JavaScript, jQuery, and the Document Object Model

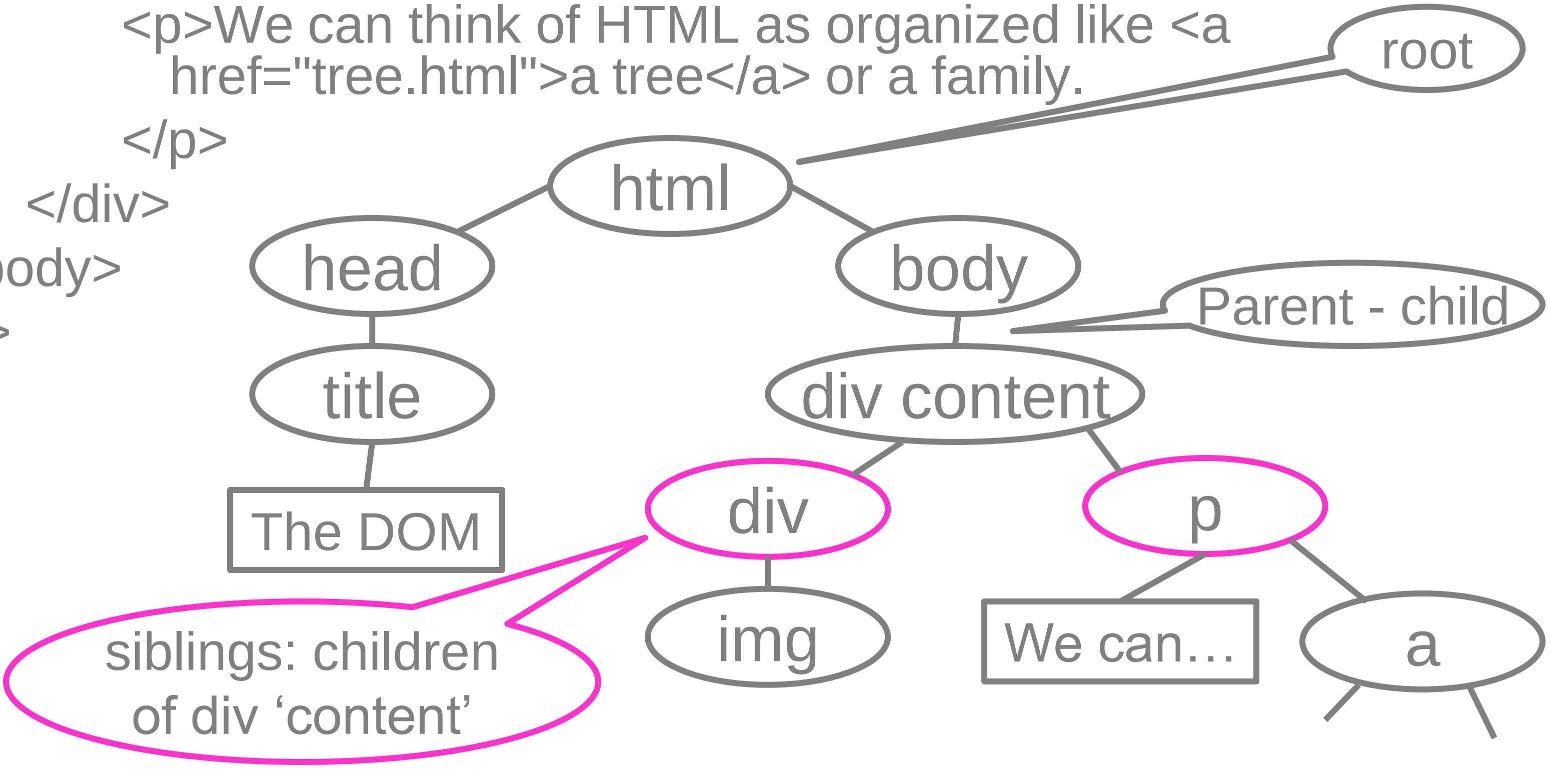
INFO/CS 2300:
Intermediate Web Design and Programming

HTML organization

One (useful) way of thinking about an HTML document is as a "tree" of "nodes".

We also use family language such as parents, children, siblings.

```
<html>
    <head>
        <title>The DOM</title>
    </head>
    <body>
        <div id="content">
            <div id=“myimg">
                <img src=“tree.gif" alt=“HTML Tree" />
            </div>
            <p>We can think of HTML as organized like <a
              href="tree.html">a tree</a> or a family.
            </p>
        </div>
    </body>
</html>
```



The DOM

Everything is an object

Your browser represents every node in the tree as a JavaScript object. The attributes of each tag are properties/fields of the object; the objects have methods that allow easy manipulation.

The top-level object is “document”.

~~How do I access the node I want?~~

~~The document object has methods “getElementById” and “getElementsByTagName”.~~

~~E.g.~~

~~var node = document.getElementById(“photoid”);~~

~~var nodearray = document.getElementsByTagname(“img”);~~

jQuery

jQuery is the most popular JavaScript library in use today

Getting jQuery

http://jquery.com/download/

Current versions 1.11.2 or 2.1.3

Compressed (min) version is fine unless you want to read the code

2.x doesn't support IE 6-8 and that's fine

Using jQuery

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>JavaScript Document Write</title>
    <script src=“js/jquery-1.11.2.min.js” ></script>
  </head>
  <body>
  </body>
</html>
```

Using jQuery from a CDN

```
<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2
  /jquery.min.js"></script>
```

If you source from a Content Delivery Network such as Google, the browser may already have the file cached.

Accessing nodes via jQuery

Pretty much any kind of CSS selector will do. These select everything that matches

```
$(“p”)
$(“.myclass”)
$(“div .header”)
$(“.mylist li”)
$(“#myid”)
```

Saving nodes in jQuery

You can save nodes accessed via jQuery in variables.

$ is part of the jQuery syntax

```
$myNode = $("#tableid");
```

$ is a convention in jQuery/JavaScript, not required as in PHP!

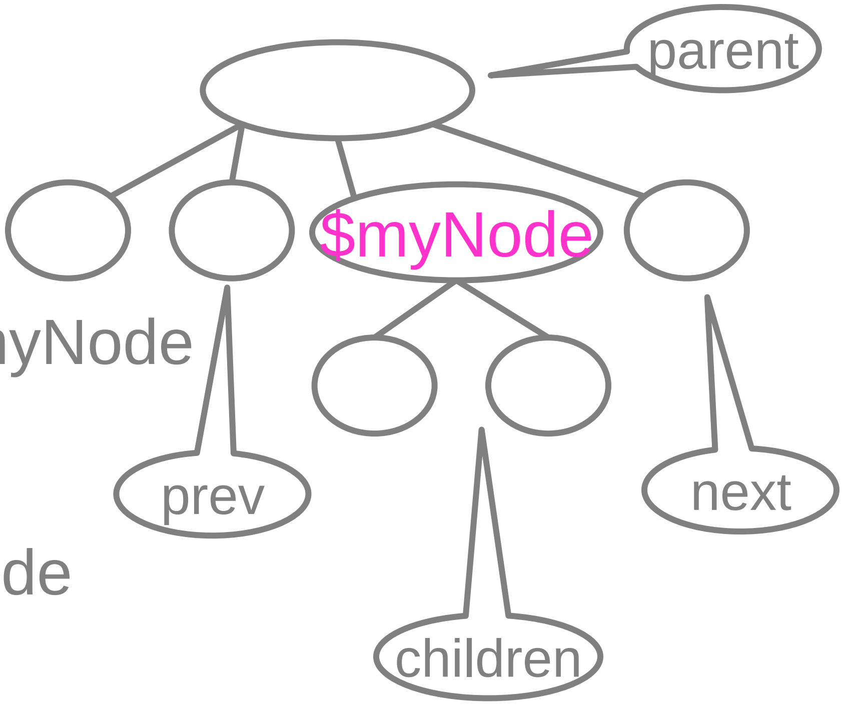
Access other nodes

$myNode.parent()
gives the parent of $myNode

$myNode.prev()
gives the previous sibling of $myNode

$myNode.next()
gives the next sibling of $myNode

$myNode.children()
gives an array of all the children of $myNode (in order)



Nodes have properties

$myImgNode.attr('src')

the src atttribute of an image node

$myNode.text()

the string of all the text contained in $mynode

$myTextInput.val()

the string of text entered in a text input area

$myNode.css('property_name')

The value of the nodes CSS property given by property_name

Changing values

$myImgNode.attr('src', newImgFile)
sets the src of $myImgNode to *newImgFile*.

$myTextNode.text(message)
sets the text of $myTextNode to *message*.

$myNode.html(newHtml)
sets the html of $myNode to *newHtml*.

$myNode.css(‘property’, ‘value’)
sets the CSS *property* of $myNode to *value*.
Ex: $myNode.css(‘background-color’, ‘#000000’)

jQuery Events

```
function myFunction() {
   console.log( "Testing myFunction" );
}
$myImgNode.mouseover( myFunction );
$myImgNodes.mouseover( myFunction );
```

Also: .click, .submit, .mouseout, etc.

Click in!

What do you think would be the effect of adding () to the end of the function name?

$myImgNode.click(myFunction());

A. No effect – same as without ()

B. myFunction won't run

C. Node click event is set to the return value

D. This line will cause a JavaScript error

Click in!

What do you think would be the effect of adding () to the end of the function name?

```
$myImgNode.click( myFunction() );
```

But what if myFunction needs a paramter?

A. No effect – same as without ()

B. myFunction won't run

C. Node click event is set to the return value

D. This line will cause a JavaScript error

Anonymous functions

You can write an event handler directly without using a named function by using *anonymous* functions.

E.g.

```
$myNode.click( function () {
        console.log(“Testing myNode.click”);
});
```

Functions as parameters

```
function myFunction( param ) {
   console.log( "Testing myFunction" );
}

$myNode.click( myFunction( valueOfParam ) );

$myNode.click( function () {
   myFunction( valueOfParam );
});
```

Can't do this

Wrapping the call to myFunction in an anonymous function allows a parameter

Chaining

Can string several method calls together.

E.g.

```
$("#p1").css("color", "red").slideUp(2000);
```

hide() and show()

```
function myFunction() {
   $("#p1").css( "display", "none" );
   $("#p1").css( "display", "" );
}
//Can be written as
function myFunction() {
   $("#p1").hide();
   $("#p1").show();
}
```

Using the DOM: an example

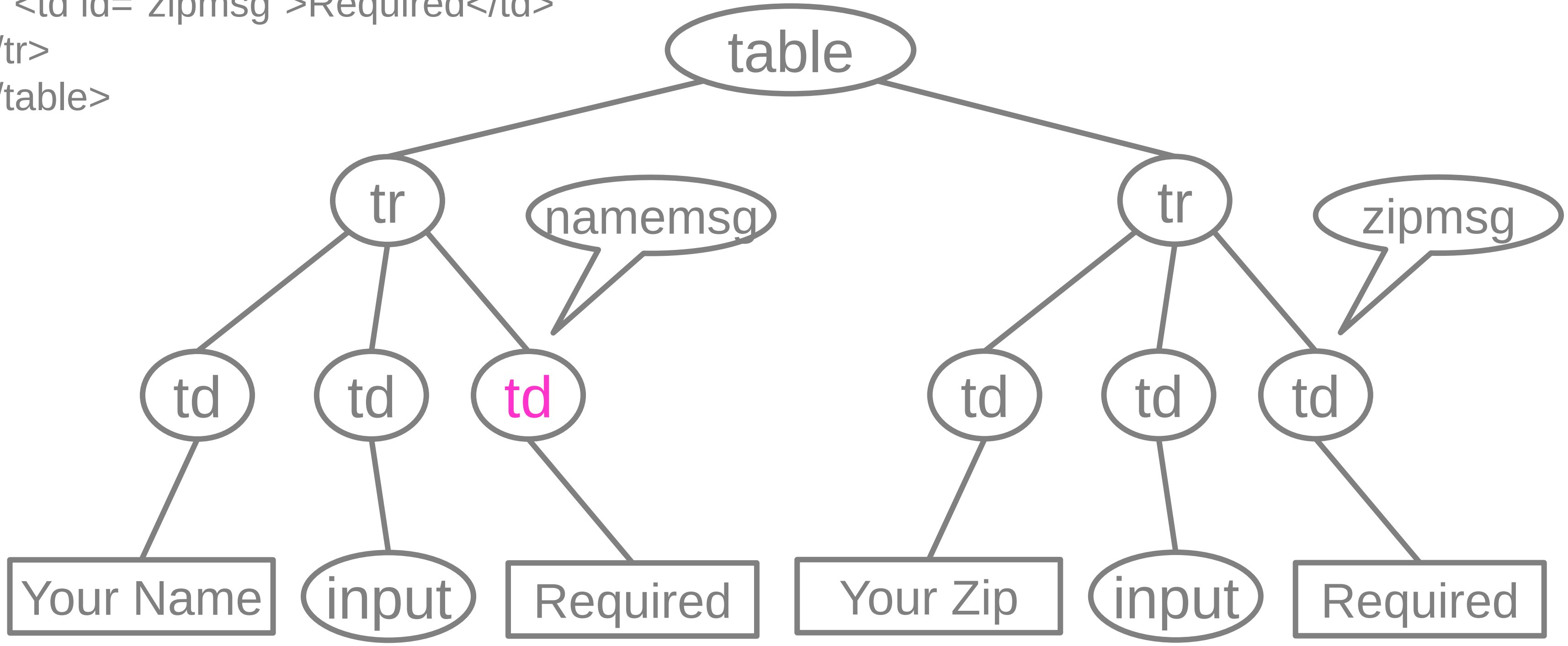
The msg function example

In the form checker we showed last week, we used a function msg(idname, message) to alter the text in a table.

```
<table>
    <tr>
        <td>Your name:</td>
        <td><input type="text" id="name"
            onchange="validName( this.value );">
        <td id="namemsg">Required</td>
    </tr>
    <tr>
        <td>Your zip code:</td>
        <td><input type="text" id="zip“
                onchange="validZip( this.value );">
        <td id="zipmsg">Required</td>
    </tr>
</table>
```

remember "this"

```
<table>
<tr>
    <td>Your name:</td>
    <td><input type="text" id="name“ onchange="validName(this.value);" />
    <td id="namemsg">Required</td>
</tr>
<tr>
    <td>Your zip code:</td>
    <td><input type="text" id="zip" onchange="validZip(this.value);" />
    <td id="zipmsg">Required</td>
</tr>
</table>
```



Msg

Now write the code for the msg function.

```
function msg(idname, message) {







}
```

```
function msg(id,message) {

    if (trim(message) == "") {
            /* Set message to non-breakable space */
            message = String.fromCharCode(160);
    }
    $("#"+id).text(message);

}
```

Altering the document

Altering the document

The DOM also has functions that allow us to alter the markup by creating new nodes, moving nodes around, etc.

Putting new nodes in the tree

$node.append($othernode)

adds *othernode* as the last child below node. $othernode can also be HTML.

$node.prepend($othernode)

adds *othernode* as the first child below node. $othernode can also be HTML.

An example

The booklist example files are on your course server accounts under lecture06

Making this work

We’ll walk through the effects a step at a time.

```
<h2> My Cart </h2>
<p id="cartcontent">Your cart is currently empty.</p>

<table id="carttable">
<thead>
<tr id="carthead"><th class="pict"></th><th class="title">Title</th><th class="author">Author</th<th class="price">Price</th>
</tr>
</thead>
<tbody id="cartbody">
</tbody>
</table>

<h2> Books </h2>
<p> Click on a book to add it to your cart. </p>

<table id="availtable">
<thead>
<tr id="availhead"><th class="pict"></th><th class="title">Title</th><th class="author">Author</th><th
class="price">Price</th>
</tr>
</thead>
<tbody id="availbody">
<tr class=“book”>
			<td class="pict"><img id="php" src="php.jpg" /></td>
			<td class="title">Learning PHP 5</td>
			<td class="author">David Sklar</td>
			<td class="price">$29.95</td>
</tr>
<tr class=“book”>
			<td class="pict"><img id="bpa" src="bpa.jpg" /></td>
			<td class="title">Bulletproof Ajax</td>
			<td class="author">Jeremy Keith</td>
			<td class="price">$34.99</td>
</tr>
```

Adding the event handlers

Suppose we have functions "styleRowOver", "styleRowOut", and "moveRow" that do the right thing on mouse over, on mouse out, and on click. How do we add these to every row in our table with class "book"?

(using styleRowOver, styleRowOut, moveRow)

```
function makeTableRollover() {







}
```

```
function makeTableRollover() {


    $(".book").mouseover(styleRowOver);
    $(".book").mouseout(styleRowOut);
    $(".book").click(moveRow);

    cartitems = 0;
    $("#carthead").hide();

}
```

Now let’s write the functions…

```
function styleRowOver() {
                              “#00C0F0”
}

function styleRowOut() {
                              “#FFFFFF”
}
```

```
function styleRowOver() {
   $(this).css(‘background-color”, “#00C0F0”);
}

function styleRowOut() {
    $(this).css( "background-color", “#FFFFFF”);


}
```

Refers to the
node that triggered the
code that got us
here

Another way to do this

In the makeTableRollover function:

```
$(“.book”).mouseover( function () {
   $(this).css(‘background-color’, ‘#00C0F0’);
});
```

```
function moveRow() {









}
```

```
function moveRow() {
    if ($(this).parent().attr('id') == "cartbody") {
          $("#availbody").prepend($(this));
          $(this).css('background-color','#FFFFFF');
          cartitems--;
    } else {
          $("#cartbody").prepend($(this));
          $(this).css('background-color','#FFFFFF');
          cartitems++;
    }
}
```

How do we get the js to run?

Use one of these in the books.js file

```
window.onload = makeTableRollover;


$(document).ready(function(){
  makeTableRollover();
});
```

Runs after all content including images are loaded

Runs as soon as the HTML is loaded but does not wait for images

Review

- Everything on your webpage is an object. You can use the properties and methods of the objects to:
  - Change properties (styling, text, event handlers)
  - Add new nodes
  - Move nodes

- Project 2 due Tuesday Feb 18 at 5 pm