Web Development Technologies: The DOM and jQuery

Paul Fodor

CSE316: Fundamentals of Software Development

Stony Brook University

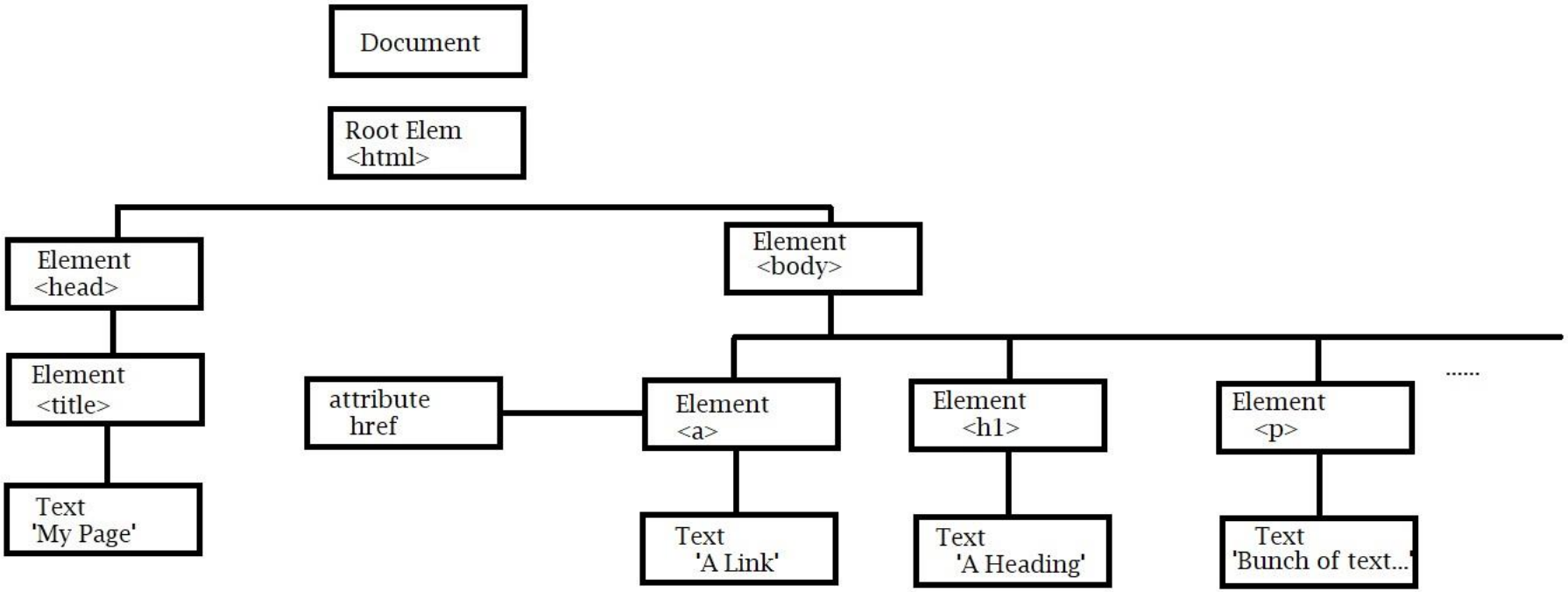
http://www.cs.stonybrook.edu/~cse316

Topics

- The DOM
- jQuery

The DOM

- The Document Object Model (DOM)
  - Tree of objects and attributes created by web browser from structure of web page



The DOM

- Javascript can:
  - Change any HTML element
  - Change any HTML attribute
  - Change any CSS Style
  - Remove any HTML element or attribute
  - Add new HTML elements and attributes
  - React to all existing HTML events
  - Create new HTML events

The HTML DOM

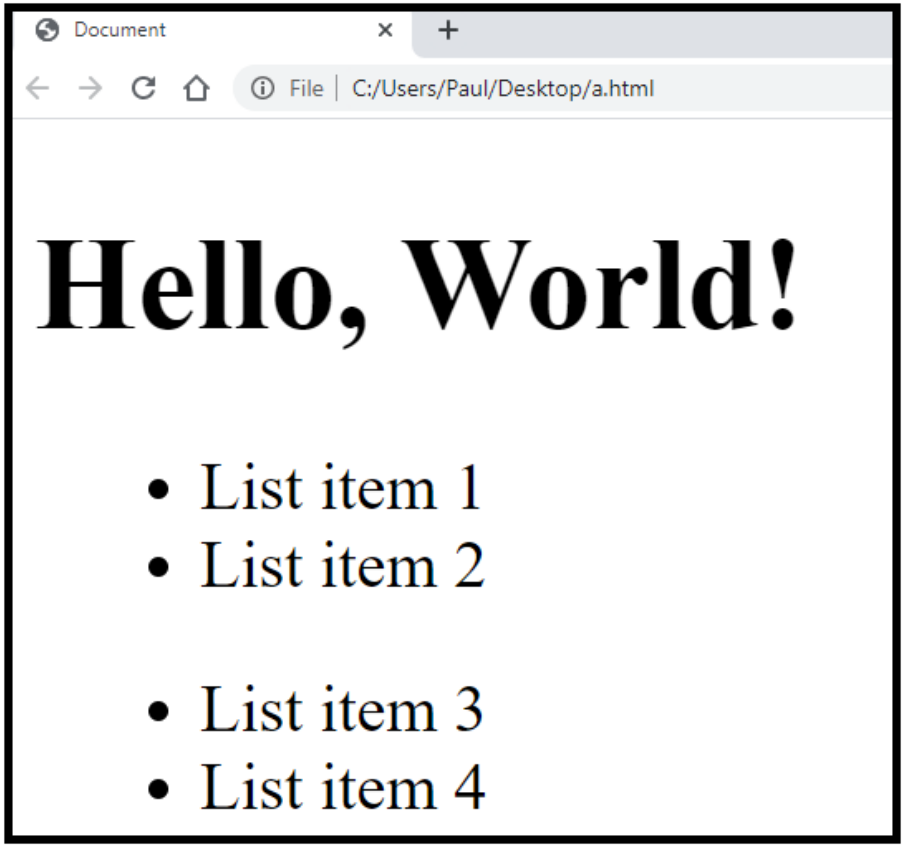
- In this model:
  - HTML elements are objects
  - HTML elements have properties (like members) that can be altered
  - The DOM provides methods that can access all HTML elements
  - The DOM defines events for all HTML elements
- As with OO Paradigm
  - **Methods** are **actions** that can be performed
  - **Properties** are **values** that can be altered

The HTML DOM

- References to any element or the below get methods start with *document*.
  - This represents the document itself.
- Finding elements:
  - **getElementById(id)** – Finds elements with an id matching the argument **(id='id')**
  - **getElementByTagName(name)** – Finds elements based on their tag name **<tag>** matches **<name>**
  - **getElementByClass(name)** – Finds elements with a class matching the argument **(class = 'name')**
- **Note: getElementXXX()** calls may return more than 1 object. Use subscripts to access a specific object.
- Key Properties – 'element' is a variable holding an object returned by one of the above get methods
  - element.innerHTML – This is the HTML content of an element
  - element.attribute – (where 'attribute' is an attribute name)
  - element.style.property – (where 'property' is the name of a style setting)

The HTML DOM Example

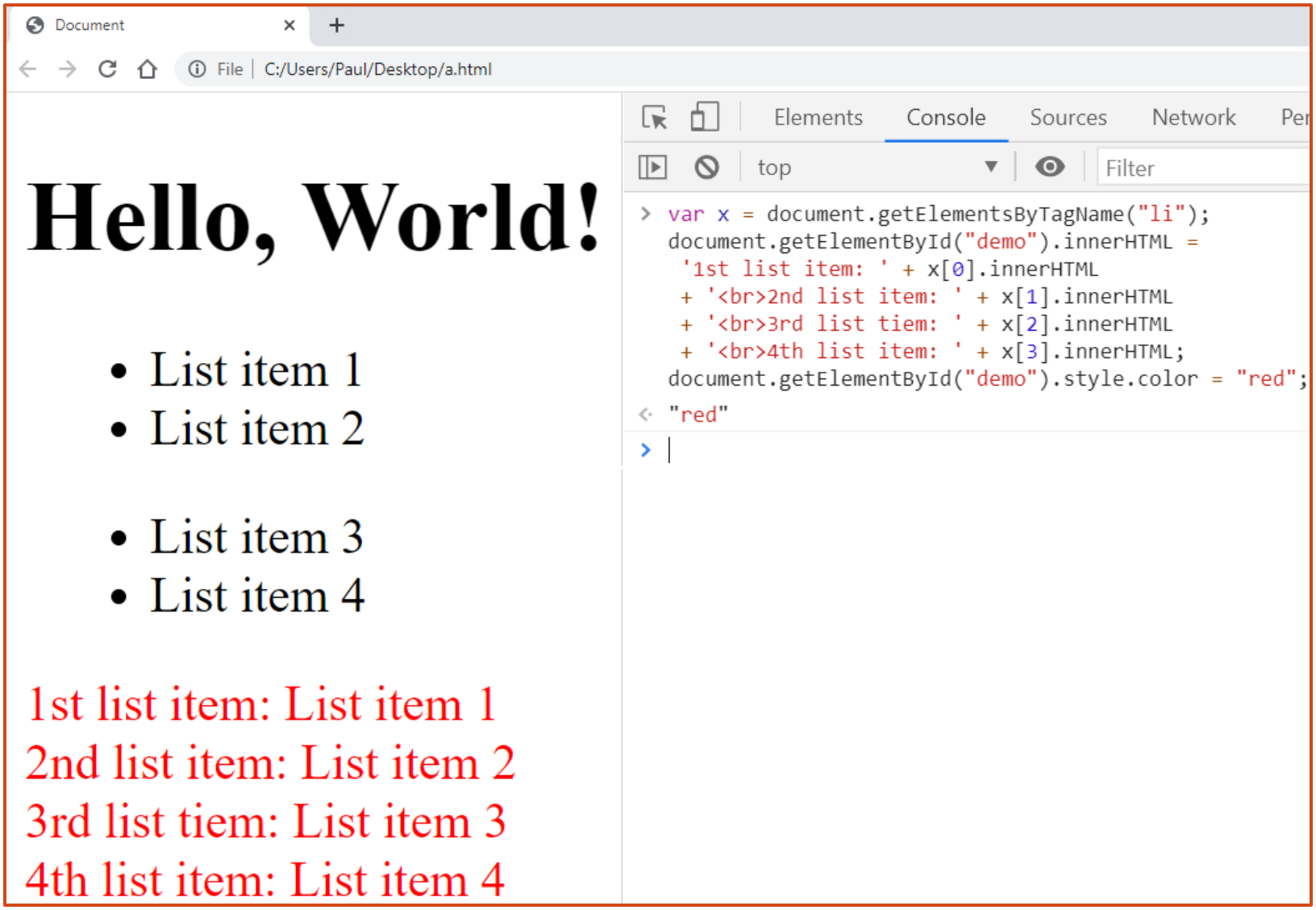
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Document</title>
</head>
<body>
<h1>Hello, World!</h1>
<ul>
<li>List item 1</li>
<li>List item 2</li>
</ul>
<ul>
<li>List item 3</li>
<li>List item 4</li>
</ul>
<p id="demo"></p>
</body>
</html>
```



The HTML DOM Example

```
// in the Console
var x = document.getElementsByTagName("li");
document.getElementById("demo").innerHTML =
 '1st list item: ' + x[0].innerHTML
 + '<br>2nd list item: ' + x[1].innerHTML
 + '<br>3rd list tiem: ' + x[2].innerHTML
 + '<br>4th list item: ' + x[3].innerHTML;
document.getElementById("demo").style.color = "red";
```

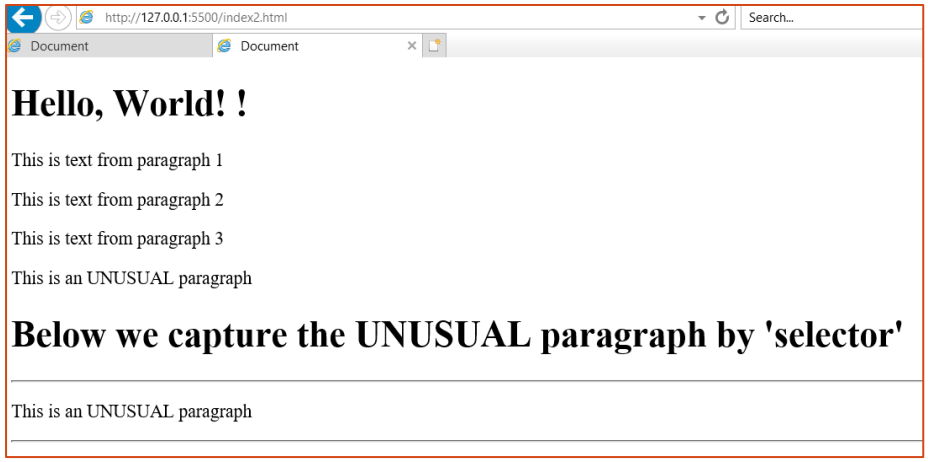
The HTML DOM Example



The HTML DOM Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<meta http-equiv="X-UA-Compatible"
content="ie=edge">
<title>Document</title>
</head>
<body>
<h1>Hello, World! !</h1>
<p id="one">This is text from paragraph 1</p>
<p id="two">This is text from paragraph 2</p>
<p id="three">This is text from paragraph 3</p>
<p class="unusual">This is an UNUSUAL
paragraph</p>
<h1>Below we capture the UNUSUAL paragraph by
A'selector'</h1>
<hr>
<p id="demo"></p>
<hr>
</body>
</html>
```

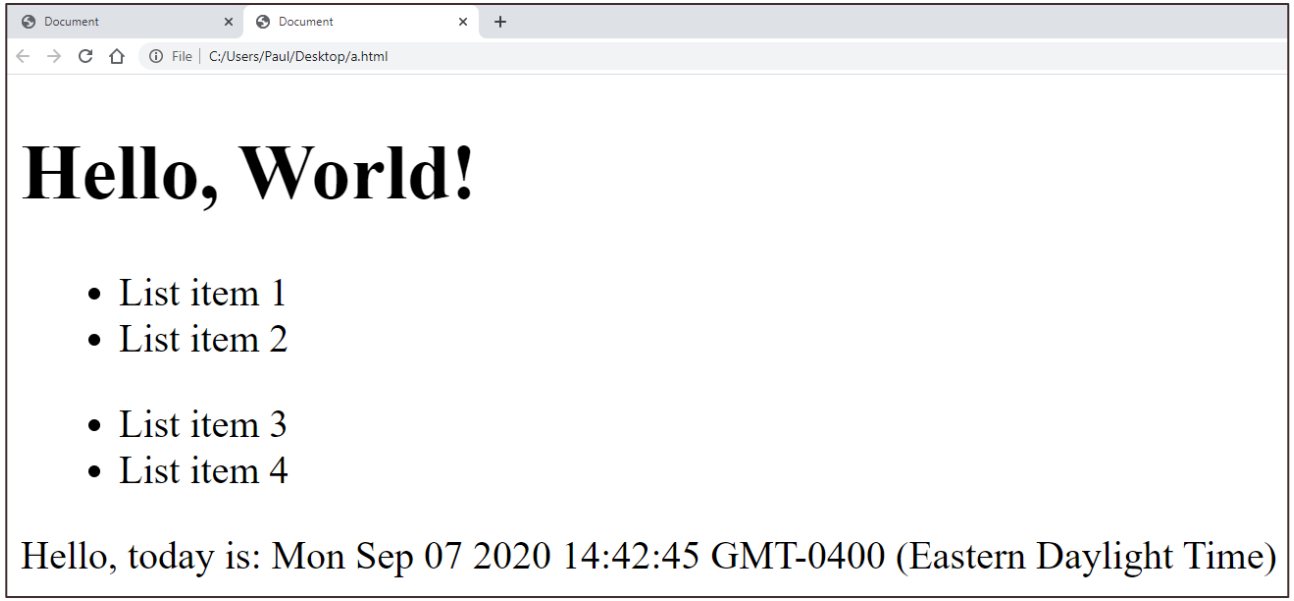
```
var x = document.querySelectorAll("p.unusual");
document.getElementById("demo").innerHTML =
  x[0].innerHTML;
```



The HTML DOM - Writing into the HTML Stream

Javascript document.write() will add text directly into an HTML page

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Document</title>
</head>
<body>
<h1>Hello, World!</h1>
<ul>
<li>List item 1</li>
<li>List item 2</li>
</ul>
<ul>
<li>List item 3</li>
<li>List item 4</li>
</ul>
<p id="demo"></p>
<script>document.write("Hello, today is: " + Date());</script>
</body>
</html>
```



The HTML DOM - Changing Attributes

- You can change an attribute by getting the element and assigning a value to the attribute
- Example:
  - <IMG id="image1" src="Mountain.jpg">
  - Can change the src with:

```
document.getElementById("image1").src =
"Stream.jpg";
```

The HTML DOM – Changing Styles

- You can change style by assigning to any of the style properties on an element
- The format to access a style property is:

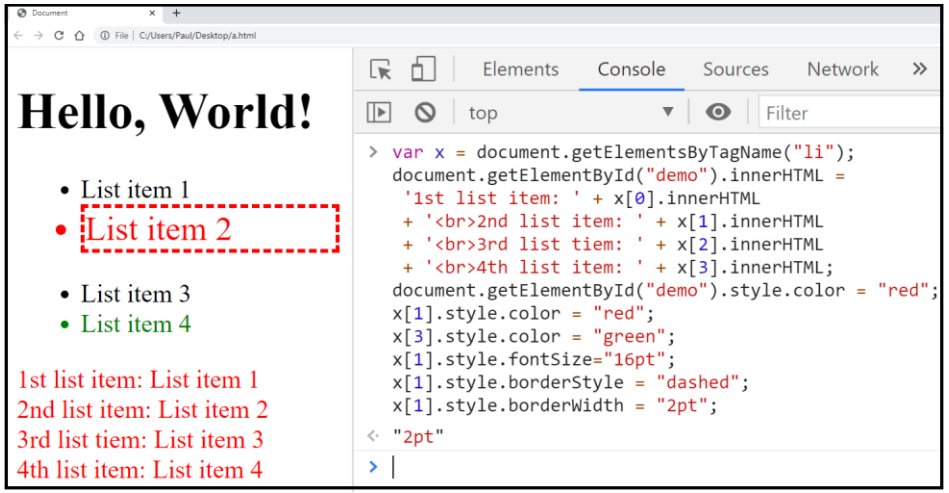
  <object>.style.<propertyname>

  - **<object>** is the object returned by getElementByXXX() call**s**
  - **style** is just a keyword to indicate the upcoming field is a style attribute
  - **<propertyname>** is the name of the CSS property

The HTML DOM – Changing Styles - Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-
width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible"
content="ie=edge">
<title>Document</title>
</head>
<body>
<h1>Hello, World!</h1>
<ul>
<li>List item 1</li>
<li>List item 2</li>
</ul>
<ul>
<li>List item 3</li>
<li>List item 4</li>
</ul>
<p id="demo"></p>
<script>document.write("Hello, today is: " +
Date());</script>
</body>
</html>
```

```
var x = document.getElementsByTagName("li");
document.getElementById("demo").innerHTML =
 '1st list item: ' + x[0].innerHTML
 + '<br>2nd list item: ' + x[1].innerHTML
 + '<br>3rd list tiem: ' + x[2].innerHTML
 + '<br>4th list item: ' + x[3].innerHTML;
document.getElementById("demo").style.color = "red";
x[1].style.color = "red";
x[3].style.color = "green";
x[1].style.fontSize="16pt";
x[1].style.borderStyle = "dashed";
x[1].style.borderWidth = "2pt";
```



The HTML DOM - Events

- Can execute code when an *event* occurs. Events include:
  - When a user clicks the mouse (onclick=)
  - When a web page has loaded (onload= , onunload=)
  - When an image has been loaded (onload=)
  - When the mouse moves over an element (onmouseover= , onmouseout=)
  - When an input field is changed (onchange=)
  - When an HTML form is submitted (onsubmit=)
  - When a user strokes a key (onkeypress=)

The HTML DOM – Events - Example

```
<!DOCTYPE html>
<html>
<body>
<h1 id="id1">My Heading 1</h1>
<button type="button" onclick="document.getElementById('id1').style.color = 'red'">Click Me!</button>
</body>
</html>
```

Before clicking the ‘Click Me!’ button

My Heading 1

Click Me!

After clicking the ‘Click Me!’ button

My Heading 1

Click Me!

The HTML DOM – Building HTML

- Javascript can modify the DOM of a document and add or remove elements
- Use 'createXXX() methods:
  - **createElement()** – Creates and returns a new element node
  - **createTextNode()** – Creates a node that holds text
- Use various methods to add or remove nodes:
  - **addChild()** – Adds a child to a node
  - **insertBefore()** – Inserts a new node before a specific child
  - **removeChild()** – Removes a node from the DOM
  - **replaceChild()** – Replaces one child node with another

The HTML DOM – Building HTML Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Document</title>
</head>
<body>
<div id="theDiv">
<h1>Hello, World! !</h1>
<p id="one">This is text from paragraph 1</p>
<p id="two">This is text from paragraph 2</p>
<p id="four">This is text from paragraph 4</p>
<button onclick="remv(document.getElementById('two'))">Click to remove p2</button>
</div>
<script src="index5.js"></script>
</body>
</html>
```

index.js:

```
var newPara = document.createElement("p");
var content = document.createTextNode("This is a new paragraph.");
newPara.appendChild(content);
var divElem = document.getElementById("theDiv");
divElem.appendChild(newPara);
function remv(element) {
  element.parentNode.removeChild(element);
}
```

The HTML DOM – Building HTML Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Document</title>
</head>
<body>
<div id="theDiv">
<h1>Hello, World! !</h1>
<p id="one">This is text from paragraph 1</p>
<p id="two">This is text from paragraph 2</p>
<p id="four">This is text from paragraph 4</p>
<button onclick="repl(document.getElementById('two'))">Click to replace p2</button>
</div>
<script>
function repl(element) {
  var para = document.createElement("p");
  var node = document.createTextNode("This is a new paragraph!");
  para.appendChild(node);
  element.parentNode.replaceChild(para, element);
}
</script>
</body>
</html>
```

The HTML DOM

- More HTML DOM actions:
  https://www.w3schools.com/js/js_htmldom.asp

jQuery

- A Javascript library
- Makes numerous tasks easier
- Browser independent
- Capabilities include:
  - **HTML/DOM manipulation**
  - **CSS manipulation**
  - **HTML event methods**
  - **Effects and animations**
  - AJAX
  - Utilities

jQuery

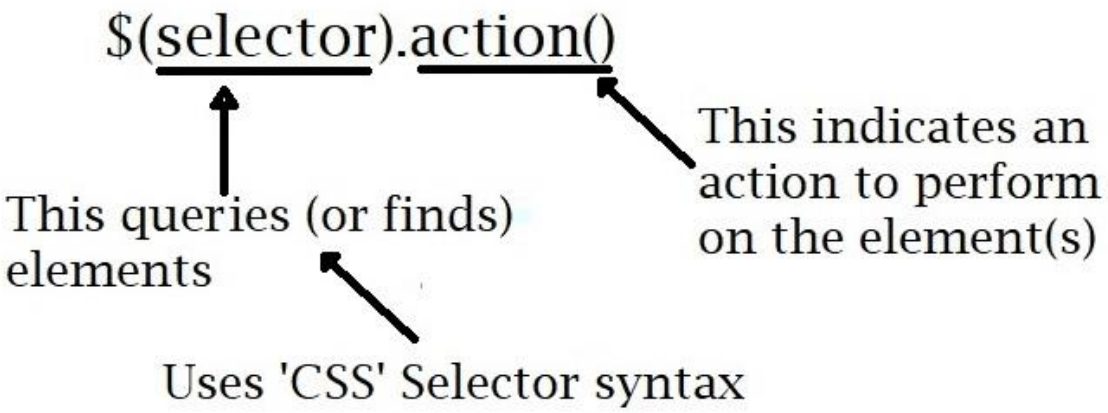
- Adding jQuery to web site
  - Download from jQuery.com
    - Production version ← Use this for live websites
    - Development version ← Use for testing
    - Place library in same folder as web pages
    - To include, use:

```
<head>
<script src="jQuery-3.5.1.min.js"></script>
</head>
```

  - Include jQuery from a CDN (Content Delivery Network), like Google or MS
  - https://code.jQuery.com/jQuery-3.5.1.min.js

jQuery

- Syntax:

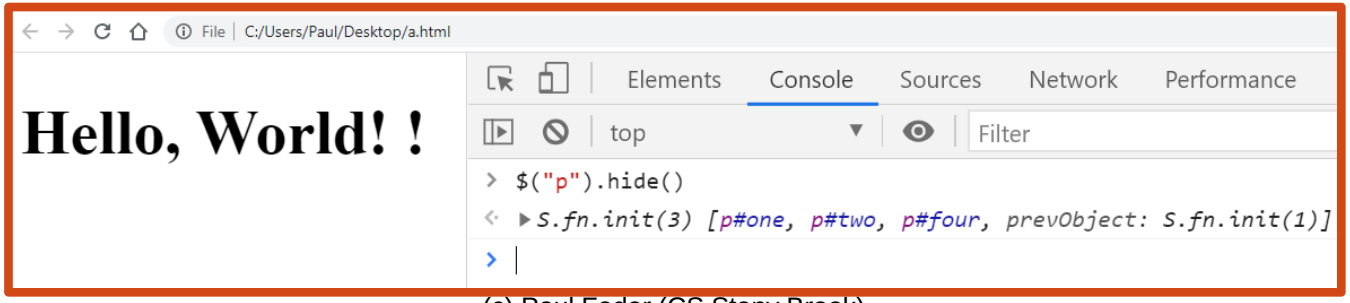


- Examples:
  - $(this).hide() – Hides current element
  - $("p").hide() – Hides all paragraphs (‘p’ elements)
  - $(".test").hide() – Hides all elements with **class** of **test**
  - $("#test").hide() – Hides elements with an **id** of **test**

jQuery

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Document</title>
<script src="https://code.jQuery.com/jQuery-3.5.1.min.js">
</script>
</head>
<body>
<div id="theDiv">
<h1>Hello, World! !</h1>
<p id="one">This is text from paragraph 1</p>
<p id="two">This is text from paragraph 2</p>
<p id="four">This is text from paragraph 4</p>
</div>
</body>
</html>
```

> $("p").hide()



jQuery – More on Selectors

- Selector forms
  - ("p.intro") – Paragraphs with class of intro
  - ("p:first") – First paragraph in document
  - ("ul li:first") – First <li> element in first
  - ("ul li:first-child") - First <li> of every
  - ("[href]") – Any element with an href attribute
  - ("a[target='_blank']") – All **anchors** with a **target** attribute equal to **_blank**
  - ("tr:even") – Selects all even <tr> elements
  - (tr:odd") – Selects all odd <tr> elements

jQuery – DOM Events

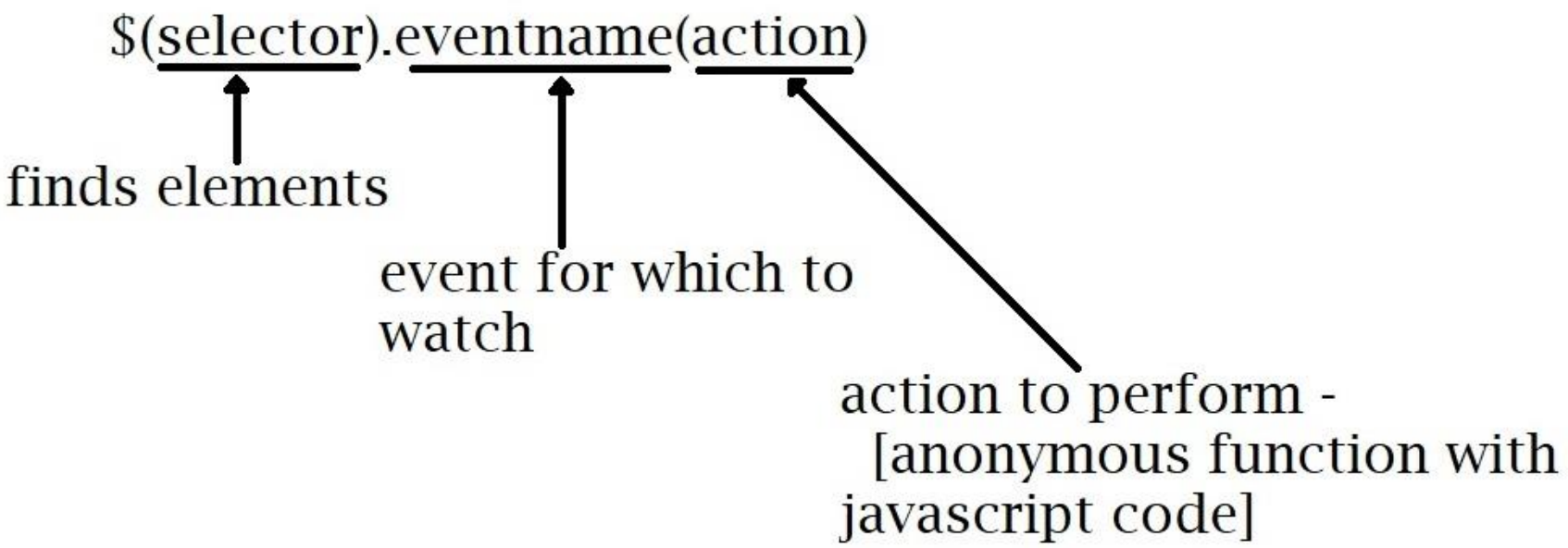
- jQuery supports catching most DOM events
  - Mouse Events
    - click
    - dblclick
    - mouseenter
    - mouseleave
  - Keyboard Events
    - keypress
    - keydown
    - keyup

jQuery – DOM Events

- jQuery supports catching most DOM events
  - Form Events
    - submit
    - change
    - focus
    - blur
  - Document Events
    - load
    - resize
    - scroll
    - unload

jQuery – DOM Events – Use in jQuery

- Syntax:



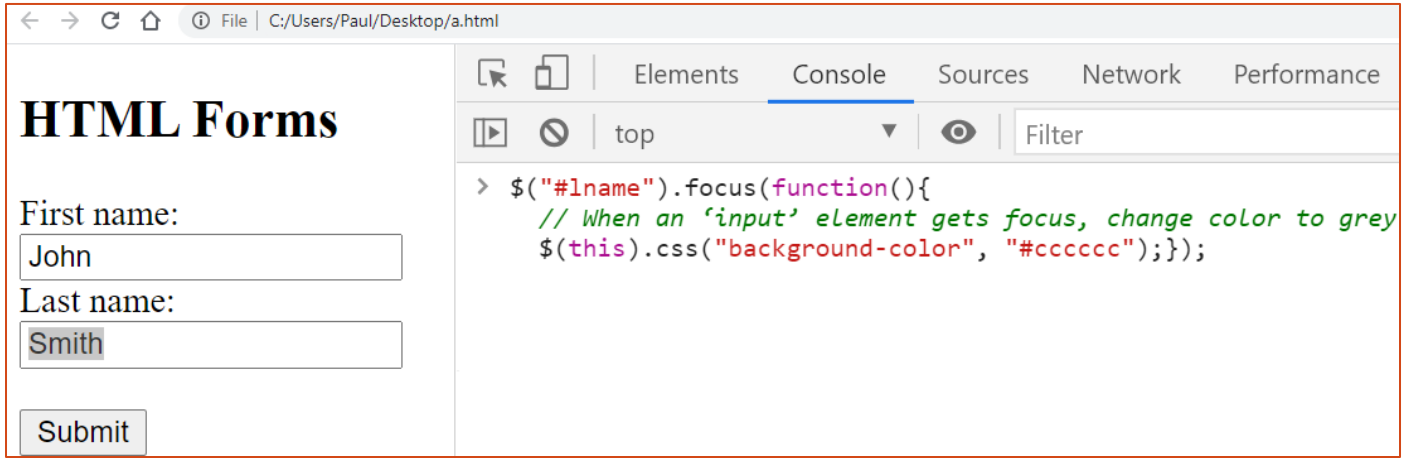
jQuery – Event Examples

```
$("#p1").mouseenter(function(){
  // When mouse enters element with id=p1, put up alert dialog
  alert("You entered p1!");
});
```

```
<div id="p1">
<h1>Hello, World! !</h1>
</div>
```

jQuery – Event Examples

```
$("input").focus(function(){
  // When an 'input' element gets focus, change color to grey
  $(this).css("background-color", "#cccccc");
});
<!DOCTYPE html>
<html>
<body>
<form action="/action_page.php">
  <input type="text" id="lname" name="lname" value="Smith"><br><br>
</form>
</body>
</html>
```



jQuery – Event Examples

Handing multiple events:

```
$("p").on({
  mouseenter: function(){
    $(this).css("background-color", "lightgray");
  },
  mouseleave: function(){
    $(this).css("background-color", "lightblue");
  },
  click: function(){
    $(this).css("background-color", "yellow");
  }
});
```

This example shows how to change CSS attribute values in an event handler!

When mouse enters a paragraph, change background color to light gray

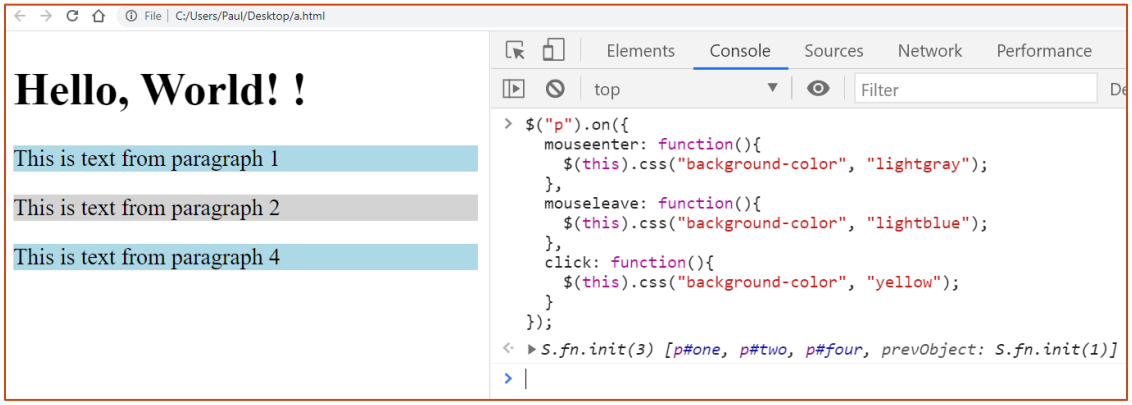
When mouse leaves a paragraph, change background color to light blue

When mouse button is clicked on a paragraph, change background color to yellow

jQuery – Event Examples

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Document</title>
<script src="https://code.jQuery.com/jQuery-
3.5.1.min.js"></script>
</head>
<body>
<div id="p1">
<h1>Hello, World! !</h1>
<p id="one">This is text from paragraph 1</p>
<p id="two">This is text from paragraph 2</p>
<p id="four">This is text from paragraph 4</p>
</div>
</body>
</html>
```

```
$("p").on({
  mouseenter: function(){
    $(this).css("background-color",
"lightgray");
  },
  mouseleave: function(){
    $(this).css("background-color",
"lightblue");
  },
  click: function(){
    $(this).css("background-color", "yellow");
  }
});
```



jQuery - Effects

- jQuery supports some 'effects' on elements
  - **hide**() – hide an element (make it disappear)
  - **show**() – show an element
  - **toggle**() – switch between hiding and showing an element
  - **slide**() – Makes elements slide up or down
  - **fade**() – Fades object between invisible and visible

jQuery

- Hide/Show have options to indicate speed and a callback function. [Callback functions allow actions after the effect is entirely completed]
  - **hide(<speed>,<callback>)** – Speed can be **slow, fast,** or a number of milliseconds
  - **show(<speed>, <callback>)**
- Fade methods – Each allow a 'speed' parameters
  - **fadeIn()**
  - **fadeOut()**
  - **fadeToggle()**
  - **fadeTo()**
- Slide methods – Each allow 'speed' and 'callback' parameters
  - **slideUp**()
  - **slideDown**()
  - **slideToggle**()

jQuery – Effects Examples

```
$("button").click(function(){
  $("p").hide(1000);
});
```

Hide paragraphs when the button is clicked

```
$("button").click(function(){
  $("#div1").fadeIn();
  $("#div2").fadeIn("slow");
  $("#div3").fadeIn(3000);
});
```

When button is clicked, fade in three ‘divs’ at different rates.

```
$("#flip").click(function(){
  $("#panel").slideToggle();
});
```

When button is clicked slide a panel down/up

jQuery – Effects Examples

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Document</title>
<script src="https://code.jQuery.com/jQuery-3.5.1.min.js"></script>
</head>
<body>
<div id="theDiv">
<h1>Hello, World! !</h1>
<p id="one">This is text from paragraph 1</p>
<p id="two">This is text from paragraph 2</p>
<p id="four">This is text from paragraph 4</p>
<button onclick="repl(document.getElementById('two'))">Click to replace p2</button>
</div>
<script>
function repl(element) {
var para = document.createElement("p");
var node = document.createTextNode("This is a new paragraph!");
para.appendChild(node);
element.parentNode.replaceChild(para, element);
}
</script>
</body>
</html>
```

```
$("button").click(function(){
  $("p").hide(1000);
});
```

Hello, World! !

Click to replace p2

jQuery – HTML Element and Attribute Manipulation

- jQuery has facilities to query or update html content and attribute values
- Main methods:
  - **text()** - Sets or returns text content of elements
  - **html()** – Sets or returns content of selected elements (including HTML markup)
  - **val()** – Sets or returns value of form fields
  - attr() – Get attribute value

jQuery

```
$("#btn1").click(function(){
  alert("Text: " + $("#one").text());
});
```

First button displays text from element with **id test**

```
$("#btn2").click(function(){
  alert("HTML: " + $("#one").html());
});
```

Second button displays text with html markup from element with **id test.**

```
$("#btn3").click(function(){
  alert($("#three").attr("href"));
});
```

When button btn3 is clicked, an alert dialog displays the **href** attribute of the element with **id** three**.**

jQuery – Setting content

```
$("#btn1").click(function(){
  $("#test1").text("Hello world!");
});
```

// Button 1 sets value of content in element with **id test1** to the string ***'Hello, world!'***

```
$("#btn2").click(function(){
  $("#test2").html("<b>Hello world!</b>");
});
```

// Button 2 sets the HTML content of element with **id test2** to ***'Hello, world!'*** with bold tags around the string.

```
$("#btn3").click(function(){
  $("#test3").attr("href", "https://www.w3schools.com/jQuery/");
});
```

// Button click changes the **href** attribute of element with **id test3** to new url

jQuery - Modifying HTML content

- jQuery can update content
  - **append()** – Append content to an element
  - **prepend()** – Prepend content to an element
  - **after()** – Insert content after a named element
  - **before()** – insert content before a named element
  - **remove()** – Removes selected element and children
  - **empty()** – removes child elements of the selected element

jQuery – Modifying HTML content - Example

```
$("h1").append("Extended header text")

$("p").prepend("Added text at beginning of a paragraph")

$("img").before("<p><b>Add this before all images</b></p>")

$("img").after("<p><i>Add this after all images</i></p>")

$("#ul1").remove()

$("#ul1").empty()
```

jQuery – Manipulating style

- jQuery can fetch and update css style information
- Apply .css() method to element set
  - form 1

$(selector).css("propertyname")

  – *Returns the value of propertyname on the element*

  - form 2a

$(selector).css("propertyname", "newvalue")

  – Sets the value of propertyname to newvalue.

  - form 2b - Can set multiple properties with:

$(selector).css({"propname1":"value1","propname2":"value2",...})

jQuery – Methods to traverse DOM tree

- jQuery provides methods to traverse a document
  - **.parent()** – Returns parent of the element
  - **.parents()** – Returns chain of parents of element back to <html>
  - **.children()** – Returns children of the element
  - **.find()** – Searches for an element within the descendants of the selected element
  - **.siblings()** – Returns all siblings of a certain element type
  - **.next()** – Next sibling
  - **.nextAll()** – List of following siblings
  - **.nextUntil()** – List of following ciblings stopping before a name sibling
  - **.prev()** – Previous sibling
  - **.prevAll()** – List of previous siblings
  - **.prevUntil() –** List of previous siblings until a named sibling is reached