ASSESSMENT 3 – ITERATION 2

GOAL:

The current goal is to further develop the JavaScript scripts used to present the election information from the controller to the HTML page.

We’ll need to consider how *Election* in the *Controller* is setting up the data for output. I will need to update the UML class and sequence diagram to reflect the relationships between the newer classes and pre-existing classes. These classes will need to be tested by creating a set of Jasmine specifications to validate the outputs of the new classes (as well as their properties and methods).

The deliverable for this iteration is the following output on a webpage...



PLAN:

Some of the Jasmine specifications are no longer relevant for the current iteration, so those need to be looked at. New methods and properties will need to be identified and applied to the existing class system. I have received an updated Controller.js which contains election data to be presented in a webpage.

DESIGN:

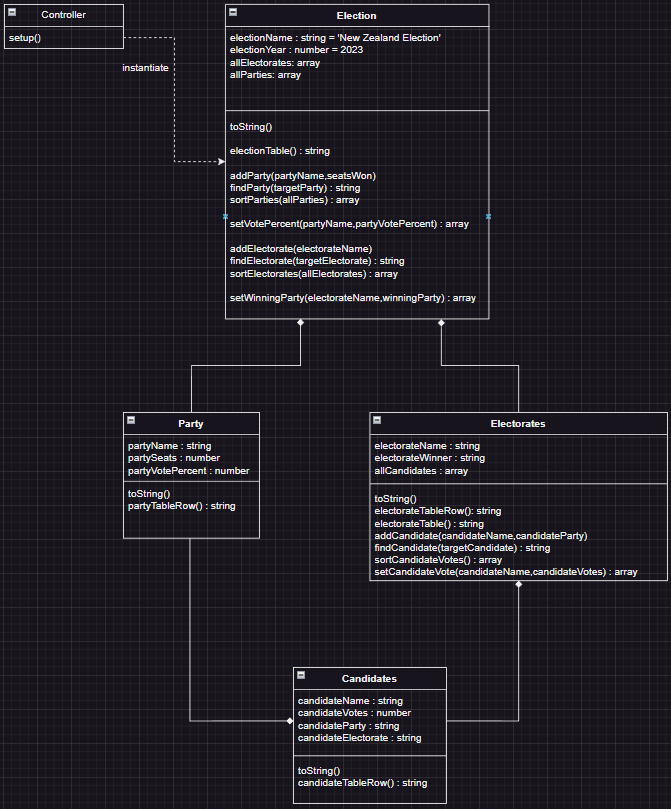
Adding new UML sequence diagram and updating UML class diagram.

TESTING:

Write Candidate Class, creating additional properties and methods for other classes. Checking everything works as designed.

1 WHERE?

*UML Class Diagram of System for This Iteration*



Planned Task Sequence

|  |  |  |  |
| --- | --- | --- | --- |
| **TASK SEQUENCE** | **DETAILS** | **ESTIMATE** | **ACTUAL** |
| PLANNING | * Identify relationships between classes | 30 min | 1 hour |
| DESIGN | * Create sequence diagram, update class diagram | 30 min | 10 min |
| CODING | * Code new class, attributes, and methods * Write code to validate output in Jasmine | 1hour | 4 hours |
| TESTING | * Check that everything works together according to design. | 1 hour | 1 hour 30 min |
| - | * - | 3 hours | 7 hours |

Unit Test

The following properties and methods will be updated in the process of testing and development:

* addParty(party,voteCount) // add vote count parameter
* addElectorate(electorate) // remove party parameter
* setWinningParty(electorate,winningParty) // new method
* setVotePercent(party,votePercent) // new method
* findElectorate(electorate) // new method
* addCandidate(candidate,party) // new method
* setCandidateVote(candidate,votecount) //new method

A screenshot of a computer

Description automatically generated

**Style Compliance**

Jshint

*Election.js*



Figure 2-JSHint: Election.js

*Party.js*

A screenshot of a computer

Description automatically generated

Figure 3-JSHint: Party.js

*Electorate.js*

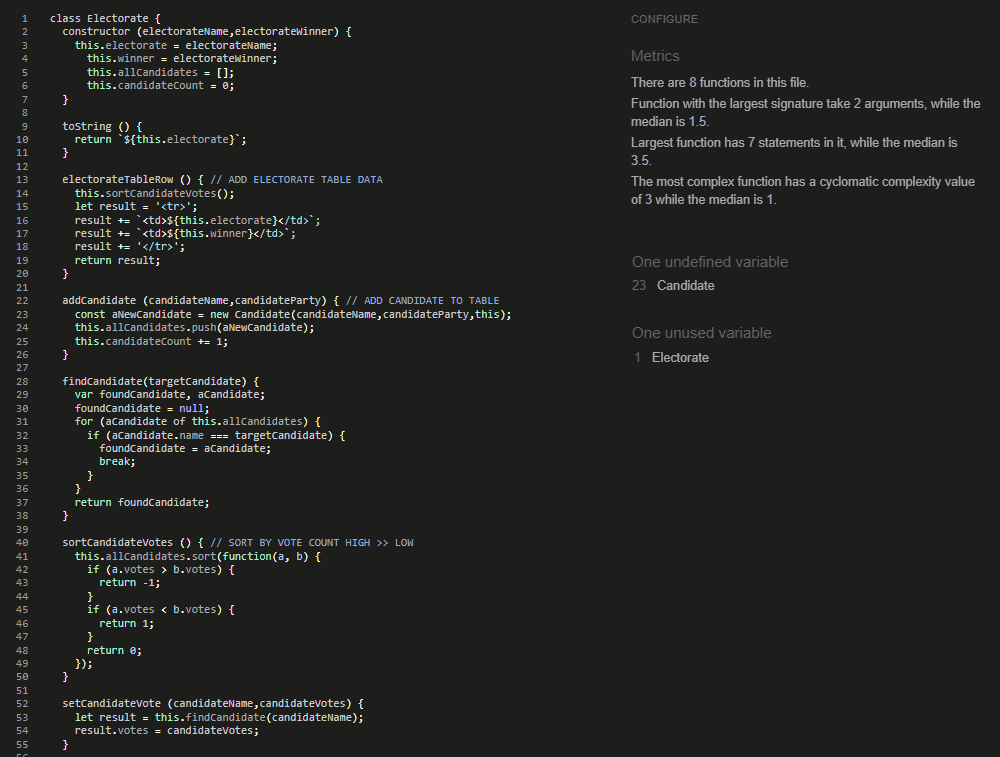


Figure 4-JSHint: Electorate.js

*Candidate.js*

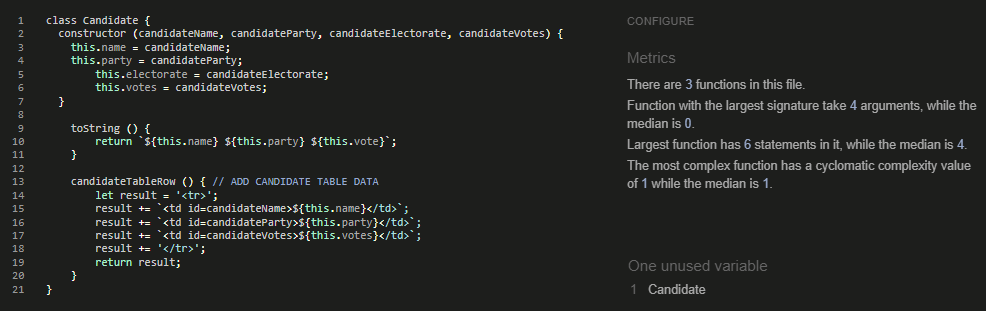


Figure 5-JSHint: Candidate.js

W3c html style validators

*index.html*

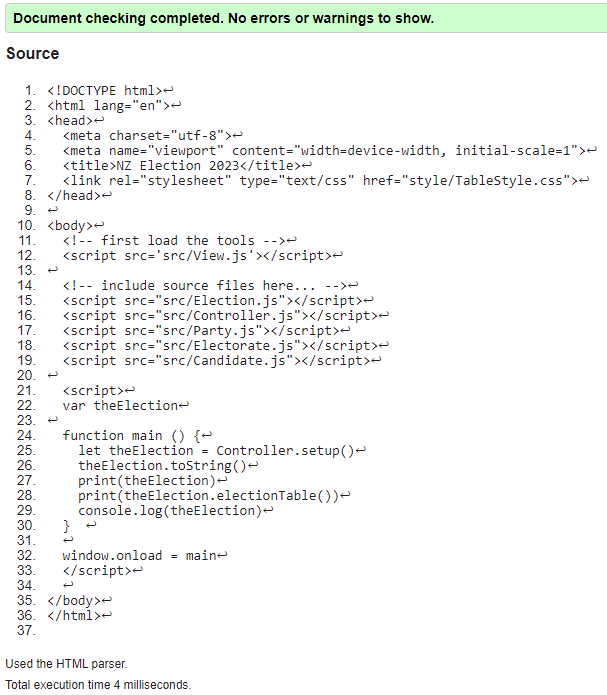


Figure 6-Nu Html Checker: index.html

**How Did the Plan Fare?**

*Unit Test Results*

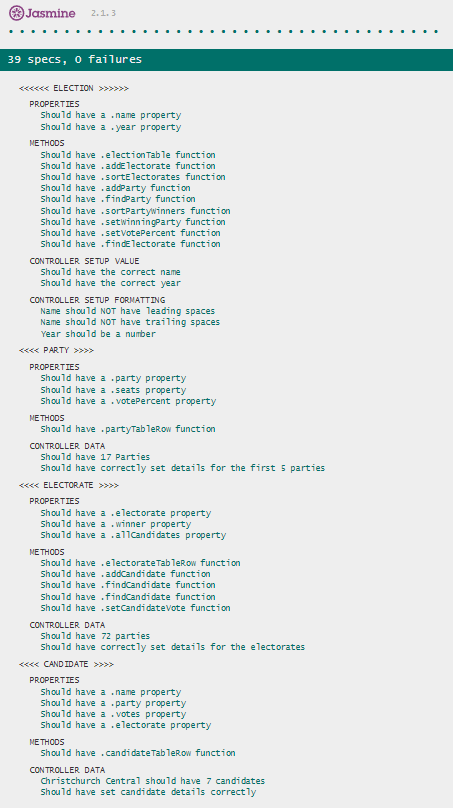


Figure 7-Jasmine Unit Tests

*Page Display*

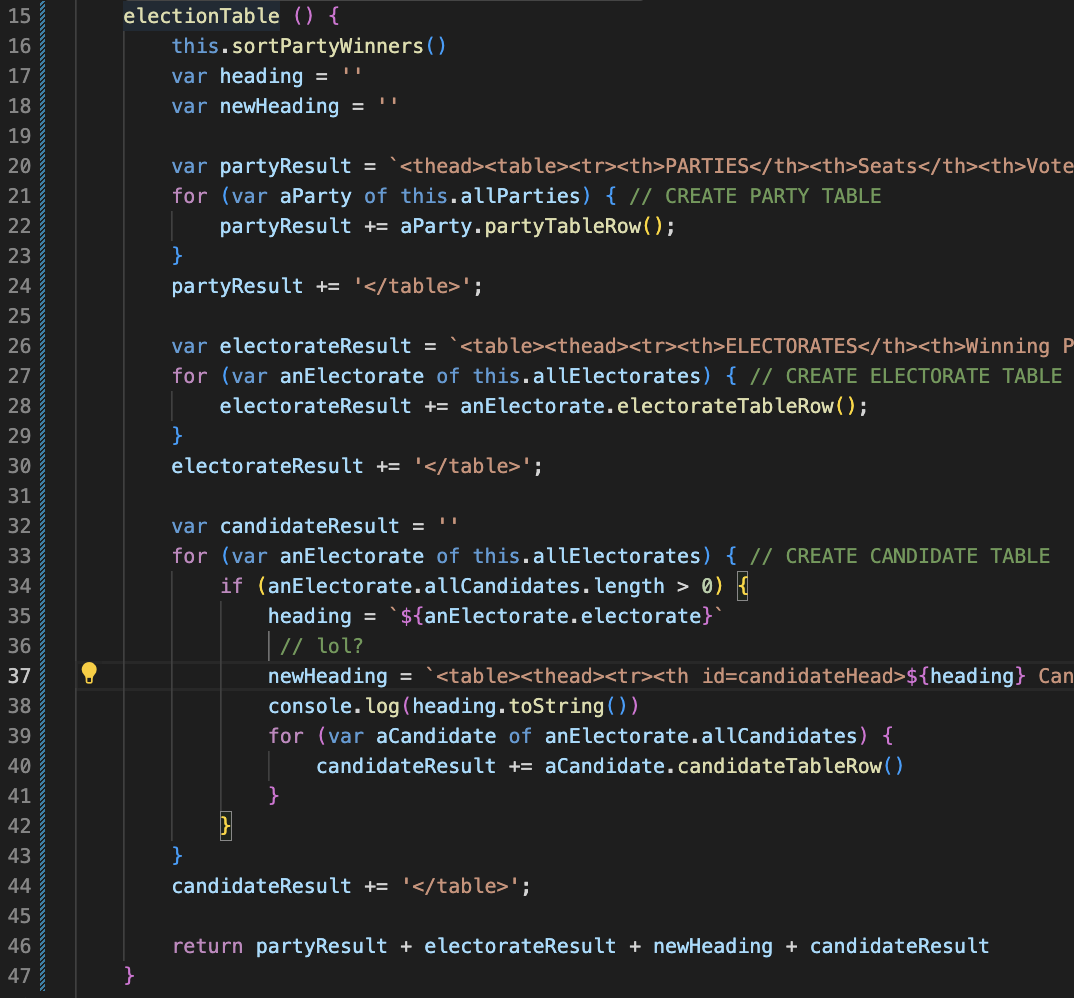
A screenshot of a computer

Description automatically generated

Figure 8-Actual Output

**Deviations From Iteration Plan**

* Didn’t draw up the sequence diagram for class interaction.
* Conditional logic to display electorate candidates isn’t correct (see line 34).



**Discussion of Performance**

|  |  |  |
| --- | --- | --- |
| What worked/didn’t work. | Why | What could be done differently next time |
| I couldn’t work out how to specifically target an electorate to access candidate array inside Christchurch Central electorate. | A possibility is that I didn’t take the time to write up a sequence diagram, which may help with identifying how to | I will maintain some self-discipline and plan everything that needs to be done, before attempting to code a solution. |
| Writing the Jasmine unit tests was much easier than it seems. | Having practical examples of the unit tests being used to validate specifications from software engineering course. | - |
| Having the self-discipline to plan the features before attempting to incorporate them into the project. | Low morale. | As |