

Document Object Model, JavaScript and jQuery

INFO/CS 2300:
Intermediate Web Design and
Programming

2 sheets on half wall

HW0 iClicker Last Call

The HW0 iClicker score will be updated (and announced) soon to show recent registrations.

If your score is still zero, your clicker is not properly registered.

You need to register according to the HW0 instructions using your netID and "not using Blackboard." After this Friday the HW0 score will not be updated again. You will still be able to register your clicker and get participation credit, just not for HW0.

Debugging

```
<?php
```

```
//For scalar variables
```

```
echo "<p>Some Variable: $some_variable</p>";
```

```
//For arrays and objects
```

```
echo '<pre>' . print_r( $_POST, true) . '</pre>';
```

```
?>
```

The DOM

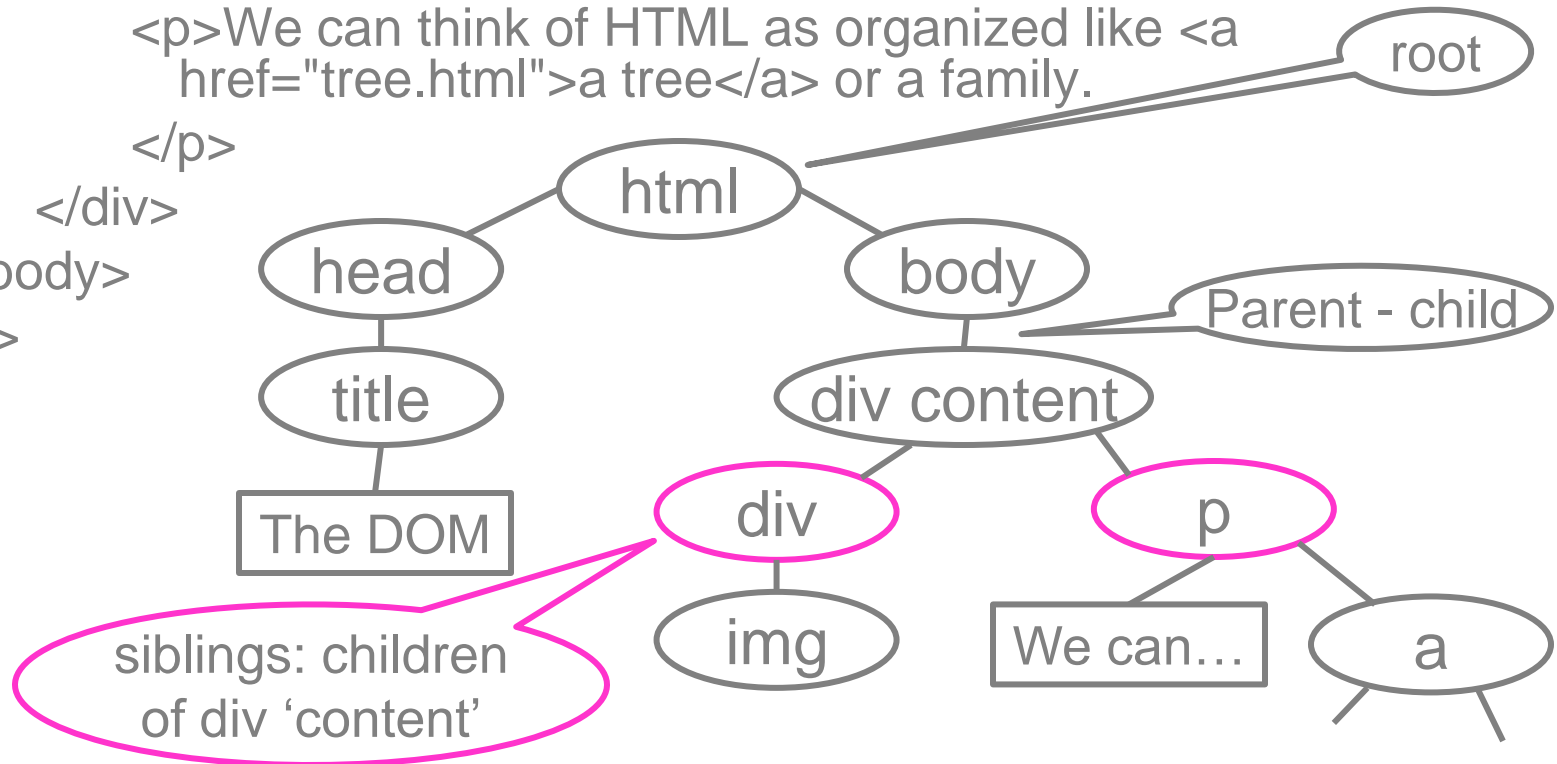
Document Object Model

HTML

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">
        
      </div>
      <p>We can think of HTML as organized like <a
        href="tree.html">a tree</a> or a family.
      </p>
    </div>
  </body>
</html>
```



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”.

Accessing a node

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

E.g.

```
var node = document.getElementById( 'photoid' );
```

```
var nodeArray =
```

```
    document.getElementsByTagName( 'img' );
```

```
var nodeArray =
```

```
    document.getElementsByClassName('someClass');
```


jQuery

A JavaScript library

jQuery

jQuery is the most popular JavaScript library in use today

- Simplifies node access
- Simplifies common tasks
- Unifies code across different browsers
- So popular that the term "Vanilla JavaScript" emerged for plain JavaScript without jQuery

Getting jQuery

<http://jquery.com/download/>

Current version 3.1.1

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

Using jQuery

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="UTF-8">
```

```
    <title>JavaScript Document Write</title>
```

```
    <script src="js/jquery-3.1.1.min.js" ></script>
```

```
  </head>
```

```
  <body>
```

```
  </body>
```

```
</html>
```

Using jQuery from a CDN

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
```

If you source from a [Content Delivery Network](#) such as Google or code.jquery.com, 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" )
```

And more

`$("[attribute='value']")`

`$(":checkbox")`

`$(":checked")`

`$(":even")`

`$(":first")`

`$("parent > child")`

<https://api.jquery.com/category/selectors/>

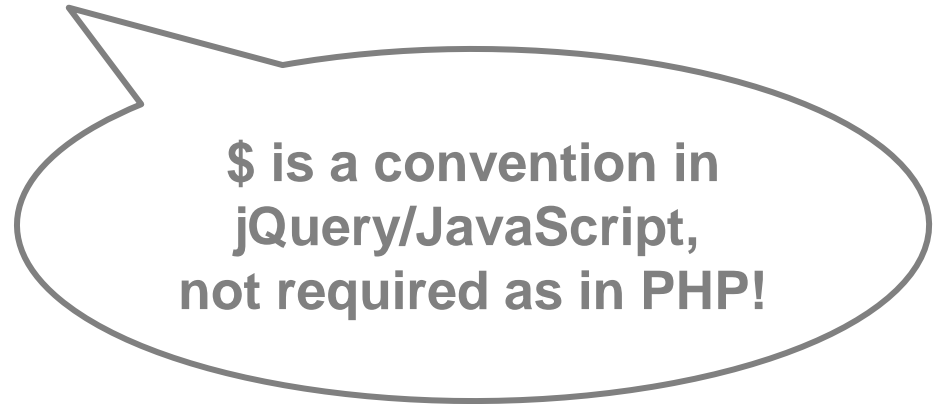
Saving nodes in jQuery

You can save nodes accessed via jQuery in variables.



\$ is part of the jQuery syntax

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



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

Access related nodes

`$myNode.parent()`

gives the parent of `$myNode`

`$myNode.prev()`

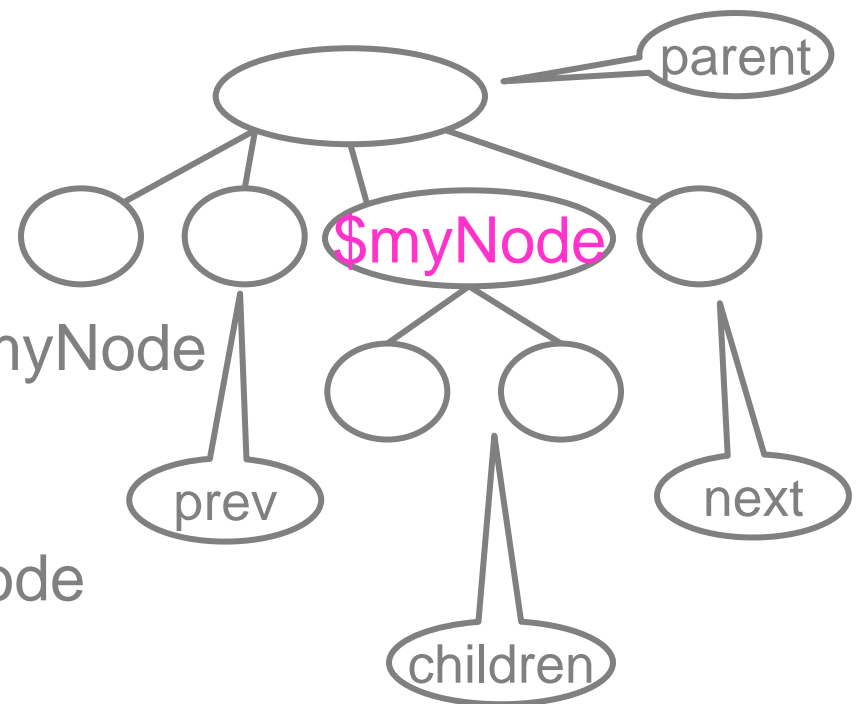
gives the previous sibling of `$myNode`

`$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 attribute 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

`$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*.

Changing styling

`$myNode.css('property', 'value')`

sets the CSS *property* of `$myNode` to *value*.

Ex: `$myNode.css('background-color', '#000000')`

`$myNode.addClass('className')`

CSS in the stylesheet is instantly applied.

Click in!

Click in!

When designing a form input validation, if I want to set the background to red for to indicate a warning, what JS would be best?

- A. `$myNode.addClass('warning')`
- B. `$myNode.addClass('red')`
- C. `$myNode.css('background-color', 'red')`
- D. `$myNode.css('background-color', '#ff0000')`
- E. They are all equivalent

Click in!

When designing a form input validation, if I want to set the background to red for to indicate a warning, what JS would be best?

- A. `$myNode.addClass('warning')`
- B. `$myNode.addClass('red')`
- C. `$myNode.css('background-color', 'red')`
- D. `$myNode.css('background-color', '#ff0000')`
- E. They are all equivalent



Equivalent to inline CSS
Avoid it

Semantics

Two separate decisions

- A warning is needed
- A warning is displayed in red


//The need for a warning is a programmer decision

```
if( Problem ) {  
    $myNode.addClass( 'warning' );  
} else {  
    $myNode.removeClass( 'warning' );  
}
```


Semantics

The decision about how to portray a warning is a design decision carried out in the style sheet.

```
<style>
  .warning {
    background-color: red;
  }
</style>
```



Maybe a warning
should have a
different font color
or borders

Specifications change

What if specifications change so that it becomes important to distinguish between warnings and errors?

Programmer

```
if( bigProblem ) {  
    $myNode.addClass( 'error' );  
} else if( smallProblem ) {  
    $myNode.addClass( 'warning' );  
} else {  
    $myNode.removeClass( 'error' );  
    $myNode.removeClass( 'warning' );  
}
```

Designer

```
<style>
```

```
  .warning {
```

```
    background-color: yellow;
```

```
}
```

```
  .error {
```

```
    background-color: red;
```

```
}
```

```
</style>
```

Responding to Events

Event Based Programming in JS

1. Choose the node(s)
2. Choose the event
3. Write the code for that node-event

Responding to Events

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

```
$( ".myImgNodes" ).click( myFunction );
```

```
$( "#myImgNode").on( "click", myFunction);
```

Also: .mouseover, .submit, .mouseout, etc.

Anonymous functions

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

E.g.

```
$( "#myNode" ).click( function () {  
    alert( "Testing myNode.click" );  
});
```


Debugging JavaScript

```
$( "#myNode" ).click( function () {  
    console.log("Testing myNode.click");  
  
    var $myNode = $( "#myNode" );  
    console.log( $myNode );  
});
```

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();  
}
```

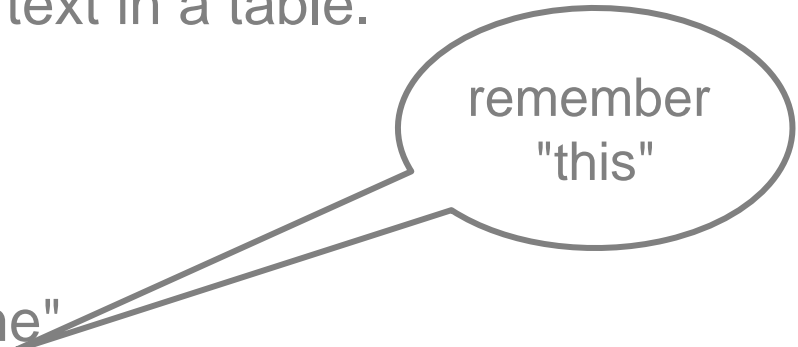
Using the DOM: an example



The msg function example

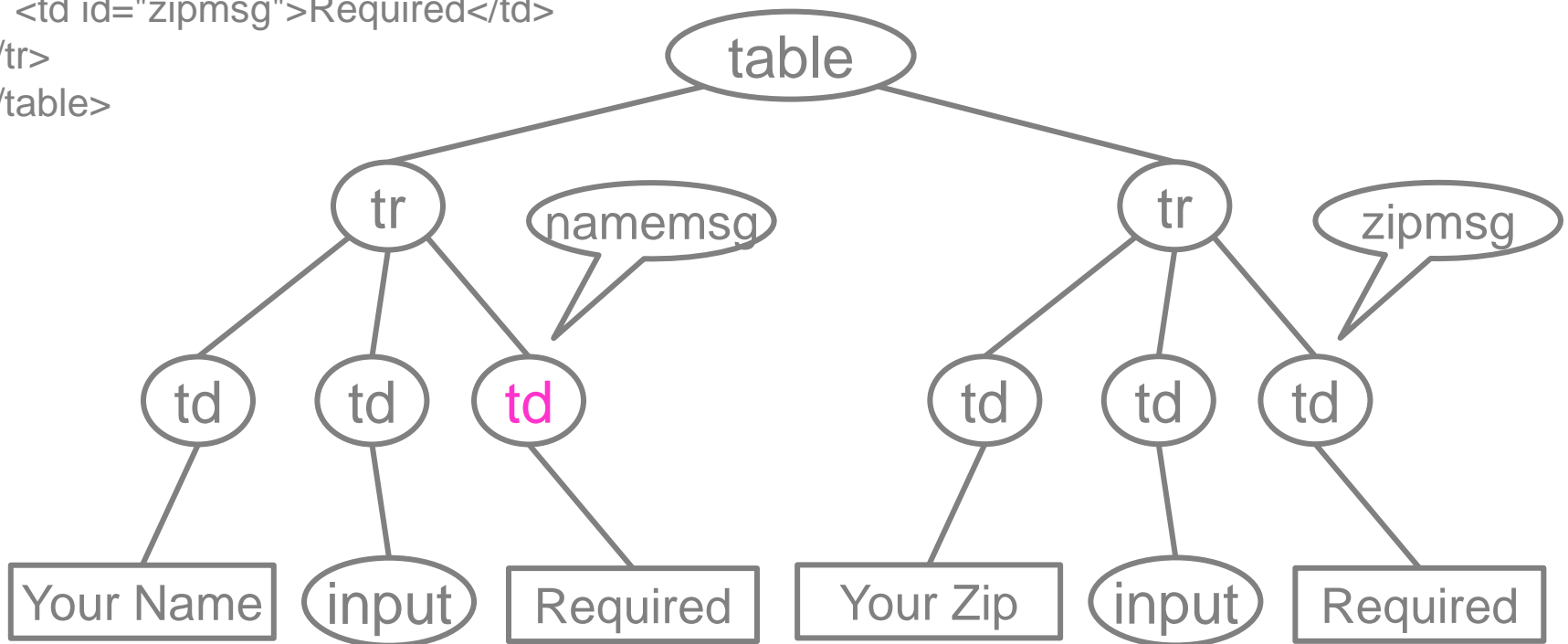
In the form checker we showed last week, we used a function `msg(idname, message)` to alter the text in a table.

```
<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) {
```

```
}
```

//Writes a message to the node with
//the given id

```
function msg( id, message ) {
```

```
  if (trim(message) == "") {
```


```
    /* Set message to non-breakable space */
```

```
    message = String.fromCharCode(160);
```

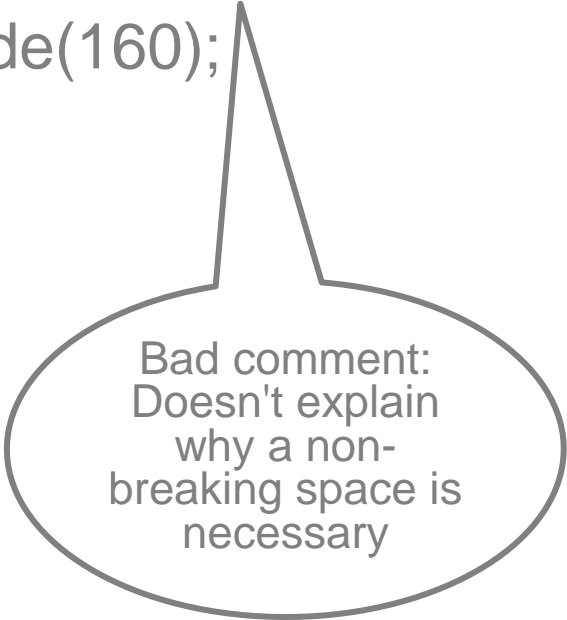
```
  }
```

```
  $( "#" + id ).text( message );
```

```
}
```



Good comment:
Explains the
obscure
CharCode(160)



Bad comment:
Doesn't explain
why a non-
breaking space is
necessary

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="avaiatable">

<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"></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"></td>

<td class="title">Bulletproof Ajax</td>

<td class="author">Jeremy Keith</td>

<td class="price">\$34.99</td>

</tr>

Adding the event handlers

```
$(document).ready(function(){  
    //When the mouse is over a row with the book class an  
    anonymous function adds a "highlighted" class  
  
    //When the mouse leaves any book rows, an anonymous  
    function removes the "highlighted" class  
  
    //Set the moveRow function for all rows so that when the  
    row is clicked, the function runs  
  
    //Initialize the empty cart  
    cartitems = 0;  
    $("#carthead").hide();  
});
```

Adding the event handlers

```
$(".book").mouseover(function() {  
    $(this).addClass('highlighted');  
});
```

```
$(".book").mouseout(function() {  
    $(this).removeClass('highlighted');  
});
```

```
$(".book").click(moveRow);
```

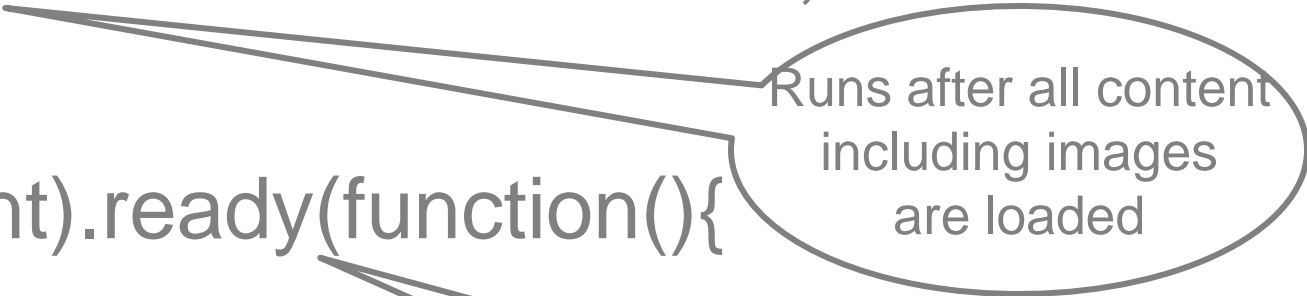

moveRow()

```
function moveRow() {  
    if ( $( this ).parent().attr( 'id' ) === "cartbody" ) {  
        $( "#availbody" ).prepend( $( this ) );  
        cartitems--;  
    } else {  
        $( "#cartbody" ).prepend( $( this ) );  
        cartitems++;  
    }  
  
    if ( cartitems >= 1 ) {  
        $( "#carthead" ).show();  
        $( "#cartcontent" ).text( "Click on a book to remove from cart." );  
    } else if ( cartitems == 0 ) {  
        $( "#carthead" ).hide();  
        $( "#cartcontent" ).text( "Your cart is currently empty." );  
    }  
}
```

How do we get the js to run?

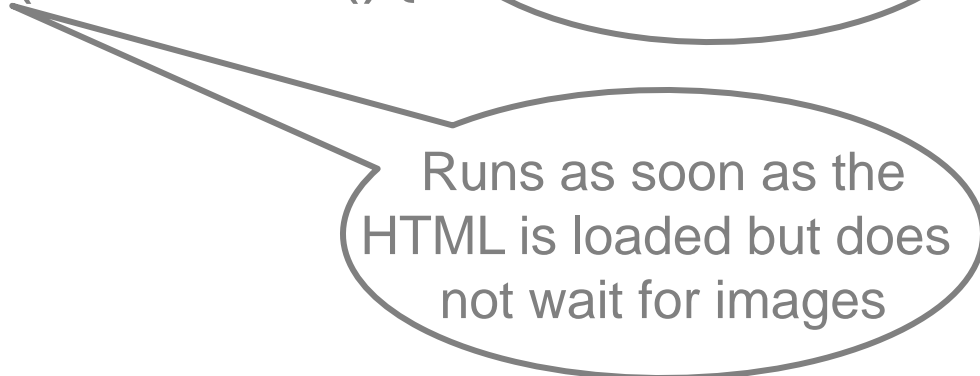
Use one of these in the books.js file

```
window.onload = someFunction;
```



Runs after all content including images are loaded

```
$(document).ready(function(){  
    //do stuff  
});
```



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 28 at 5 pm