nkatz01\_13128128 JavaScript (2018\_2019) BBK\_SSCS019H5 7/5/2019

**Understanding what is happening**

With this regard, my preference a majority of the time is JavaScript because when then compared with the html file, I **understand** better what the JS does. I often find it more intuitive after the fact and I find that because of the relatively limited number of options available, to picking up elements to work with them, the JavaScript code will then be less dependent on the specific html content and attributes. As an example, whilst in my contactform.jq file, I’m targeting a specific id attribute $(‘#title) when resetting to the default of ‘Title’ every time the user refreshes the page, in my diabetestool.js file, I use the document.getElementByTagName(‘input’) statement, a more general way of setting the default value for the fields, something which I prefer. In general, after working with JavaScript, I’ve learnt more about what is happening under the hood than from working with jQuery. Examples include, realising what other nodes there are in the DOM model (e.g. text nodes) which aren’t immediately obvious but one gets to find out when working with collections and seeing what operations can and can’t be done on the various objects and nodes picked up.

**Succinctness of code**

However, due to the detail I have to spell out at every operation and due to the long windedness of the JavaScript syntax, I found JQuery code more succinct and easier to **read** (e.g. shorter function names, chaining abilities and built in .each loop); especially after leaving the code and coming back to it a couple of days later. When I am not interested in the inner workings for the moment but rather to refresh to my mind, the narrative of the code, JQuery certainly adds value to the traditional JS. That’s why In the contactform\_jq.js file, I’ve taken the liberty to have shorter validation functions and have the code of the ‘picking-up-fields’ operations as well as the preliminary functions (processing upon refreshing the page etc.) written inside the main $().submit(function(event) function{}) body as I felt it would still be succinct enough for the reader to follow whereas in the contatform.js.

**Event handling**

Of particular usefulness were the binding functions that JQuery allowed me to use; whereas in JavaScript I had a significant bigger amount of overhead to do this (e.g. first query whether a particular id exists or has been changed already so that not to attempt an operation on a null object, pick up the element, assign to a temporary variable, target the id or the innerhtml etc. ) with JQuery event handling syntax done together with a binding function, I felt this was much easier done, shorter effort and at the appropriate location in the code.

**Targeting elements**

Because I wanted to remain as much general as possible, especially in the diabetes tool and being mindful that getElementsByTagName and getElementByClass don’t work across all browsers I tried to keep, wherever possible, to using getElementByID. However, given the nature of IDs that they are specific, in order to get to them in the first place (that is, when I wanted to collect the values of the tool that the user had selected, for processing), I’ve elected to use the other methods to begin with. Given the different types these methods return (e.g. single object versus collection of objects) and then the operations available on them or attributes associated with them (which are not as well standardized as JQuery), this meant I’ve had to go through a number of functions (since only one array per function, for return, was allowed) and repeated loops until I finally got a list of values to work with; something which I’m sure would have been much easier and less complicated to achieve had I used JQuery for it.