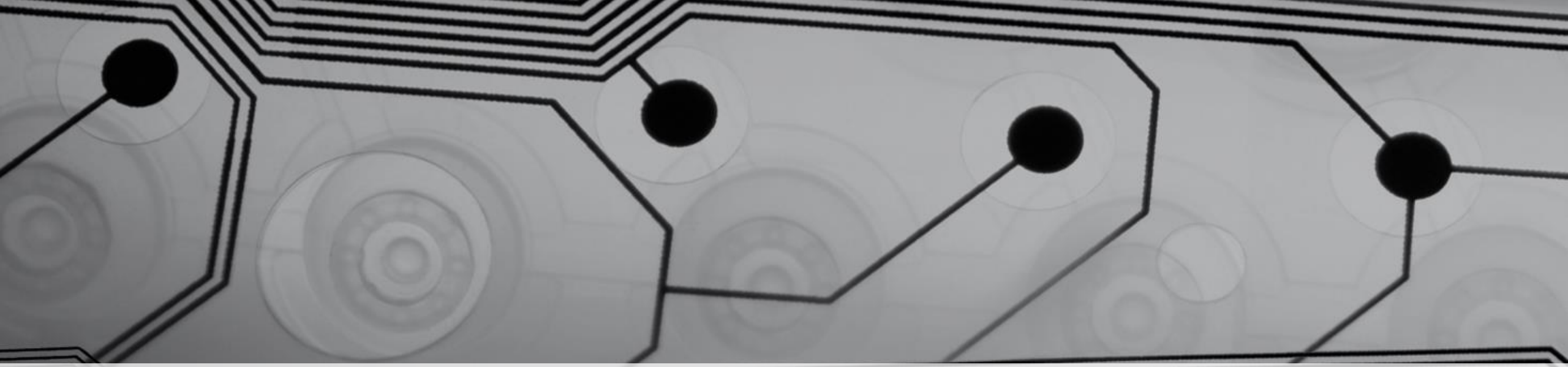


jQuery

2020/21 COMP3322 Modern Technologies on WWW

Contents

- Intro to jQuery
- Selectors
- Events
- Manipulating the DOM
- DOM Traversal
- Special effects
- Using AJAX
- Others

What is jQuery?

JQuery was introduced in 2006, but JS querySelectorAll() was introduced in 2013.

- jQuery is a **JavaScript file**.
- jQuery offers a quick and simple way to achieve a variety of common JavaScript tricks
  - Allows us to **find elements** using CSS-style selectors and jQuery custom selectors.
  - **Do something** with the selected element(s) using **jQuery methods.**
- Main advantages:
  - **Support across all major browsers** and without any fallback code needed.
  - Offer developers simpler ways to perform common tasks, i.e. **less code**.
- Someone calls it a framework, but I consider it as a library which offers some **syntactical shortcuts** and methods to simplify the tasks.

Including jQuery

- You must either:
  - Download jQuery to your Website and link to it
    - You can download a copy from jQuery.com. https://jquery.com/download/
    - There is usually a choice between a Production (**compressed**) version and a Development (**uncompressed**) version.
      - For learning, use the Development version as it is uncompressed and readable code.
      - In your HTML document, reference it with the <script> tag in the <head> section.

```
<head>
  <script src="jquery-3.3.1.js"></script>
</head>
```

  - Reference to a third-party CDN, such as Google and Microsoft

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
</head>
```

Including jQuery

- Using a third-party **content delivery network (CDN)** is advantageous for several reasons.
  - The **bandwidth** of the file is offloaded to reduce the demand on your servers.
  - The user may already have **cached** the third-party file and thus not have to download it again, thereby reducing the total loading time.
- A disadvantage to the third-party CDN is that your jQuery will fail if the third-party host fails (unlikely but possible)

jQuery Syntax

- A jQuery statement consists of:
  - A **$** symbol, **followed** by a **Selector** in parentheses, and then **a period** and **a method** to apply to the **selected element(s)**

```
$("#divTest1").text("Hello, world!");
```

    - which is the same as this **vanilla JavaScript** code:

```
document.getElementById("divTest1").innerHTML = "Hello, world!";
```

  - Actually, **$** is just an alias for jQuery; another way to issue the jQuery request is:

```
jQuery("#divTest1").text("Hello, world!");
```

Waiting until DOM is ready

- jQuery is so closely related to DOM, it is a good practice to wait for the document to **be fully loaded and ready**, before manipulating the document.
- Instead of placing the JavaScript script at the end of the HTML document, you can use this jQuery feature:

```
$(document).ready(function() {
    // your code goes here

});
```

A shorter version

```
$(function() {
    // your code goes here

});
```

  - This makes sure the page DOM is ready for the code to execute. In that case, you can place the JavaScript (and jQuery) code in the <head> section

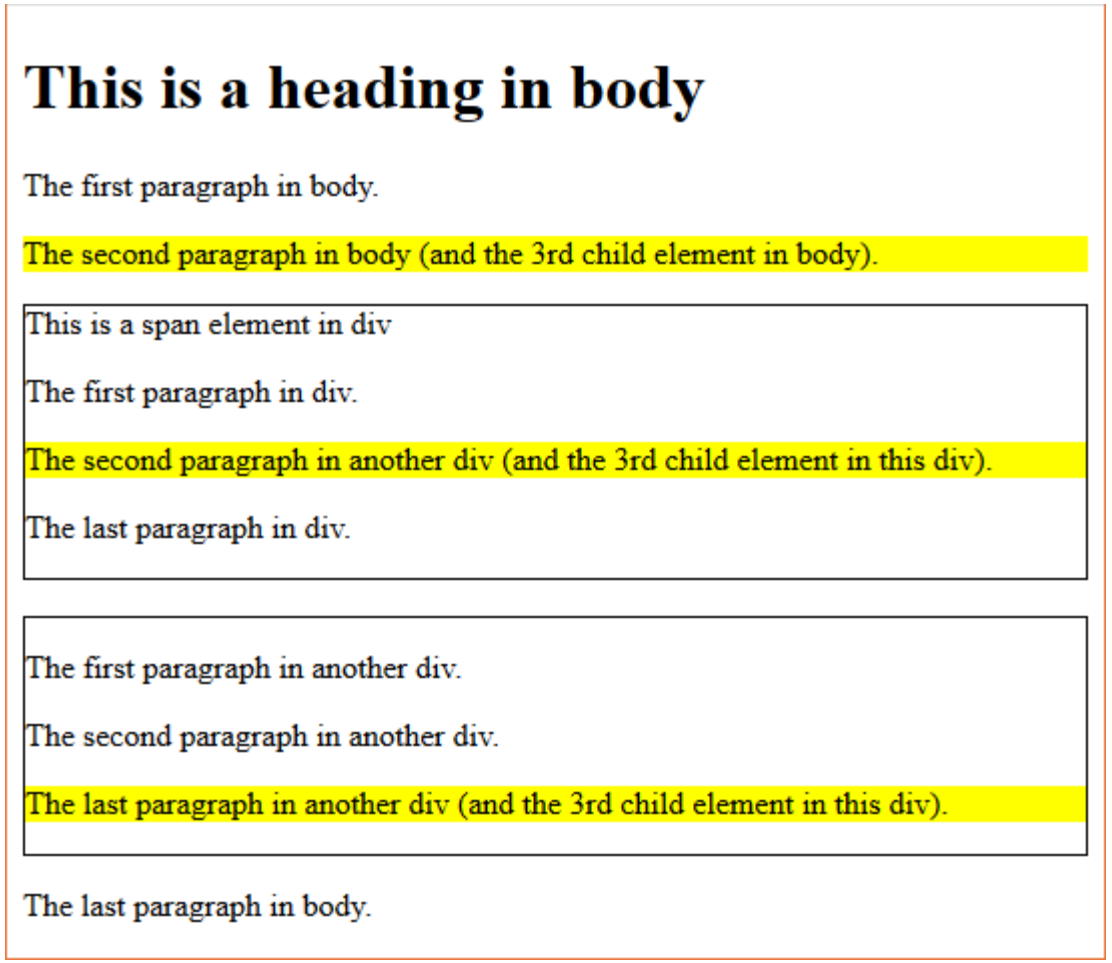
Selectors

- jQuery can help us find elements based on their ID, classes, types, attributes, values of attributes, . . . which works in exactly **the same way** as CSS selectors.
- Selectors
  - The element selector `$("p")`
  - The #id selector `$("#test")`
  - The .class selector `$(".test")`
  - The attribute selector `$("a[target='_blank']")`
  - Compound and contextual selector `$("p.test")` `$("#test > b")`
  - jQuery custom selector `$("ul li:lt(3)")` `$("div:contains('text')")`

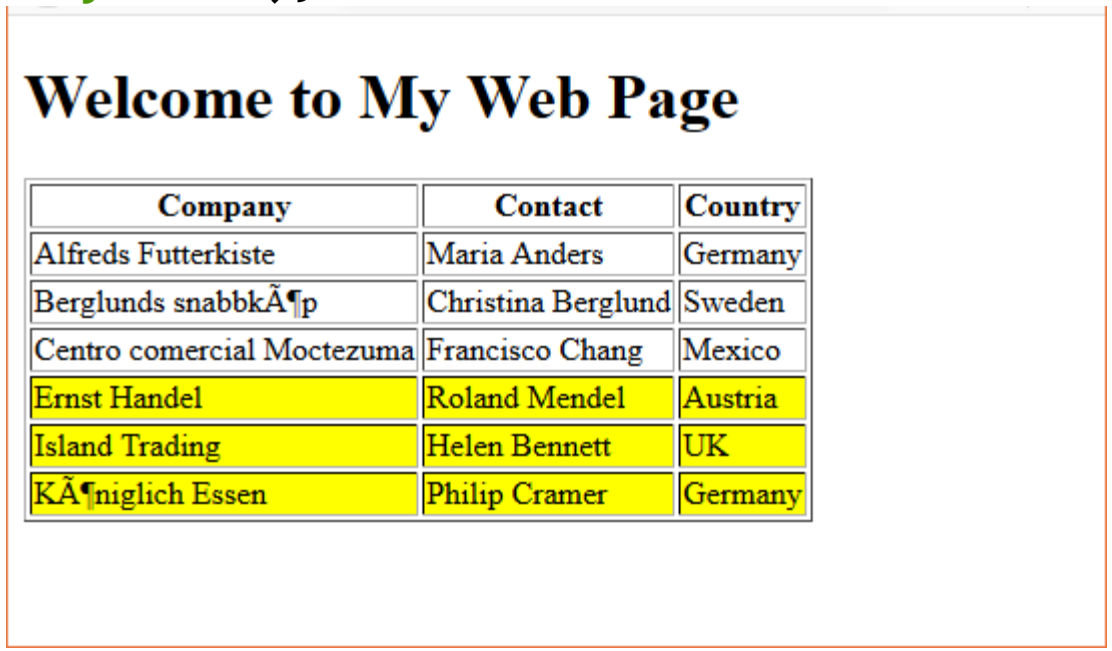
For the complete list, please visit: https://www.w3schools.com/jquery/jquery_ref_selectors.asp

Examples

```
$("p:nth-child(3)").css("background-color",
  "yellow");
```




```
$("tr:gt(3)").css("background-color",
  "yellow");
```



jQuery Object

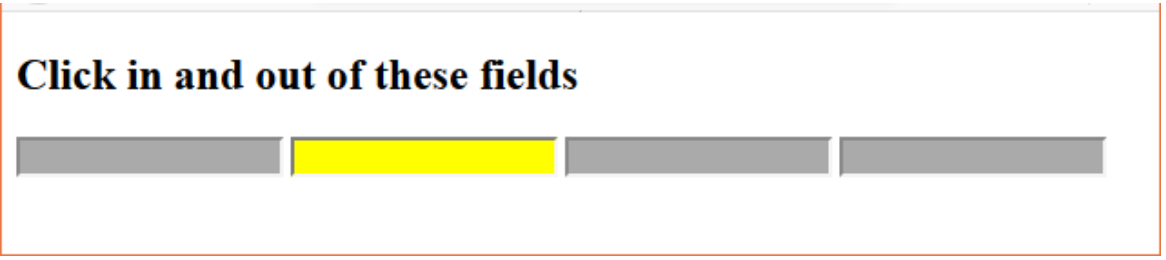
- When you **select** an element or set of elements, jQuery returns an **jQuery object**, which contains references to the element(s).
  - Sometimes, jQuery object is also being called as the **matched set** or **jQuery selection**.
- Like any object, the jQuery object has **properties and methods**. They allow you to work with those element(s).
- Example:
  - jQuery object has a property called length, which gives us the number of elements in the object.

Events – jQuery Listeners

- jQuery **enhances** the basic event-handling mechanisms by offering the **events methods** for most native browser events.
- Here are some common events:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

For the complete list, please visit: https://www.w3schools.com/jquery/jquery_ref_events.asp



http://i7.cs.hku.hk/~atctam/c3322/JQ/blur-1.html

jQuery version

```
<!DOCTYPE html>
<html>
  <head>
    <title>Events: blur</title>
    <script src='jquery-3.3.1.js'></script>
  </head>
  <body>
    <h2>Click in and out of these fields</h2>
    <input id='first'> <input> <input> <input>
    <script>
      $('#first').focus();
      $('input').focus(function() {
        $(this).css('background', '#ff0') } );
      $('input').blur(function() {
        $(this).css('background', '#aaa') } );
    </script>
  </body>
</html>
```

Vanilla JavaScript

```
<!DOCTYPE html>
<html>
  <head>
    <title>Events: blur</title>
  </head>
  <body>
    <h2>Click in and out of these fields</h2>
    <input id='first'> <input> <input> <input>
    <script>
      var Blk = document.querySelectorAll('input');
      for (var i = 0; i < Blk.length; i++) {
        let obj = Blk[i];
        obj.addEventListener('focus', function () {
          this.style.backgroundColor = '#ff0';
        });
        obj.addEventListener('blur', function () {
          this.style.backgroundColor = '#aaa';
        });
      };
      Blk[0].focus();
    </script>
  </body>
</html>
```

.on() method

- Previous listed methods are just shorthand for the .on() method.
- The on method can:
  - *register multiple events and handlers to the jQuery object*
  - *bind the same handler function to multiple events*
  - provide data to the event handler

```
$('input').focus(function() {
  $(this).css('background', '#ff0') } );
$('input').blur(function() {
  $(this).css('background', '#aaa') } );
```

```
$('input').on(
  {'focus': function() {
     $(this).css('background', '#ff0')},
   'blur': function() {
     $(this).css('background', '#aaa')} });
```

```
$( "button" ).on( "click",
  {name: "Karl"}, greet( event ) {
  alert( "Hello " + event.data.name );
} );
```

```
$( "#cart" ).on( "blur mouseleave", function(event) {
  $(this).hide();
});
```

.off method

- The .off() method removes event handlers that were **attached with .on()**.
- To use the .off() method to remove an event listener, pass in the event type to off.

```
$('input').off('focus');

$('input').off();
```

Getting and Setting Information

- **Some jQuery methods** both **retrieve** information from, and **update** the contents of, selected element(s). But they do **not always** apply to all elements.
- GET information
  - if jQuery object holds more than one element, **some 'get' methods** only retrieve information from the **first element** only.
- SET information
  - The 'set' methods **update all the elements** in the jQuery selection, not just the first one.

Getting and Setting Content

- The .html() and .text() methods both retrieve and update the **content** of elements.
  - .html()
    - **gets** the HTLM content (including markup) inside the **first element** together with its descendants.
    - **replaces** the content of **all selected elements** with the new content.
  - .text()
    - **gets** the text content only, from **every element** in the jQuery selection, together with its descendants.
    - **replaces** the content of **all selected elements** with the new content **with any markup being shown as text**.
- The .val() method gets the value of **the form** first element and sets the value of all matching form elements.

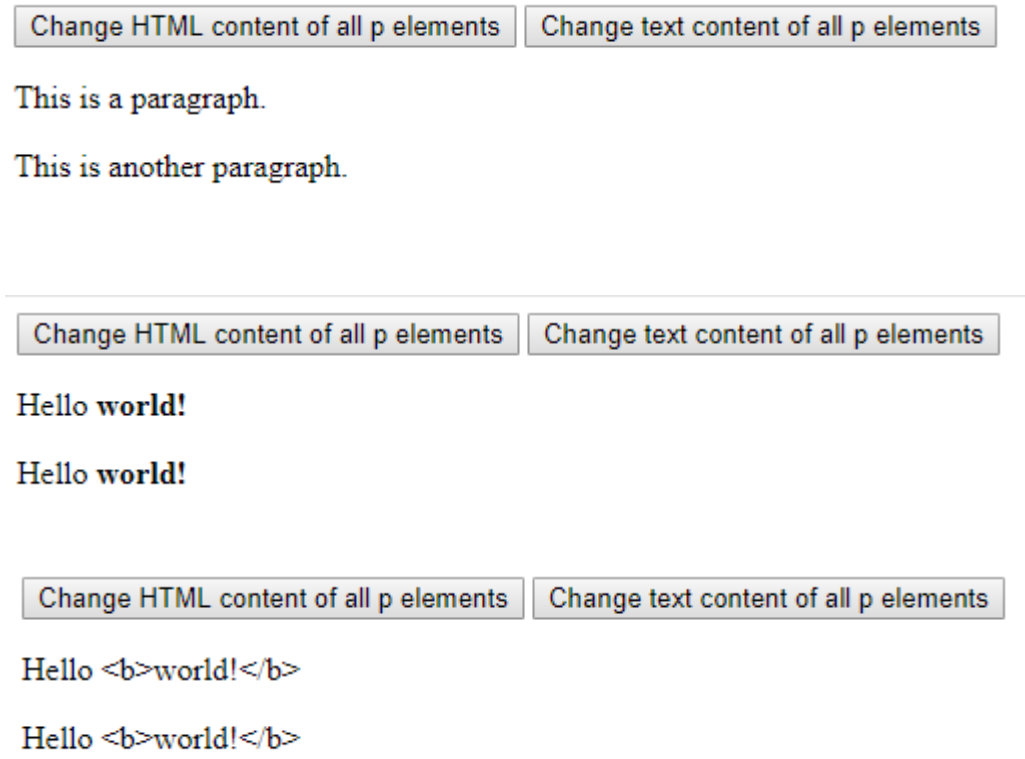
Example

```
<script>
$(document).ready(function(){
  $("#bttn1").click(function(){
    $("p").html("Hello <b>world!</b>");
  });
  $("#bttn2").click(function(){
    $("p").text("Hello <b>world!</b>");
  });
});
</script>

    :
    :
```

```
<button id="bttn1">Change HTML content of all p
elements</button>
<button id="bttn2">Change text content of all p
elements</button>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```



CSS Manipulation

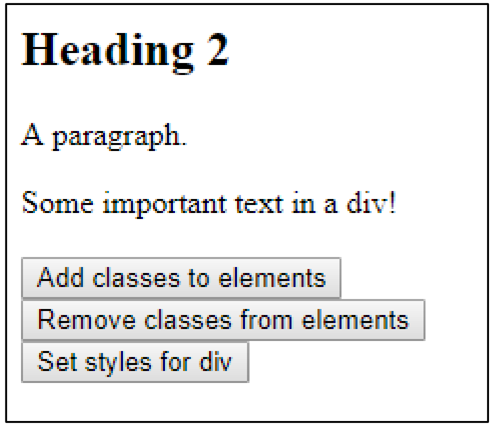
- .attr() – this method can **get or set** a specific attribute and its value.
- .removeAttr() – this method **removes** a specified attribute (and its value)
- .addClass() – this method **adds** a new value to the class attribute.
- .removeClass() – this method **removes** a value from the class attribute.
- .css() – this method can **get or set** one or more style properties.

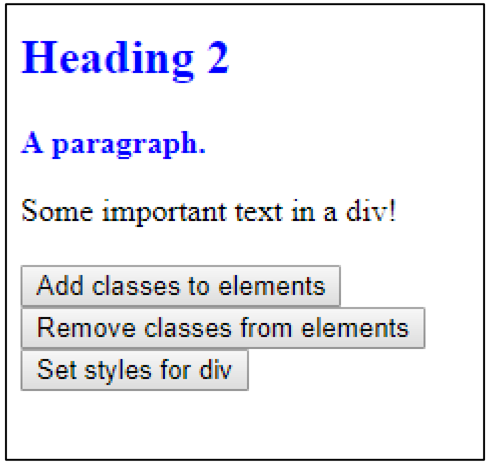
```
$("p").css({
    "color": "white",
    "background-color": "#98bf21",
    "font-family": "Arial",
    "font-size": "20px",
    "padding": "5px"
});
```

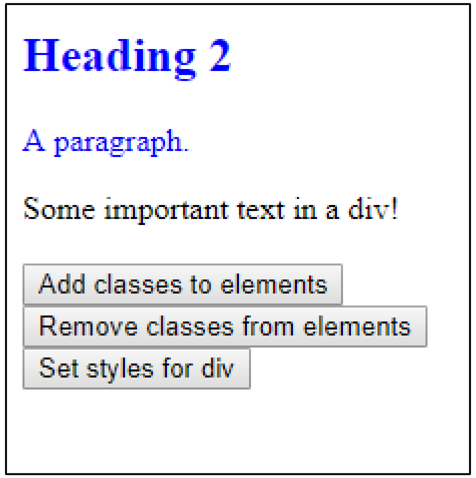
Example

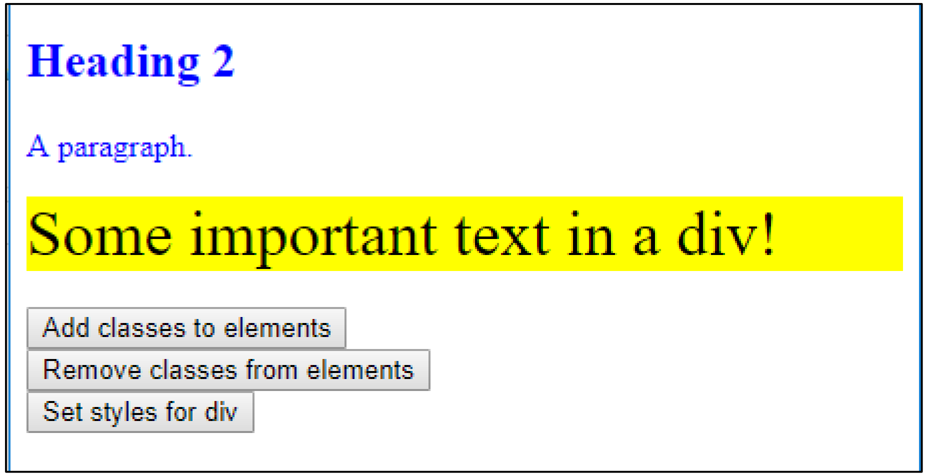
```
<script>
  $("#btn1").click(function(){
    $("h2, p").addClass("important blue");
  });
  $("#btn2").click(function(){
    $("p").removeClass("important");
  });
  $("#btn3").click(function(){
    $("div").css({"background-color": "yellow",
      "font-size": "200%“,
      "width": "300px"});
  });
</script>
<style>
.important {
    font-weight: bold;
}
.blue {
    color: blue;
}
</style>

<body>
  <h2>Heading 2</h2>
  <p>A paragraph.</p>
  <div>Some important text in a div!</div><br>
  <button id="btn1">Add classes to elements</button><br>
  <button id="btn2">Remove classes from elements</button><br>
  <button id="btn3" >Set styles for div</button>
</body>
```









Chaining

- It is possible to **call multiple methods** on the same jQuery selection.

```
$("#message").text("Hello, world!").css("color", "blue");
```

First Method **Second Method**

- ***Only methods related to update the jQuery object can be chained.*** Methods related to retrieve information cannot be chained.
- If one method in the chain does not work, **the rest will not run**.

DOM Traversal

- These methods select element nodes relative to current selection.

Method	Description
.parent()	Direct parent of current selection
.parents()	All ancestors of current selection
.parentsUntil(selector)	All ancestors of current selection up to but not include the argument selector
.children()	All children of current selection
.siblings()	All siblings of current selection
.next()	Next sibling of current selection
.nextAll()	All siblings that follow current selection
.prev()	Previous sibling of current selection
.prevAll()	All previous siblings of current selection
.find(selector)	All descendent elements of current selection that matches argument
.closest(selector)	Nearest ancestor of current selection that matches argument

Adding and Removing Elements

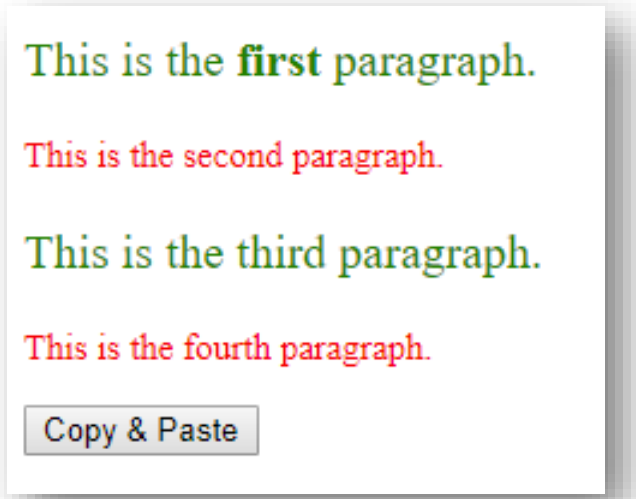
Method	Description
.append()	Inserts content inside the selected element(s) before the closing tag
.prepend()	Inserts content inside the selected element(s) after the opening tag
.after()	Inserts content after the selected element(s)
.before()	Inserts content before the selected element(s)
.remove()	Removes current selected element(s) and descendants from DOM tree
.empty()	Removes children and descendants of current selected element(s)
.detach()	Same as .remove() but keeps a copy of them in memory
.clone()	Creates a copy of current selection (including descendants)

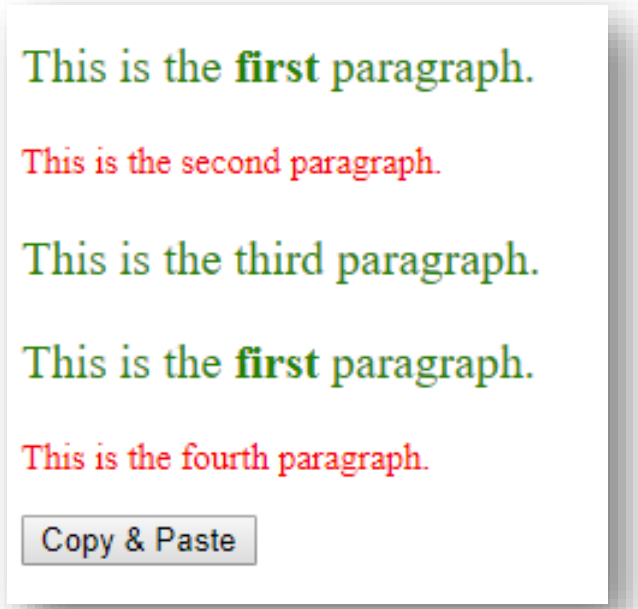
Example

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $copy = $("p:first").clone();
    $("p:first").next().next().after($copy);
  });
});
</script>
<style>
.odd {
  color: green;
  font-size: 20px;
}
.even {
  color: red;
  font-size: 15px;
}
</style>

<body>
  <p class="odd">This is the <b>first</b> paragraph.</p>
  <p class="even">This is the second paragraph.</p>
  <p class="odd">This is the third paragraph.</p>
  <p class="even">This is the fourth paragraph.</p>

  <button>Copy & Paste</button>
</body>
```





Special Effects

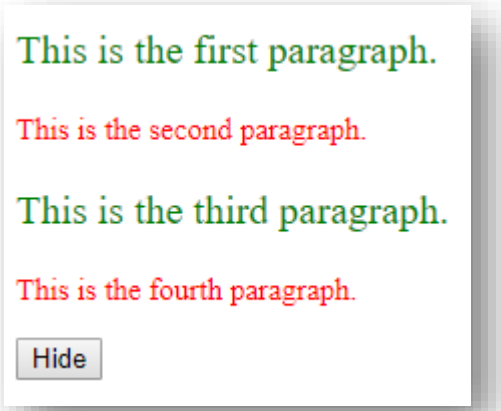
- jQuery is quite special when processing special effects.
- Using special effect really is as simple as selecting one or more elements and then applying one or more effects to them.
- The core effects available are:
  - Hiding and showing
  - Fading in and out
  - Sliding
  - Animations

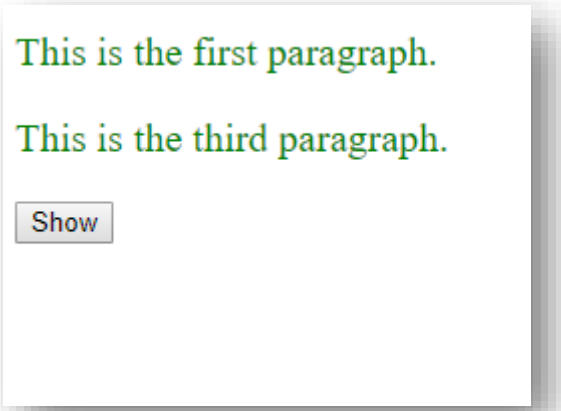
Special Effects

- .show() – displays selected element(s)
- .hide() – hides selected element(s)
- .toggle() – toggles between show and hide
- All three accept two optional arguments:
  - Speed – speed of the animation: “slow”, “fast”, or in milliseconds
  - Callback – register a callback function to be executed **after** the effect

```
<script>
$(document).ready(function(){
  $("#btn2").hide();
  $("button").click(function(){
    $("p.even").toggle("slow");
    $("button").toggle();
  });
});
</script>
```

```
<body>
  <p class="odd">This is the first paragraph.</p>
  <p class="even">This is the second paragraph.</p>
  <p class="odd">This is the third paragraph.</p>
  <p class="even">This is the fourth paragraph.</p>
  <button id="btn1">Hide</button>
  <button id="btn2">Show</button>
</body>
```





Special Effects

Method	Description
.fadeIn()	Fades in hidden element(s) by making them opaque https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_fadein
.fadeout()	The reverse of .fadeIn()
.fadeToggle()	The toggle function of fade
.fadeTo(speed, opacity)	Fades to the given opacity level
.slideUp()	Slides the element(s) up to show/hide
.slideDown()	Slides the element(s) down to show/hide
.slideToggle()	The toggle function of slide https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_slide_toggle

Special Effects

- .animate() – allows us to create custom animations by **changing CSS properties**.

```
.animate({params}, speed, easing, callback)
```

  - params – required parameter that **defines the CSS property/value pairs** to be animated.
    - We can animate **any** CSS property whose value can be **represented as number**, e.g., height, width, and font-size.
    - It is also possible to **define relative values** (the value is then relative to the element's current value). This is done by putting += or -= in front of the value.

```
width: '+=150px'
```

    - All property names must be **camel-cased**, i.e., font-size to fontSize
  - Easing – optional parameter which indicates **the speed** of the element **in different points** of the animation.
    - Two possible values:
      - swing – [default] moves slower at the start/end, but faster in the middle
      - linear – moves in constant speed

Example

```
<style>
  #ball {
    position  :relative;
  }
  #box {
     :
  }
</style>

<div id='box'>
  <img id='ball' src='ball.png'>
</div>
<script>
  bounce();

  function bounce()
  {
    $('#ball')
      .animate( { left:'288px', top :'416px' }, "slow", "linear")
      .animate( { left:'576px', top :'192px' }, "slow", "linear")
      .animate( { left:'288px', top :'0px'   }, "slow", "linear")
      .animate( { left:'0px',   top :'192px' }, "slow", "linear")
  }
</script>
```

By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!

Special Effects

- .delay() – inserts a delay (in milliseconds or “slow” or “fast”) between the execution of next special effect.
- .stop() – stops the currently running animation (but continues to the next one if there is one).

jQuery AJAX

- **$.ajax()** method https://api.jquery.com/jQuery.ajax/
  - This is a powerful way of creating AJAX requests. You have many parameters to set and control.
  - This method is mostly used for requests where the other methods cannot be used.
- **.load()** method `.load(url,data,function(response,status,xhr))`
  - Allows us to fetch HTML content from a URL and **place** the returned data **into the selected element.**
    - url – the URL to which the request is sent
    - data (optional) – an object or string that is sent to the server
    - function (optional) – **a callback** that is executed when the request completes and data **insertion has been performed**.
      - response - contains the **result data** from the request
      - status - contains the **status of the request**
      - xhr - contains the **XMLHttpRequest object**
    - The POST method is used if data is provided as an object; otherwise, GET is assumed.

jQuery AJAX

- **$.get()** **$.getJSON()**

```
$.get(url,data,function(response,status,xhr),dataType)
```

```
$.getJSON(url,data,function(response,status,xhr))
```

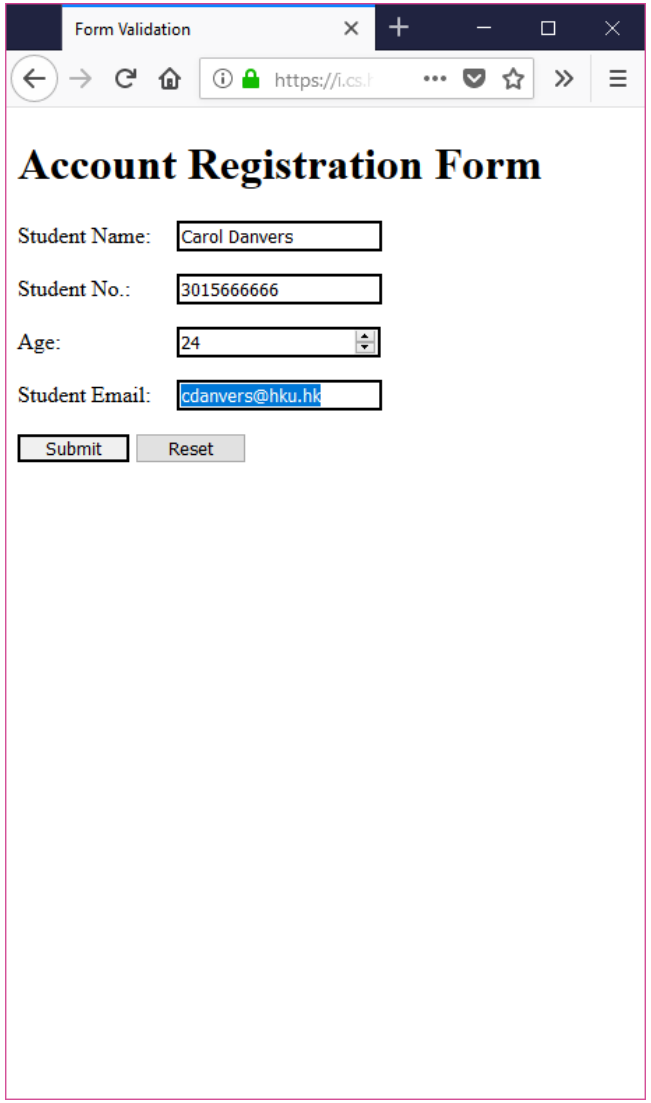
  - These two methods load data or JSON data from a server using the HTTP GET request.
  - The third parameter is a callback function to be executed if the request succeeds.
    - The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.
  - datatype – specifies the data type expected of the server response. Default: Intelligent Guess (xml, json, script, text, html).

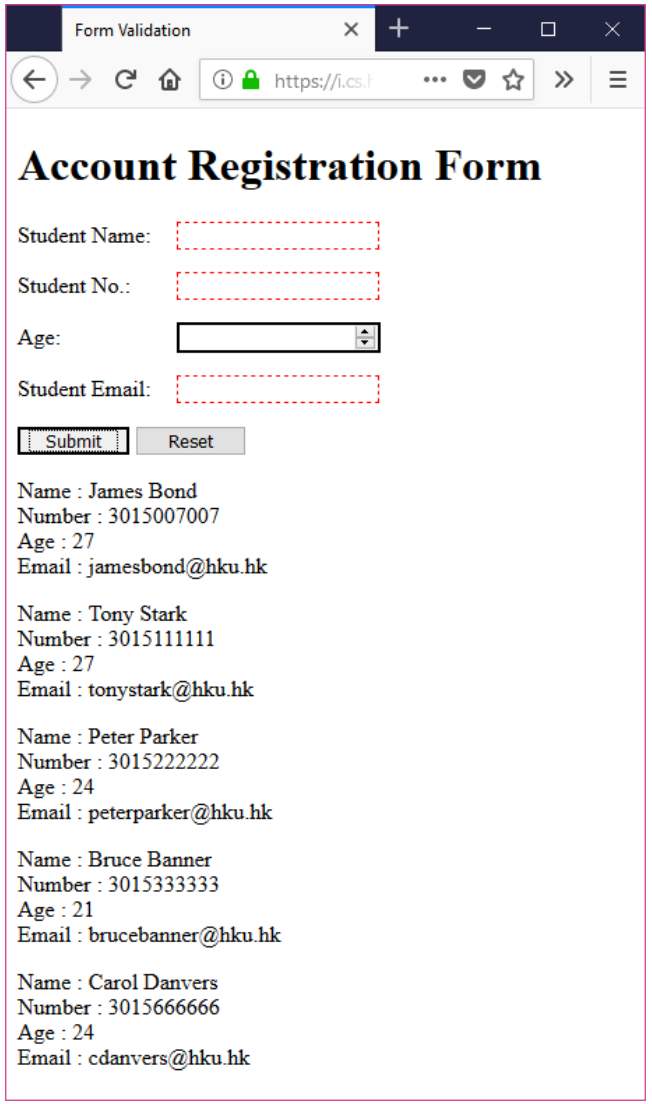
- **$.post()**

```
$.post(url,data,function(response,status,xhr),dataType)
```

  - This methods load data from a server using the HTTP POST request.

Demo – Our Registration Form Again
Add new record and display all





Vanilla JavaScript

```
var rform = document.getElementById("RegForm");
rform.onsubmit = addRequest;
function addRequest(evt) {
  evt.preventDefault();
  let formData = new FormData(rform);
  let init = {
    method: 'POST',
    body: formData
  };
  fetch('view7.php', init)
  .then( response => {
      if (response.status == 200) {
        response.json().then(jsObj => {
          let txt = '';
          for (let i in jsObj) {
            txt += "<p>Name   : "+jsObj[i].name+"<br> ";
            txt +=    "Number : "+jsObj[i].number+"<br> ";
            txt +=    "Age    : "+jsObj[i].age+"<br> ";
            txt +=    "Email  : "+jsObj[i].email+"</p> ";
          }
          document.getElementById('stdTable').innerHTML = txt;
          rform.reset();
        });
      }
  });
}
```

jQuery version

```
$("#RegForm").submit(function () {
    $rfrom = this;
    $.post("view7.php",
      $("#RegForm").serialize(),
      function (data, status) {
        if (status == "success") {
          let txt = '';
          for (let i in data) {
            txt += "<p>Name   : "+data[i].name+"<br> ";
            txt +=    "Number : "+data[i].number+"<br> ";
            txt +=    "Age    : "+data[i].age+"<br> ";
            txt +=    "Email  : "+data[i].email+"</p> ";
          }
          $("#stdTable").html(txt);
          $rfrom.reset();
        }
      }
    );
    return false;
});
```

To get more information, please visit: http://api.jquery.com/category/utilities/

Some jQuery Utility functions

- jQuery offers several utility methods in the $ namespace. These methods are helpful for accomplishing routine **programming** tasks.
- $.each() – iterates over arrays and objects
  - It accepts two arguments:
    - An array or an object to loop through
    - A function to be called **on each iteration**.
      - This function is called with two arguments: The current index of the item and the item itself.

```
$.each([ "foo", "bar", "baz" ], function( idx, val ) {
  console.log( "element " + idx + " is " + val );
});

$.each({ foo: "bar", baz: "bim" }, function( k, v ) {
  console.log( k + " : " + v );
});
```

Some jQuery Utility functions

- $.trim() – removes leading and trailing whitespace

```
$.trim( "    lots of extra whitespace    " );
```

- $.inArray() – returns a value's index in an array, or -1 if the value is not in the array.

```
var myArray = [ 1, 2, 3, 5 ];
if ( $.inArray( 4, myArray ) !== -1 ) {
  console.log( "found it!" );
}
```

- $.isArray(), $.isFunction(), $.isNumeric() are for checking if the argument is of a specific type.
- $.type() is for finding the type of the argument.

Avoiding Library Conflict

- By default, jQuery uses $ as a shortcut for jQuery.
- If you are using another JavaScript library that uses the $ variable, you can run into conflicts with jQuery.
- There are **various ways** to work around.
  - Put jQuery in no-conflict mode to avoid the conflict.
    - $.noConflict()
    - After that, you must call the jQuery function to use jQuery; e.g. jQuery("div").hide() rather than $("div").hide()
  - Create a new alias to jQuery
    - var jq = $.noConflict()
    - Now the new alias jq replaces the use of $.

Others

- Dimensions
  - jQuery offers a variety of methods for obtaining and modifying dimension and position information about an element.
    - .width() and .height() – set or get the width/height of an element (excludes padding, border and margin).
    - .innerWidth() and .innerHeight() – returns/sets the width/height of an element (including padding).
    - .outerWidth() and .outHeight() – returns/sets the width/height of an element (excluding margin)
    - .outerWidth(true) and .outHeight(true) – returns/sets the total width/height of an element.
- Extending the jQuery prototype to include a new method
  - You can make your own plugins and use them privately or release them publicly.
  - To add a new method:

```
$.fn.myNewPlugin = function() {

  return this.each(function() {
    // Do something to each element selected by jQuery.
  });

};
```

Reading

- jQuery Learning Center
  - https://learn.jquery.com/
  - Two sections:
    - About jQuery
    - Using jQuery Core

References

- jQuery Learning Center
  - https://learn.jquery.com/
- jQuery Tutorial - W3Schools.com
  - https://www.w3schools.com/jquery/
- jQuery Tutorial – Tutorialspoint.com
  - https://www.tutorialspoint.com/jquery/jquery-plugins.htm