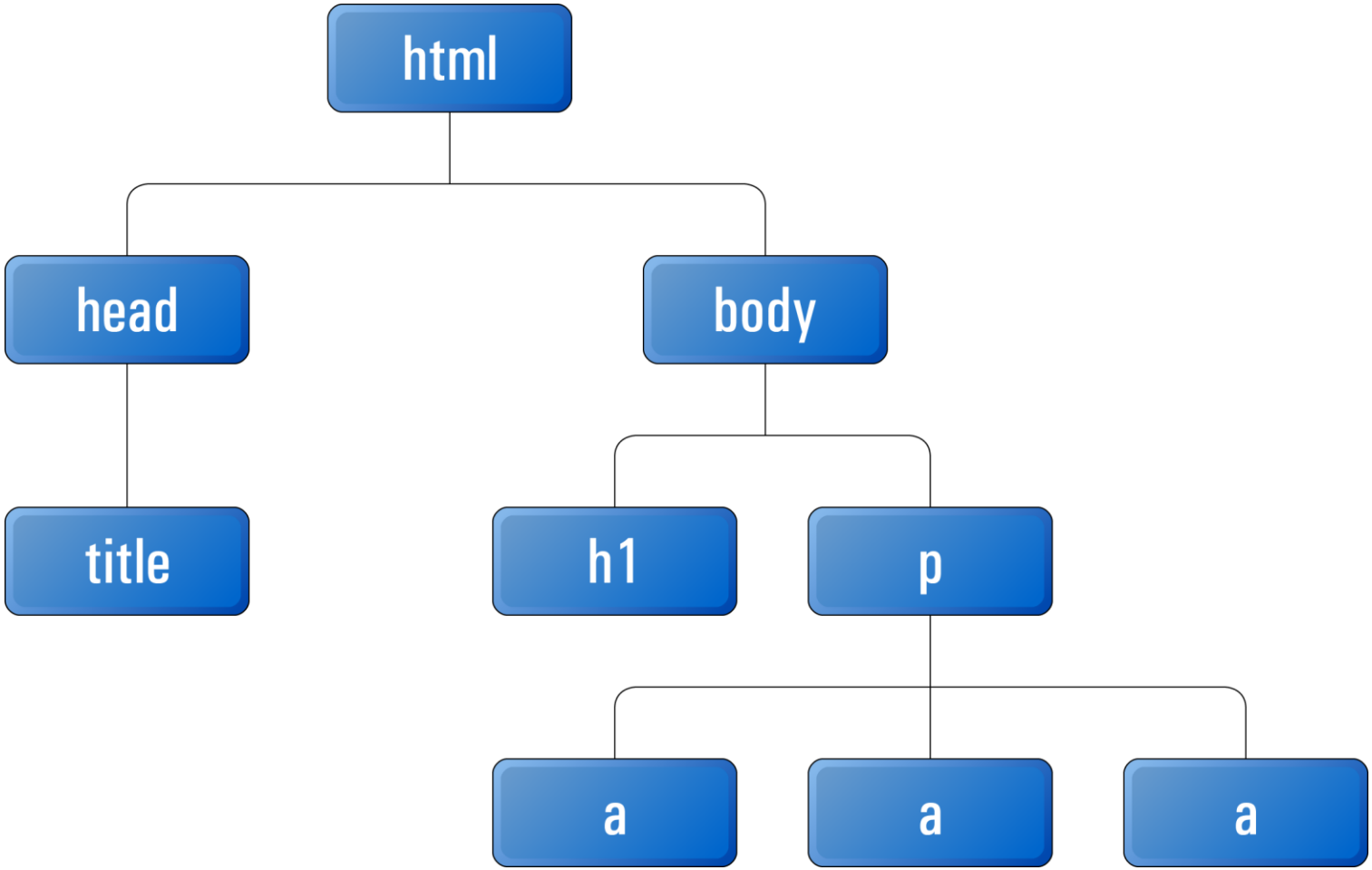
# Example HTML Code

```
<!DOCTYPE HTML>
<html>

<head>
    <title>DOMinating JavaScript</title>
</head>

<body>

    <h1> DOMinating JavaScript </h1>

    <p> If you need some help with your JavaScript read this
        <a href=http://www.danwebb.net/   rel="external">Dan Webb</a>,
        <a href="http://www.quirksmode.org/" rel="external">PPK</a> and
        <a href="http://adactio.com/" rel="external">Jeremy Keith</a>.
    </p>

</body>
</html>
```

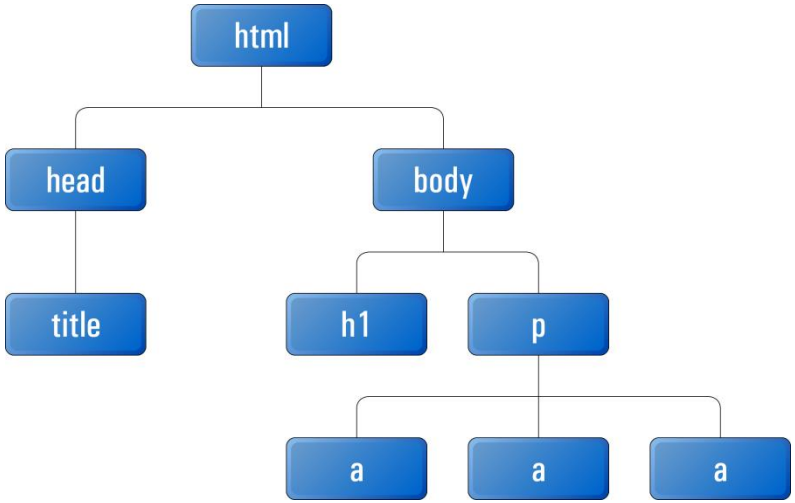
# Elements Become Objects

- Each Element on Web page becomes an objects
  - Has properties
  - Has methods
- Property values can often be changed
  - Causes Web page to change in-place

# Example Page Elements - Mapped



# Nodes



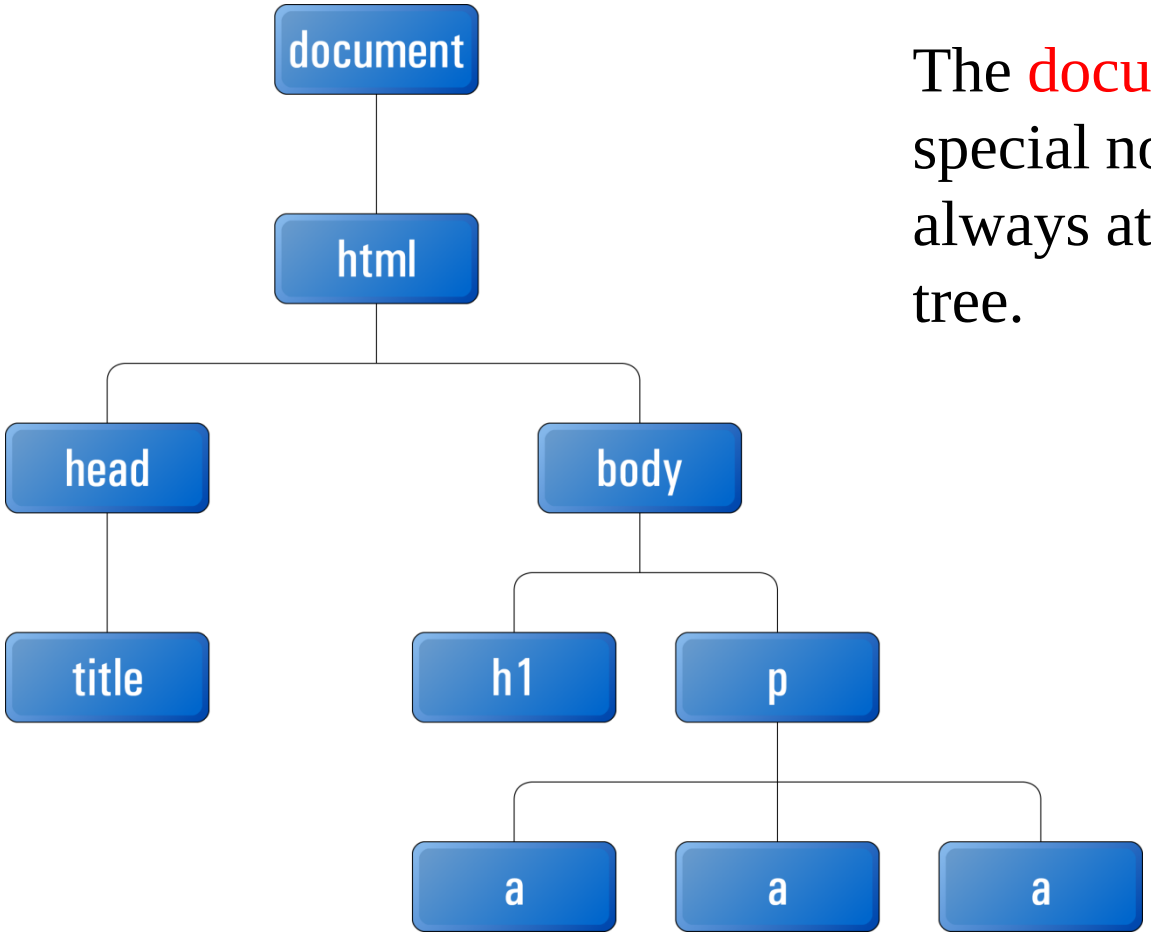

Each Object is represented by a Node

These are Element Objects

Every Element is identified by its tag name (e.g h1, p)

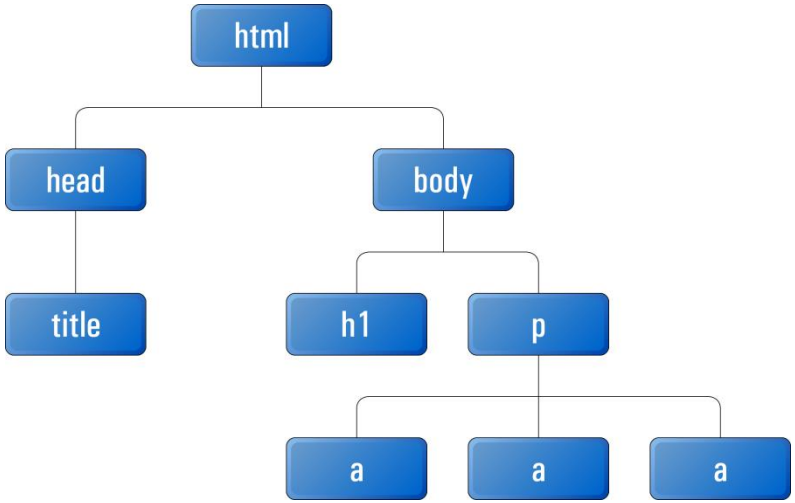
There is a parent-child relationship between the nodes

# The Document Node




The document node is a special node that is always at the top of the tree.

# Types of Nodes



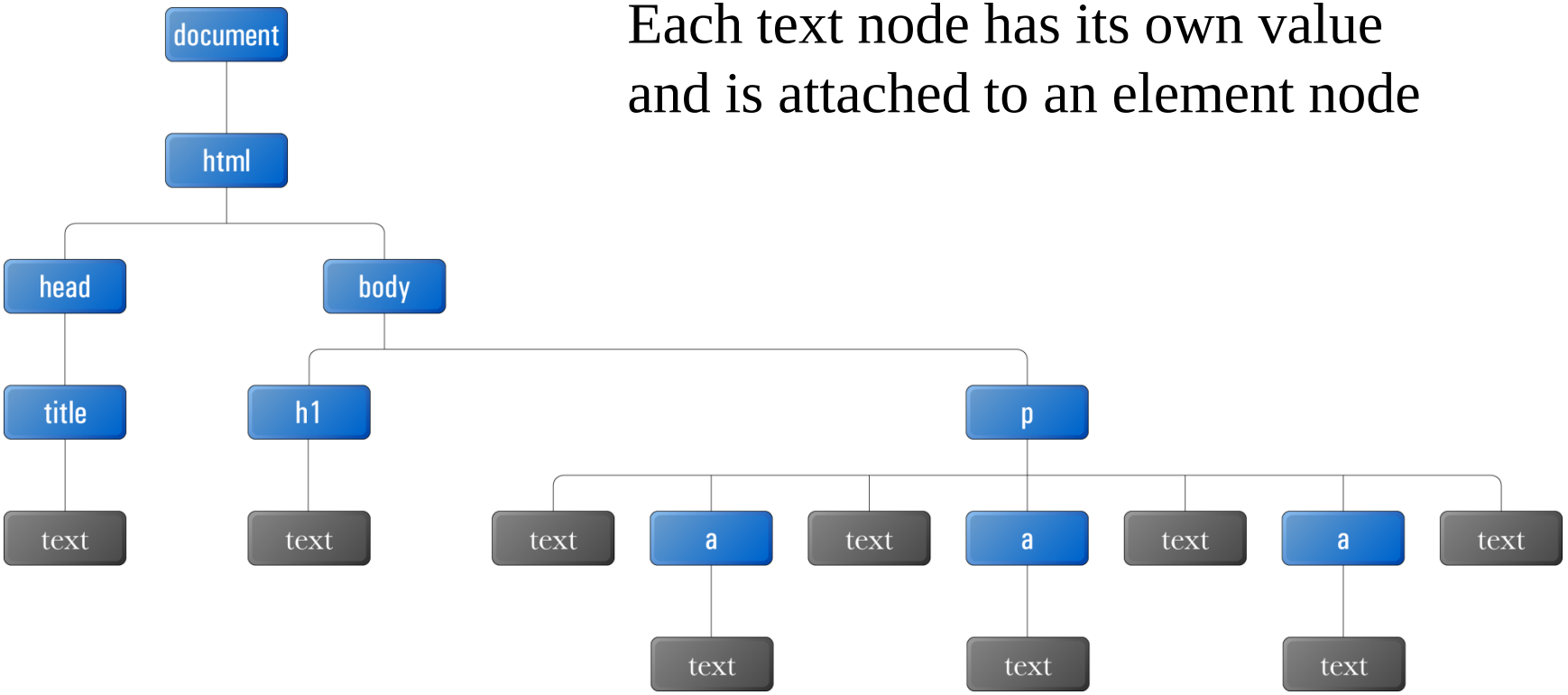

Element nodes point to the element itself, not its content!

Two other kinds of nodes for content

A text node is anything contained between tags or outside tags (e.g. text-element text-element

An attribute node is used to access attributes of a tag (e.g. 'href')

# Text Nodes

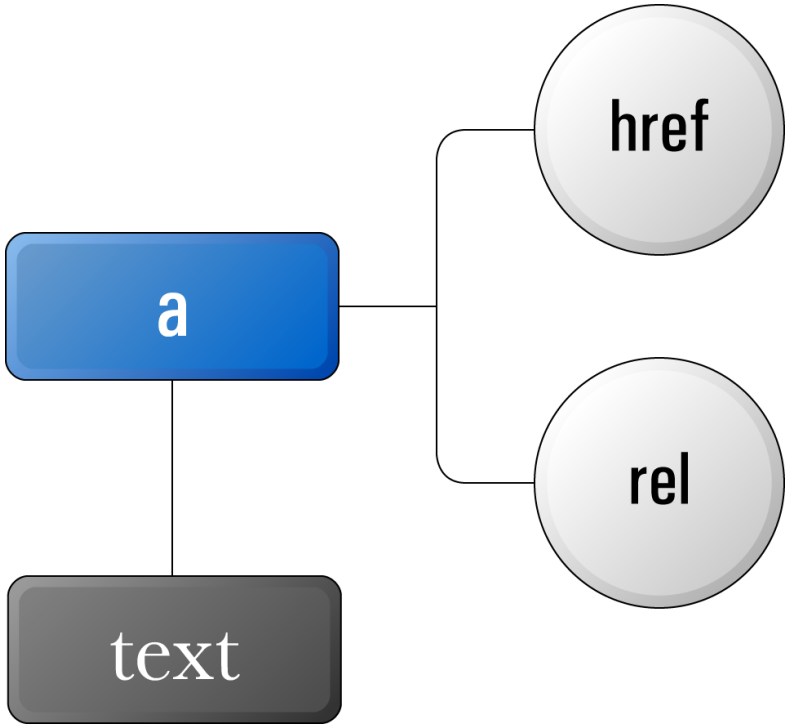



Each text node has its own value and is attached to an element node

# Whitespace and Text Nodes

- Whitespace may produce text nodes
- Different browsers handle whitespace differently… so be careful
  - Never rely upon the number or order of nodes when accessing the DOM

# Attribute Nodes




Attribute nodes point to the attributes of the element

Here we see an "Anchor" element node with a text node and two attribute nodes, "href" and "rel"

# Accessing Nodes

- Finding an element by a specific ID

```
<p id="uniqueElement">
    Hi There
</p>
```

- Use getElementById() method

```
var myPara1
    = document.getElementById("uniqueElement");
```

- We use the $ shortcut

```
var myPara1 = $("uniqueElement");
```

# Node Properties

- Once you have reference to an element, you have access to it's properties
- Examples: nodeName and nodeType

```
<p id="para1">Hi There</p>

var target = $("para1");

alert(target.nodeName);    //displays  "p"
alert(target.nodeType);     //displays  "1"  - Element Object
alert(target.firstChild.nodeType);     //displays  "3"  - Text Object
```

# Exercise 6-1

- Do the exercises for this section

  (shown to the right for the link to this presentation)

# Section 6-20

## Navigating the DOM

# Navigating the DOM Tree

- Finding a Parent
- Finding Children
- Finding Siblings

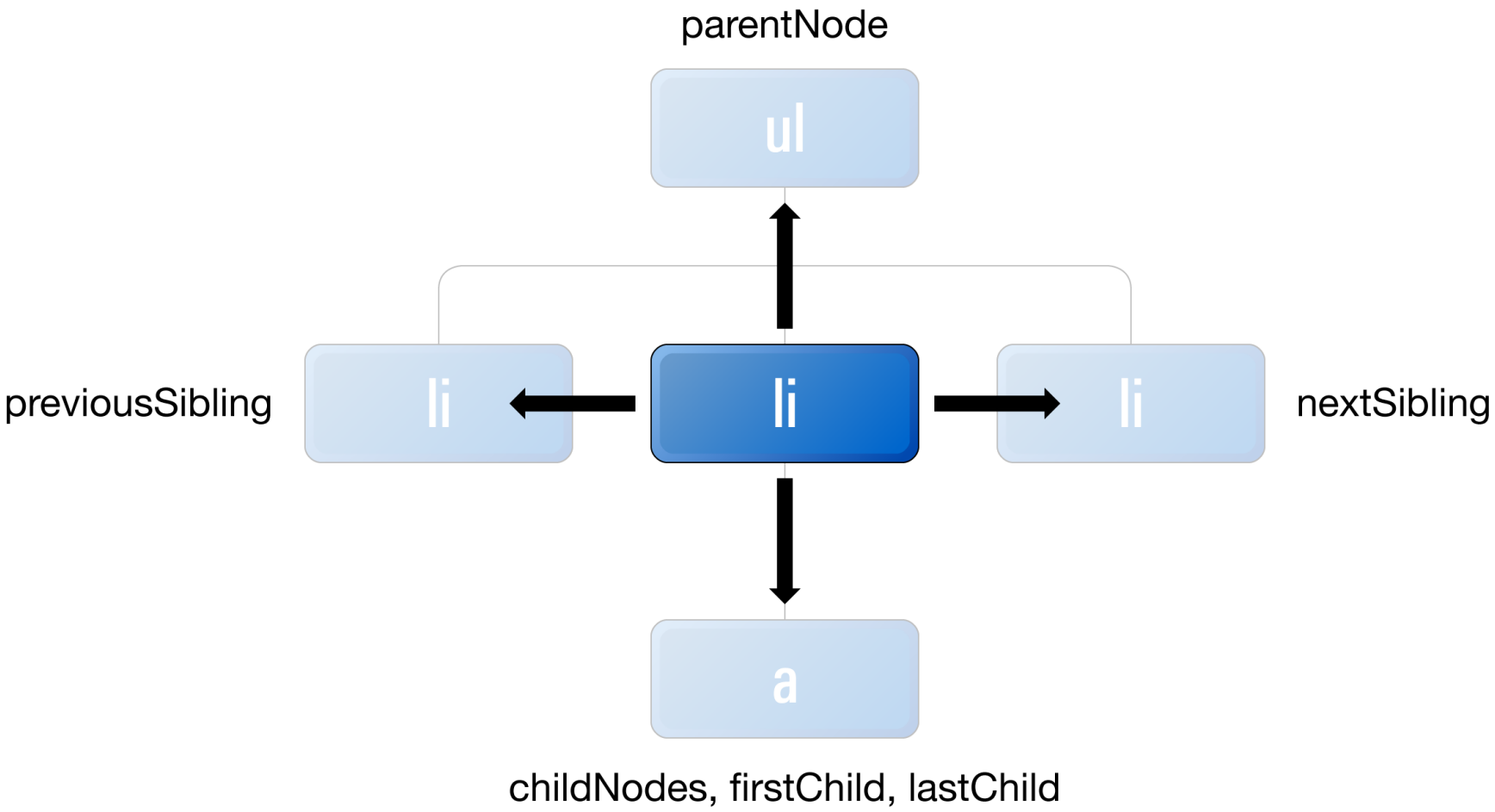
- Getting Attributes
- Setting Attributes

# Finding a Parent

- Finding a parent node

```
<p>
   <a id="oliver" href="someURL">Oliver Twist</a>
</p>

var myOliver = $("oliver");
var myPara = myOliver.parentNode;
```

# Visual Relationships



# Finding Children and Siblings

- Finding children nodes

```
<ul id="baldwins">
      <li>Alec</li>
      <li>Daniel</li>
      <li>William</li>
      <li>Stephen</li>
</ul>

var baldwins = $("baldwins");

var alec = baldwins.firstChild;
var stephen = baldwins.lastChild;

var william = baldwins.childNodes[2];
var stephen = william.nextSibling;
var daniel = william.previousSibling;
```

# Finding Elements by Tag Name

- Use getElementsByTagName()

```
var myParaList =
        document.getElementsByTagName("p");

alert(myParaList.length);  // displays "2"

<p>Hi There</p>
<p>How are you?</p>
```

- myParaList is a node list

# Node List

- A node list can be treated much like an Array
- You can use a for-loop to process each item in list

```
var myParaList =
    document.getElementsByTagName("p");

for (var i=0; i<myParaList.length; i++)
{
    alert(myParaList[i].firstChild.nodeValue);
}
```

# Finding Elements by Name

- Use getElementsByName()

```
<input type="radio" name="choice" checked value"Y">Yes
<input type="radio" name="choice" value"N">No

var targets = document.getElementsByName("choice");

alert(targets[0].nextSibling.nodeValue);  // displays "Yes"
alert(targets[1].nextSibling.nodeValue);  // displays "No"
```

# Finding Elements by ClassName

- Use getElementsByClassName()

```
<p class="special">Hello</p>
<p class="special">There</p>

var targets =
   document.getElementsByClassName("special");

alert(targets[0].firstChild.nodeValue);  // displays "Hello"
alert(targets[1].firstChild.nodeValue);  // displays "There"
```

# Exercise 6-2

- Do the exercises for this section

  (shown to the right for the link to this presentation)

# Section 6-30

## Element Attributes

# Interacting with Attributes

- Attributes are always associated with a particular Element tag

- There are no DOM functions that retreive a set of attribute nodes.  You must access an Attribute node as part of an Element node

# Getting an Attribute

- Use the getAttribute() function

- Example….

```
<a id="koko" href="http://koko.org">Koko</a>

var koko = $("koko");
var kokoHref = koko.getAttribute("href");
```

The variable kokoHref will now be "http://koko.org"

# Setting an Attribute's Value

- All HTML attributes are writable as well as readable
- You can make dynamic changes happen to your Web page
- Use the setAttribute() function
  - Pass the attribute name and its value

# setAttribute Example

```
<a id="koko" href="http://koko.org">Koko</a>

var koko = $("koko");
koko.setAttribute("title", "Web site for Koko!");
```

Now the HTML is changed to …

```
<a id="koko" title="Web site for Koko!"
        href="http://koko.org">Koko</a>
```

# Exercise 6-3

- Do the exercises for this section

  (shown to the right for the link to this presentation)

# Section 6-40

## Creating and Removing Elements

( Instructor Topic – Not in book )

# Create an Element

- To dynamically create a new element on a page it is a two step process
  1. Create the element into a JavaScript variable
  2. Place the element on the page

- Elements are things like <p>, <li>, , etc.

- To create an element use the DOM method createElement, e.g.

```
var mypara = document.createElement("p");
```

  - This creates a paragraph element
  - Stores a reference to the element in mypara

# Adding a Child Element

- One way to place a dynamically created element on a page is with the appendChild
- First get a reference to a parent node and add another child element to the end e.g.

```
var mypara = document.createElement("p");
var myparent = $("body");
myparent.appendChild(mypara);
```

- We are still not finished…

# Adding Text to the Paragraph

- In the last slide, I created an empty paragraph and added to the end of elements in the body.
- To add text to an paragraph you first must use the createTextElement method and then append that to the paragraph element, e.g.

```
var myText = document.createTextNode("Hello World!");
mypara.appendChild(myText);
```

# Removing Elements

- To remove elements use the removeChild method
  - NOTE:  If an Element that you remove has children then all children are removed as well

```
var message = $('msg');
var container = message.parentNode;
container.removeChild(message);
```

# Exercise 6-4

- Do the exercises for this section

  (shown to the right for the link to this presentation)

# End of Lesson 6