MIS2402 Style Guide (v1.0)

The purpose of this style guide is to outline the JavaScript, HTML, and jQuery language elements that are taught in MIS2402. The intention of the style guide is to establish clear expectations between the student and the instructor, and to nurture clear, step-by-step thinking. Clear, step-by-step thinking helps the student to adapt to programming languages other than JavaScript in the future.

This style guide will evolve over time, with clarifications and additions being made as needed.

Topic List (WARNING: Not an exhaustive list)

The following is a list of topics that students should have confidence in by the end of the semester. This confidence should extend to combining these topics together to create solutions that may be different from what students have previously encountered.

Obviously, students will not be immediately aware of each topic. But, as the semester progresses, more and more items on this list will become familiar. Ultimately, competence with every topic on the list is expected.

There is an upper limit to how much detail such a listing can provide. The ultimate authority on what is / is not part of the course is the content of the lectures, not this list.

<table>
<tr>
<td>1. HTML
a. <table> and related tags</td>
<td>Example – a two-column table:

<table>
 <tr>
 <th>Column Header 1</th>
 <th>Column Header 2</th>
 </tr>
 <tr>
 <td>A</td>
 <td>B</td>
 </tr>
 <tr>
 <td>D</td>
 <td>E</td>
 </tr>
</table></td>
</tr>
<tr>
<td>b. and
c. and </td>
<td>Examples:
<table><tr><td>
 First thing
 Second thing
</td><td>
  thing
  another thing
</td></tr></table></td>
</tr>
<tr>
<td>d. The anchor tag: <a></td>
<td>Example:

Temple University</td>
</tr>
</table>

e. <form> f. <input>	Example: <form id="thestuff"> <input type="text" id="thecustomerid" name="customer"> <input type="button" id="thebuttonid" value="Click me"> </form>
g. h. i.	Examples: It was the best of times, it was the worst of times. Hello World

2. JavaScript commands and syntax a. declare a variable with let b. algebraic expressions c. functions / return d. console.log e. parseInt() f. parseFloat() g. toFixed() h. isNaN() i. Math.ceil() j. Math.floor() k. Math.random()	Example: // define a function to convert Celsius to Fahrenheit function c2f(c){ let answer = (c * 9 / 5) + 32; return answer; } // console log is only good for debugging and testing console.log(c2f(100) + " degrees Fahrenheit."); /* Not shown here: students should be able to add, subtract, multiply, divide, modulo, add one with ++, subtract one with -- */ // more examples let x = parseInt("3.145"); // x will be 3 let y = parseFloat("3.145"); // y will be 3.145 let a = 3.145; // this variable is used below let b = a.toFixed(2); //b will be the string "3.15" console.log(isNaN(a)); // false because 3.145 is a number let c = Math.ceil(a); // c will be 4 let d = Math.floor(a); // d will be 3 /* The variable r will be a random floating point number greater than or equal to 0 and less than 1 */ let r = Math.random();

l. "if" statements, logical operators and Boolean expressions	Example: // determine your tax rate let grossincome = 12000; let taxrate = 0; if (grossincome == 0){ taxrate = 0; } If (grossincome < 11600){ taxrate = .10; } // the comparison operators are == , != , < , > . <= , >= // not shown here. // students should also know how to use else // student are also responsible for use of &&, ||, and ! let watertempC = 40; if (watertempC <= 0){ console.log("Ice"); } else if (watertempC < 100){ console.log("Liquid"); } else { console.log("Steam"); }

m. "for" loops	Example: // this for loop writes 1 through 100 to the console log for(let i=1; i<=100;i++){ console.log(i); } // this for loop sums the numbers 0 to 990 // storing the sum in a variable called answer let answer = 0; //this needs to be declared before the loop for(let i=0; I < 100; i++){ answer = answer + i; }

n. Simple objects / arrays	Examples: // A numerically indexed array of fruits let fruits = ["apples", "grapes", "pears"]; // A simple object let person = { "firstname" : "Sam" , "lastname" : "McHiggens" , "age" : 24}; // A two dimensional object – two "rows" and three "columns" let people = [{ "firstname" : "Sam" , "lastname" : "McHiggens" , "age" : 24}, { "firstname" : "NoName" , "lastname" : "McHiggens" , "age" : 22}];

3. jQuery selectors a. # is used to select a tag by its id attribute 4. Query methods a. .html() b. append() c. .addClass() d. .removeClass() e. .hide() f. .show() g. .val() h. .serialize()	Examples: // assign the inner html $("#message").html("Try again."); // add to the existing inner html $("#message").append("Try again."); // add two classes at the same time $("#message").html("alert alert-danger"); //remove all classes $("#message").removeClass(); //hide a tag $("#message").hide(); //show a tag $("#message").show(); //retrieve the value of an input tag let custname = $("#thecustomerid").val(); //serialize all the data on a form let the_serialized_data = $("#thestuff").serialize();

5. jQuery events
    a. .click()

Example:

```
$("#thebuttonid").click(function(){
  //work happens here
  console.log(“the button was clicked”);
});
```

6. jQuery methods used to make Ajax calls
    a. $.getJSON()
    b. $.post()

```
// simplest case – a GET with no querystring parameters

let url = "https://someserver.com/api"; // this is an example

$.getJSON( url,function(result){
  console.log(result);
});

// the_serialized_data holds querystring parameters
$.getJSON( url, the_serialized_data, function(result){
  console.log(result);
});

// the_serialized_data holds data to be sent in the POST
$.post( url, the_serialized_data, function(result){
  console.log(result);
});
```

7. Bootstrap
    a. The container class
    b. The row class
    c. The col class
    d. Common classes (“alert”,”btn”,”form-control”,”text”)
    e. Common contextual classes (“alert-success”,”alert-primary”,”alert-danger”, etc.)

Example:

```
<div class="container">
  <div class="row">
    <div class="col-md-12">
      <div class="alert alert-primary">
        Hello World!
      </div>
    </div>
  </div>
</div>
```

CONTINUED

DO NOT USE

The intention of the style guide is to document the finite subset of JavaScript, HTML, and jQuery language elements that are expected of students in MIS2402. There are an infinite number of incorrect approaches. Consequently, this "do not use" list is not an exhaustive list. It does, however, attempt to identify the elements most commonly misused by students. They are presented in the table below, in no particular order.

Element	Note
===	The triple equal sign is the "strong comparison" operator. The distinction between strong and weak comparison is not taught in MIS2402. Don't use it.
Math.min()	The min method of the Math object is not used in MIS2402. Use comparison operators instead.
// Abbreviated if statement syntax if (a == b) console.log("yes");	In this example, the if statement is used without a code block {}. There are rare situations where this *sometimes* works. But what *always* works is the correct use of the code block. Expected usage: if (a == b) { //code blocks are cool. Use them. console.log("yes"); }
// Another abbreviated if syntax let age = 20; let message = age >= 18 ? "Adult" : "Minor";	Confusing, isn't it? This abbreviated syntax isn't taught in MIS2402 and MIS2402 students should not use it.
// loop structures other than "for" **for** (key **in** object) { // code block to be executed } **for** (variable **of** iterable) { *// code block to be executed* } **while** (*condition*) { *// code block to be executed* } **do** { *// code block to be executed* } **while** (*condition*)	Confusing, isn't it? These loop structures are not taught in MIS2402 and MIS2402 students should not use them. MIS2402 students only need to use the simplest loop structure. // the simple for loop let text = ""; for (let i = 0; i < 5; i++) { text = text + "The number is " + i + " "; } Students should not use the for/in, for/of, while, or do/while loop structures
//push and pop methods let stuff = ["A", "BB", "CCC", "DDDD"]; stuff.push("EEEEE"); let item = stuff.pop(); console.log(item);	Confusing, isn't it? We don't need the push and pop methods in MIS2402 and you should not use them.

//isInteger method of the Number object Number.isInteger(3.145); //this is false Number.isInteger("3"); //this is false and misleading Number.isInteger(3); //this is true	We don't use the Number object anywhere else in the class. The isInteger method only works on numbers. This is potentially misleading for a student who forgets to parse a string into a number.
// string template literals let animal = "fox"; console.log(`the quick brown ${animal} jumped over the lazy dog`);	This technique for constructing a string is unique to JavaScript. However, all languages support some form of concatenation. Use concatenation instead. let animal = "fox"; console.log("the quick brown " + animal+ " jumped over the lazy dog");
/*Use of the document object for click events and other manipulation of the HTML DOM*/ document.addEventListener('keydown', (event) => { console.log(`Key pressed: ${event.key}`); }); //or let tag = document.getElementById("message") tag.innerHTML = "Hello world";	We have jQuery for DOM manipulation. JQuery is a library that makes DOM manipulation code easier to type, easier to read, and easier to code. DOM is short for Document Object Model. It is the representation of an HTML document in the browser's memory.
//Object.keys const person = { name: "Alice", age: 30, city: "New York" }; // Get an array of keys const keys = Object.keys(person); console.log(keys); // Output: ["name", "age", "city"]	We don't use the Object object anywhere else in the class. Under no circumstances will you need to dynamically determine the presence of a key in a collection of object keys using the .keys() method.
// JavaScript alerts alert("Hello world");	Use of alerts is generally bad and should be avoided.