**web app**: be used mostly by web browsers in a server-based program

**web service**: be used programmatically in a server-based program

- use HTTP protocol (wide use and scalability)

- application (NOT humans) useto the web service, and handles the response

- known as an application programming interface (API)

- on any web-connected technology platform, in any language, and on any kind of device

**resource**: a digital asset

**representation**: digital asset that is formatted as a specific internet media type

**internet media type**: defined as data format for representation of resource on the internet

**state**: current value of a resource

**REST**: Roy Fielding, REpresentational State Transfer, as a way to use the HTTP protocol with a standard message format to preform operations on data

**web API**: captures the essence of “REST API”

**$(function()** { // this is the same as: **$(document).ready(function()**{

**// jQuery Selectors**

let selector1 = $( "\*" );

let selector2 = $( "#animal-table" );

let selector3 = $( ".table-heading" );

let selector4 = $( ":input" );

let selector5 = $( ":radio" );

let selector6 = $( ":checkbox" );

let selector7 = $( ":visible" );

let selector8 = $( ":hidden" );

let selector9 = $( "tr:odd" );

let selector10 = $( ".row:has(#animal-table)" );

console.log('$("\*") selected: ' + selector1.length + ' element(s)');

console.log('$("#animal-table") selected: ' + selector2.length + ' element(s)');

console.log('$(".table-heading") selected: ' + selector3.length + ' element(s)');

console.log('$(":input") selected: ' + selector4.length + ' element(s)');

console.log('$(":radio") selected: ' + selector5.length + ' element(s)');

console.log('$(":checkbox") selected: ' + selector6.length + ' element(s)');

console.log('$(":visible") selected: ' + selector7.length + ' element(s)');

console.log('$(":hidden") selected: ' + selector8.length + ' element(s)');

console.log('$("tr:odd") selected: ' + selector9.length + ' element(s)');

console.log('$(".row:has(#animal-table)") selected:'+selector10.length + 'element(s)');

// Accessing the elements (and applying a style):

selector9.each(function(index){ (this).css({"background-color":"red"}); });

**// "Wiring" up Event Listeners**

$(".row").on("change", ":input", function(){ // watch existing "row" div elemennts for when input elements change

console.log("id: " + $(this).attr("id") + "changed");

});

$("document").on("click", "tr", function(){ // watch the whole document for when existing (or new) tr elements are clicked

console.log("table row clicked!");

});

**// Modifying the DOM using jQuery**

// create a new div

let newDiv = $('<div>'); // can also specify using '<div></div>'

// add some CSS

newDiv.css({"border":"1px solid lightgray", "padding":"10px"});

// add some content (HTML)

newDiv.html("<span>New Div!</span>");

// append it to "new-content"

$("#new-content").append(newDiv); // we could also call ".remove() to remove it"

// clone newDiv

let newDiv2 = newDiv.clone();

// Add an attribute to the cloned Div

newDiv2.attr("id", "clonedDiv1");

// Add a class to the cloned Div

newDiv2.addClass("bg-color-lightgray");

// wrap the cloned Div in an 'outer' div

newDiv2.wrap("<div class='outer'></div>");

// set it's text to "Hello World!"

newDiv2.text("Cloned Div!");

// output it's parent's (newly added <div class='outer'></div>) HTML to the console (this will show the whole new div)

console.log(newDiv2.parent().html());

// replace the "Replace Me!" text with the new div

$("#to-be-replaced p").replaceWith(newDiv2);

// Modify the value in the "exampleInputEmail1" text input

$("#exampleInputEmail1").val("from jQuery!");

});

**Performing an AJAX query using jQuery**

$(function() { // this is the same as: $(document).ready(function(){

// store a reference to $("#employees-table body")

let tbody = $("#employees-table tbody");

$.ajax({

url: "http://localhost:8081/employees",

type: "GET",

//data: JSON.stringify({ some: "data" }),

contentType: "application/json"

})

.done(function (employees) {

for(let i = 0; i < 5; i++){ // only show the first 5 employees

// Create each table cell for the c-urrent employee row and add the text

let fNameTD = $('<td>').text(employees[i].FirstName);

let lNameTD = $('<td>').text(employees[i].LastName);

let positionTD = $('<td>').text(employees[i].Position.PositionName);

// append all table cells to a new row

let row = $('<tr>').append(fNameTD).append(lNameTD).append(positionTD); // create the row and append all of the TD elements.

// append the row to the table body

tbody.append(row);

}

})

.fail(function (err) {

console.log("error: " + err.statusText);

});

});