IN4MATX 133: User Interface Software

Lecture 6:
DOM Manipulation

Today’s goals

By the end of today, you should be able to…

- Describe the different roles JavaScript has in client-side and server-side development
- Explain the role of the Document Object Model (DOM)
- Write code which edits the DOM using built-in JavaScript functions and jQuery

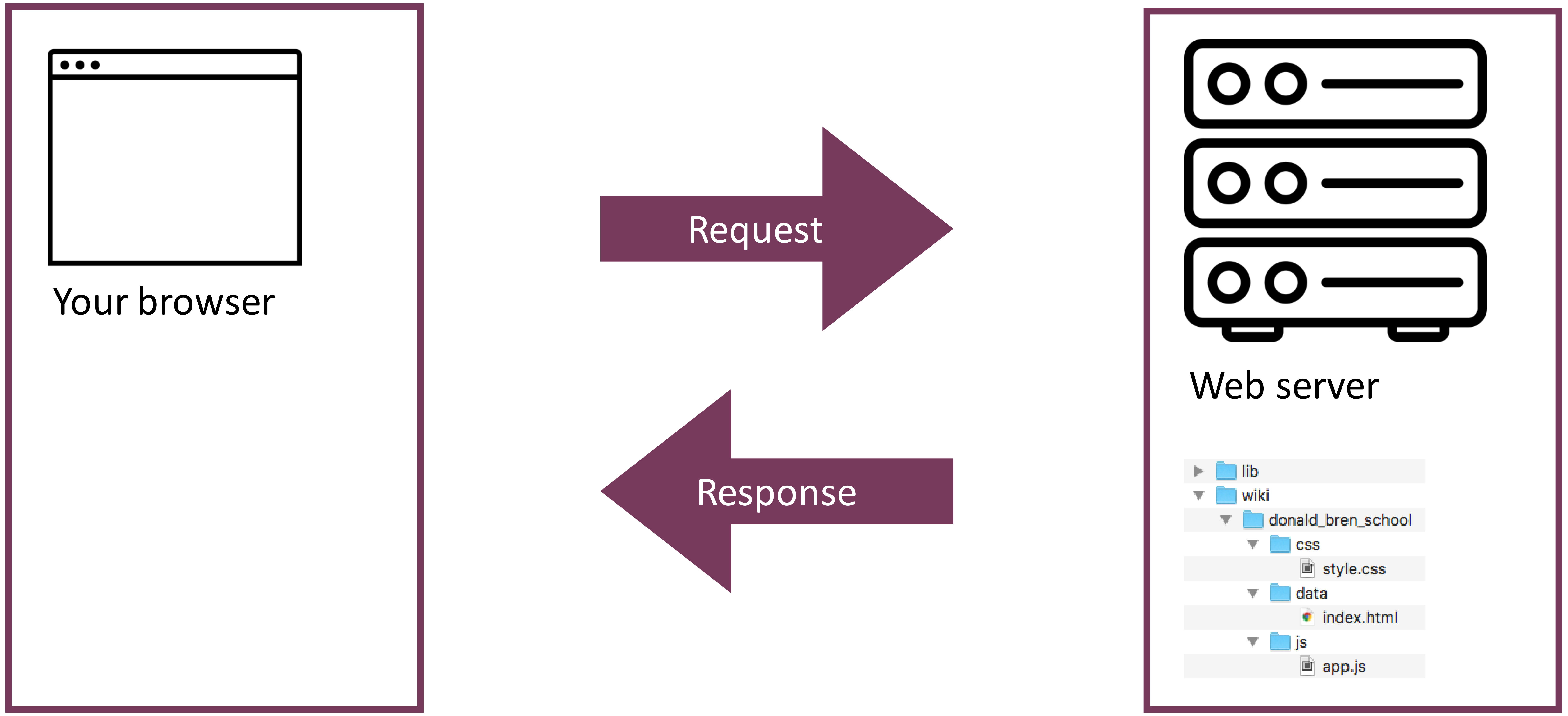
Thus far, JavaScript looks just like any other language

What about JavaScript makes it used so widely on the web?

JavaScript has many functions for editing webpages

Today, JavaScript is used both client-side and server-side

Client-side and server-side JavaScript



Edit what's being rendered
Trigger or react to events

Navigate file system programmatically
Dynamically generate pages or views
Transport, store, or interact with data

Client-side

- Can be seen by the user
- Changes happen in real-time in the browser
- Examples: AJAX, Angular, React, Vue.js

Server-side

- Cannot be seen by the user
- Changes happen on the server in response to HTTP requests
- Examples: Node, ASP.NET

It can be confusing to follow your code if JavaScript is on both sides

Client-side object: Window

- The window object refers to the browser itself.
  You can access properties and execute functions on it

```
/* example properties */
var width = window.innerWidth;    //viewport width
var height = window.innerHeight; //viewport height

/* example functions */
window.alert("Boo!");

var confirmed = window.confirm("Are you sure?");
var quest = window.prompt("What is your quest?”);
```

↑

Bad form, put it on your page instead

Client-side object: Window

- It’s possible to use window to control the browser, but behavior varies drastically by browser

```
var xPos = window.screenX; //offset from screen edge
var yPos = window.screenY; //offset from screen edge
var scroll = window.scrollY; //pixels scrolled down
var url = window.location.href; //url for this page

window.scrollTo(0,1000); //scroll to position
window.open(url); //open a new window loading the URL
window.close(); //close window
```

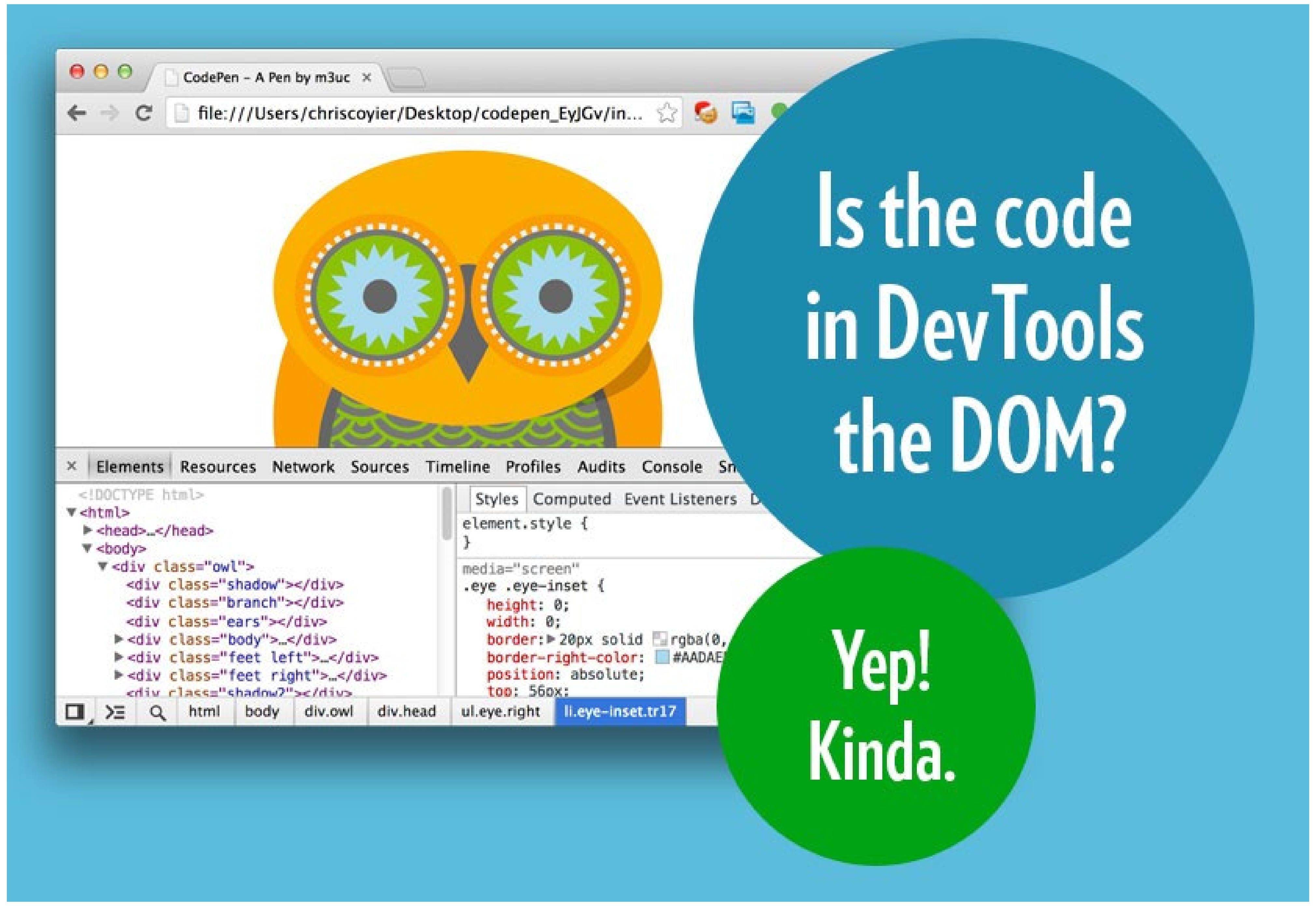
Again, better to keep your program inside the window

Server-side: has no Window

- No window object exists in server-side JavaScript
- On a server, there would be nothing to scroll to and no one to alert
- We will see server-side development more next week

HTML Document Object Model (DOM)

- “A standard for how to get, change, add, or delete HTML elements”
- “the HTML you write is parsed by the browser and turned into the DOM”
- Client-side JavaScript can then edit the DOM
  - Server-side JavaScript might specify what HTML to show, but will not edit the DOM


CodePen – A Pen by m3uc
file:///Users/chriscoyier/Desktop/codepen_EyJGv/in...
Elements Resources Network Sources Timeline Profiles Audits Console
Styles Computed Event Listeners
html body div.owl div.head ul.eye.right li.eye-inset.tr17
Is the code in DevTools the DOM?
Yep! Kinda.

```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
    ▼<div class="owl">
       <div class="shadow"></div>
       <div class="branch"></div>
       <div class="ears"></div>
      ▶<div class="body">…</div>
      ▶<div class="feet left">…</div>
      ▶<div class="feet right">…</div>
       <div class="shadow2"></div>
```

```
element.style {
}
media="screen"
.eye .eye-inset {
   height: 0;
   width: 0;
   border:▶ 20px solid rgba(0,
   border-right-color: #AADAE
   position: absolute;
   top: 56px;
```

https://css-tricks.com/dom/

JavaScript in HTML

- Can insert inline using the `<script>` tag

```
<head>
  <script>
    alert("Hello, world!");
  </script>
</head>
<body>

</body>
```

JavaScript in HTML

- Your script should wait until after the page has loaded
    - Otherwise elements you’re trying to access might not exist

```
<head>
 <script>
  function pageLoaded() {
   alert("Page Loaded!");
  }
 </script>
</head>
<body onload="pageLoaded();">

</body>
```

JavaScript in HTML

- Functions can respond to events

```
<head>
  <script>
    function buttonClicked()  {
      alert("Button Clicked!");
    }
  </script>
</head>
<body>
  <!--Bad style! Don't do this-->
  <button onclick="buttonClicked()">Click me!</button>
</body>
```

JavaScript in HTML

- Like CSS, better to load an external script

```
<head>
  <script src="js/script.js"></script>
</head>
```

Editing the DOM

- document object model
- Write into the DOM with `document.write`

```
<script>
  document.write("<h1>Hello, World!</h1>");

  var myCourse = "IN4MATX 133";
  document.write("<h1>Hello, " + myCourse + "!");
</script>
```

Selecting elements

- We can reference individual HTML elements by calling selector functions

```
//element with id="foo"
var fooElem = document.getElementById('foo');

//elements with class="row"
var rowElems = document.getElementsByClassName('row');

//<li> elements
var liElems = document.getElementsByTagName('li');
```

Selecting elements

- We can reference individual HTML elements by calling selector functions

```
/*easiest to select by reusing CSS selectors! */
var cssSelector = 'header p, .title > p';

//selects FIRST element that matches css selector
var elem = document.querySelector(cssSelector);

//matches ALL elements that match css selector
var elems = document.querySelectorAll(cssSelector);
```

Editing elements

- Properties and functions of elements can manipulate them

```
/* properties to access the CONTENT of the element */

var elem = document.querySelector('p');

var text = elem.textContent; //the text content of the elem
elem.textContent = "I'm different!"; //change the content


var html = elem.innerHTML; //content including tags
elem.innerHTML = "I'm <strong>different</strong>";

var parent = elem.parentNode; //get the parent node
```

Editing elements

- Can add/remove classes, IDs, and inline style

```
<style>/*Bad form! Just for demo*/
  .title {
   font-style: italic;
  }
</style>
<h1>Hello, IN4MATX 133!</h1>
<script>
  var elements = document.getElementsByTagName("h1");
  for(var i = 0; i < elements.length; i++) {
   elements[i].classList.add("title");
   elements[i].style.color="blue";
  }
</script>
```

Changing the DOM

```
//create a new <p> (not yet in the tree)
var newP = document.createElement('p');
newP.textContent = "I'm new!";

//create Node of textContent only
var newText = document.createTextNode("I'm blank");

var div = document.querySelector('div#container');
div.appendChild(newP); //add element INSIDE (at end)
div.appendChild(newText);

//add node BEFORE element (new, old)
div.insertBefore(document.createTextNode("First!"), newP);

//replace node (new, old)
div.replaceChild(document.createTextNode('boo'), newText);

//remove node
div.removeChild(div.querySelector('p'));
```

Validating data

- Check form fields before sending to a server
  - Provide instant feedback, reduce server back-and-forth

```
<script>
 function validateForm() {
  var x = document.forms["myForm"]["fname"].value;
  if(x==null || x=="") {
   alert("Name must be filled out");
   return false;
  }
 }
</script>
<form name="myForm" onsubmit="return validateForm()" method="post">
 <label>Name: </label>
 <input type="text" name="fname">
 <input type="submit" value="Submit">
</form>
```

Gather and use information

- Remember: this is client-side!

```
<script>
 var x = document.getElementById("demo");
 function getLocation() {
     if (navigator.geolocation) {
         navigator.geolocation.getCurrentPosition(showPosition);
     } else {
         x.innerHTML = "Geolocation is not supported by this browser.";
     }
 }
 function showPosition(position) {
     x.innerHTML = "Latitude: " + position.coords.latitude +
     "<br>Longitude: " + position.coords.longitude;
 }
</script>
```

How do we make interactive pages?

Listeners

- Can attach a listener to that method, specifying that we want to do something when that element causes an event

```
//respond to "click" events
elem.addEventListener('click', callback);

//often use an anonymous callback function
elem.addEventListener('click', function(){
  console.log('clicky clicky!');
});
```

Listeners

- Listener callback function will be passed an **event** parameter
  - Sometimes useful, but can often be ignored

```
elem.addEventListener('click', function(event) {
   console.log('You clicked me!');

   //get what was clicked;
   var clickedElem = event.target;
});
```

Remember, parameters are optional

The “target” (source) of the event

Event types

```
'click'      //mouse or button clicked

'dblclick'   //double-clicked

'hover'      //mouse hover

'focus'      //element gains focus (important!)

'mouseenter' //mouse is moved over an element

'mouseleave'  //mouse leaves the element

'mousedown' //mouse button is pressed

'keydown'   //key is pressed

//... and more!
```

https://developer.mozilla.org/en-US/docs/Web/Events

Manipulation in pure JavaScript is verbose

- If you’re editing a lot of tags, your code can get very long and difficult to read

jQuery

- Predefines methods for manipulating the DOM
  - Include <u>before</u> your own script
- Remember:
  - Integrity: hashes to ensure the downloaded file matches what’s expected
  - Crossorigin: some imports require credentials, anonymous requires none

```
<script
  src="https://code.jquery.com/jquery-3.3.1.min.js"
  integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
  crossorigin="anonymous"></script>
```

jQuery selector

- Use the `jQuery()` function to select DOM elements.
  The parameter is a CSS selector String (like `querySelector`)
  - More common to use the `$()` shortcut

```
//selects element with id="foo" (e.g., <p id="foo">)
var fooElem = jQuery('#foo');

//selects all <a> elements (returns an array)
var allLinksArray = jQuery('a');

//selects element with id="foo" (e.g., <p id="foo">)
var fooElem = $('#foo');

//selects all <a> elements (returns an array)
var allLinksArray = $('a');
```

jQuery selector

- jQuery handles all CSS selectors, as well as some additional pseudoclasses

```
$('#my-div') // by id
$('div') // by type
$('.my-class') // by class
$('header, footer') // group selector
$('nav a') // descendant selector
$('p.red') // scoped selector

$('section:first') // first <section> element
                   // not a css selector!
```

jQuery and elements

- Similar to pure JavaScript, jQuery provides methods to access and modify attributes and CSS

```
//Pure JavaScript
var txt = document.querySelector('#my-div').textContent;
document.querySelector('#my-div').textContent = 'new
info!';
document.querySelector('#my-div').innerHTML = '<em>new
html!</em>';

//jQuery
var txt = $('#my-div').text();       //get the textContent
$('#my-div').text('new info!');      //change the
textContent
$('#my-div').html('<em>new html!</em>'); //change the HTML
```

jQuery and the DOM tree

```
//create an element (not in DOM)
var newP = $('<p class="new">This is a new element</p>');

//add content to DOM
$('main').append(newP); //add INSIDE, at end
$('main').append('<em>new</em>'); //can create element on the fly
```

Works without closing tag

```
$('main').prepend('<em>new</em>'); //add INSIDE, at beginning

$('main').before('<header>'); //insert BEFORE

$('footer').insertAfter('main'); //insert target AFTER param
```

Selects existing element, so will move!

```
$('main').wrap('<div class="container"></div>'); //surround

$('footer').remove(); //remove element
$('main').empty(); //remove all child elements
```

jQuery event handling

- jQuery also provides convenience methods for registering event listeners

Like `addEventListener('click')`

```
$('#button').click(function(event) {
    console.log('clicky clicky!');

    //what was clicked
    var element = $(event.target);
});
```

`event.target` is equivalent to `this`

Get as jQuery-style element to call jQuery methods on it ("wrap it")

Document ready: JavaScript

- Remember earlier: your script should wait until after the page has loaded
  - Otherwise elements you’re trying to access might not exist

```
<head>
 <script>
  function pageLoaded() {
   alert("Page Loaded!");
  }
 </script>
</head>
<body onload="pageLoaded();">

</body>
```

Document ready: jQuery

```
$(document).ready(function() {
  //your program goes here
  //this need not be an anonymous function

});
```

Document ready: jQuery

```
//shortcut: just pass the function to the jQuery method
$(function() {
  //your program goes here
  //this need not be an anonymous function

});

//shortest cut: use the abbreviated syntax
$(() => {
  //your program goes here
  //this need not be an anonymous function

});
```

Importing JavaScript

- When your script uses document ready, convention is to load it in the `<head>` tag
  - Important to think about ordering, particularly for libraries
  - e.g., import JQuery before you use it in a script

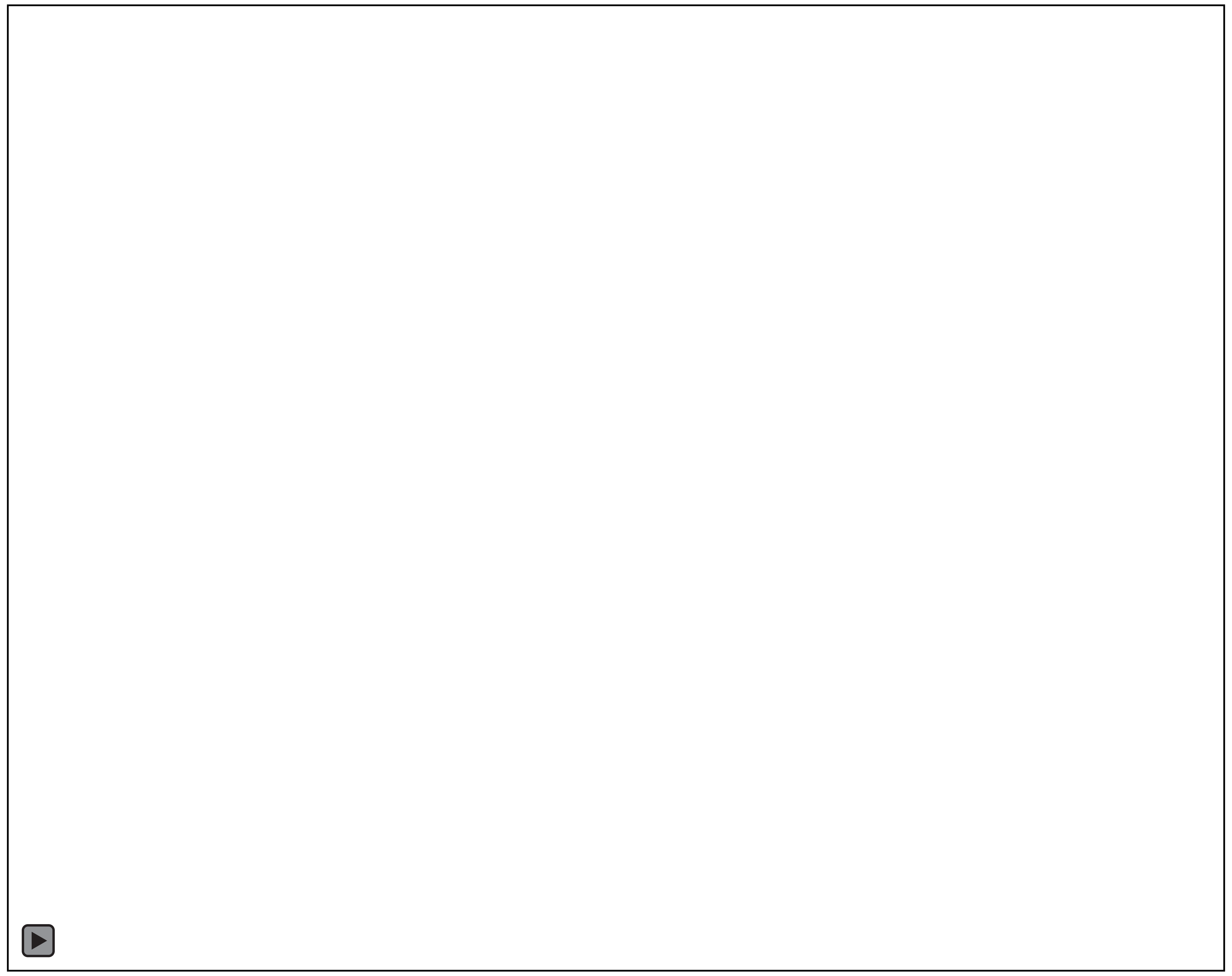
```
<head>
  <script src="https://code.jquery.com/jquery-
3.3.1.min.js"></script>
  <script src="index.js"></script>
</head>
```

jQuery effects

- jQuery supports adding transitions to modify the appearance of effects

```
$('#id').fadeIn(1000);    //fade in over 1 second
$('#id').fadeOut(500);    //fade out over 1/2 sec
$('#id').slideDown(200); //slide down over 200ms
$('#id').slideUp(500);    //slide up over 500ms
$('#id').toggle();        //toggle whether displayed
```

jQuery demo



Today’s goals

By the end of today, you should be able to…

- Describe the different roles JavaScript has in client-side and server-side development
- Explain the role of the Document Object Model (DOM)
- Write code which edits the DOM using built-in JavaScript functions and jQuery

Utility functions

jQuery utility functions

- jQuery includes many utility functions to simplify syntax

```
//check if an item is in an array
$.inArray(4, [3,4,3] );

//this is like .filter, but works on old browsers
$.grep( [3,4,3], function(item) {
   return item > 3;
});

//iterate over arrays or objects -- works for either!
$.each( [1,3,3], function(key, value) {
   console.log('Give me a '+value);
});

$.each( {dept:'IN4MATX',num:'133'}, function(key, value) {
   console.log(key+' name: '+value);
});
```

http://api.jquery.com/category/utilities/

Even more utilities: Lodash

- A handy library for working with basic data structures

```
_.flatten([1, [2, [3, [4]], 5]]);
// => [1, 2, [3, [4]], 5]

var zipped = _.zip(['a', 'b'], [1, 2], [true, false]);
// => [['a', 1, true], ['b', 2, false]]

_.unzip(zipped);
// => [['a', 'b'], [1, 2], [true, false]]
```

https://lodash.com/