CSC 441 Rubric: JavsScript Code Artifacts

The following rubric is used to assess the level of student learning in CSC 441 as it relates to writing JavaScript code.

- For purposes of computing an assignment grade, any criteria deemed **Unacceptable** shall have a numeric value of (50%).
- An **Unacceptable** grade is earned when the evaluation deems the work submitted is worse than what is stated as **Acceptable**.
- All criteria are weighted equally when averaging for an assignment grade.

Criteria	**Acceptable (C=75%)**	**Better (B=85%)**	**Best (A=100%)**
Separation of Concerns	• Two functions should be split into smaller functions to improve separation of concerns. • Two globally scoped variables are not necessary.	• One function should be split into smaller functions to improve separation of concerns. • One globally scoped variable is not necessary.	• Each function performs a distinct part of the processing and cannot be split into smaller functions. • Use of globally scoped variables appropriate.
Design for Reuse	• Two functions could be eliminated by generalizing other functions. • Missed two opportunities to create a more general function.	• One function could be eliminated by generalizing another function. • Missed one opportunity to create a more general function.	• No function could be eliminated by generalizing another function. • No (reasonable) opportunity to create a more general function.
Design Only What is Needed	• One “minor” assignment requirement has not been implemented. • Solution contains two “extra” processing steps that are relatively small and not required.	• Same as *Best*. • Solution contains one “extra” processing step that is relatively small and not required.	• All assignment requirements have been implemented. • Solution contains zero “extra” processing steps that are relatively small and not required.
Number of Logic Defects Found	• Two.	• One.	• Zero.
Number of Syntax Defects Found	• Not applicable.	• Zero.	• Not applicable.
Function Comments	• Each function has a comment that describes the purpose of the method. • Comments accurately describe all but two functions.	• Same as *Best*. • Comments accurately describe all but one function.	• Each function has comments matching one of the two styles shown below. • Comments accurately describe each function.
Readability	• 3 or 4 identifier names do not describe the purpose of the item. • Minor variations in code indentation makes it harder to read the source code.	• 1 or 2 identifier names do not describe the purpose of the item. • Same as *Best*.	• All identifier names describe the purpose of the item. • Code indentation makes it easy to read the source code.
Consistency with Current Project Scope	• Implementation is consistent with all but 2 items in the project scope.	• Implementation is consistent with all but 1 item in the project scope.	• Implementation is consistent with project scope.
Language Features	• Uses 1 inappropriate language feature.	• Not applicable.	• Uses appropriate features.

Two styles for function comments: Describe ***Purpose***, ***Inputs***, and ***Outputs*** or describe ***Purpose***, ***Pre-conditions***, and ***Post-conditions***.

Last Updated: September 14, 2020