General Usage:

- Making a Custom Autocompleter Using Triple “Get” Functions:
    - Inside of registered autocompleter, change the “get” entry to:
      get: {
        getSubject: function(...){ ... }, (or String Array)
        getPredicate: function(...){ ... }, (or String Array)
        getObject: function(...){ ... } (or String Array)
      }
    - Each function’s parameters are in the following order:
        - context (Object)
        - subjectClass (String or undefined)
        - subject (String or undefined)
        - predicate (String or undefined)
        - object (String or undefined)
        - token (Object)
        - callback (Function)
    - Operates exactly as normal YASQE “get” Function / Array, with matching return values, “token”, and “callback”
- Using Partial Match Filter:
    - Assigned at “yasqe.autocompleters.partialMatchFilter(list, token)”
    - list (Array or Object)
        - Array of strings that should be checked
        - Object with keys only as strings that should be checked
    - token (String or undefined)
        - String to search for inside of “list”
        - Undefined will default to returning all values
    - Returns an array of strings that match entries for “list” based on any matches to “token” without case sensitivity

- Simple Local Definitions:
    - Provides a simple dictionary style lookup for triple statements
    - yasqe.autocompleters.addLocalDefinition(subject, predicate, object)
        - subject, predicate, object (Strings or undefined)

        - addLocalDefinition("<my_subject>")
          Adds only "<my_subject>" to suggestions for class and subject sections
        - addLocalDefinition("<my_subject>", "<my_predicate>", "\"My Object\"")
          Adds subject, predicate, and object to corresponding sections
        - addLocalDefinition(null, "<my_predicate>")
          Currently illegal as subject has to be a String
    - yasqe.autocompleters.removeLocalDefinition(subject, predicate, object)
        - subject, predicate, object (Strings or undefined)

        - removeLocalDefinition("<my_subject>")
          Removes the "<my_subject>" class and ALL children
        - removeLocalDefinition("<my_subject>", "<my_predicate>", "\"My Object\"")
          Removes the "My Object" object from the "<my_subject>" "<my_predicate>" chain of Strings
        - removeLocalDefinition(null, "<my_predicate>")
          Currently illegal as subject has to be a String

YASQE Changes:

- Allow Multiple Autocompleters to Provide Suggestions:
    - Previously, once an autocompleter was discovered to be valid and had returnable suggestions all other completers would be ignored. Now, all completers will be tested to see if suggestions exist and contribute to the same displayed list.
    - If any autocompleters use an asynchronous callback to collect suggestions, the suggestion list will not display until all return, therefore implemented completers are responsible for timeout after short period.

- Extend Autocompleter “get” Functionality:
    - Previously, registered autocompleters used a “get” entry that could either be an array or function that returned an array to retrieve suggestable strings. Now, in addition to being a function or array, “get” can also be an object that contains three optional keys for each of the three parts of a triple statement, called appropriately based on cursor’s current location when suggestions are requested:
        - getSubject: function / array
        - getPredicate: function / array
        - getObject: function / array
    - If an array, acts exactly like normal “get” array with changes provided in remaining documentation.
    - If a function that returns an array, uses the same header:
      function(context, subjectClass, subject, predicate, object, token, callback)
        - context: Object that contains additional pieces of information for autocompleter implementation
            - cursor: See “cursor” for “Triples Function” in “YASQE Additions”
            - triples: See “data” for “Triples Function” in “YASQE Additions”
            - variables: An array of all variable strings used in the current triple statement block
        - subjectClass: See “Assign Subject Class” in “YASQE Additions”
        - subject: String representing the subject of the triple statement, can be partial text if requesting from “getSubject”, “undefined” if not written
        - predicate: String representing the predicate of the triple statement, can be partial text if requesting from “getPredicate”, “undefined” if not written
        - object: String representing the object of the triple statement, can be partial text if requesting from “getObject”, “undefined” if not written
        - token: Token object from normal “get” implementation (not string)
        - callback: Callback function from normal “get” implementation when used
- Modified Displayed Suggestions:
    - Previously, the returned value for any “get” function or array, or value passed to the “callback,” could only be an array of strings. Now, in addition to strings, entries in the returned array can also be an array of two strings.
        - [0]: String listed in suggestion list, also string that is used to search for matches
        - [1]: String inserted into query if entry is selected
    - If any “get” item is an array, they will be filtered using “Partial Match Filter” in “YASQE Additions” rather than the previous exact prefix matching. Functions are not automatically filtered, however can use the same filter if used in the autocompleter.

- Previous Token Function:
    - On the “yasqe.getPreviousNonWsToken” function, a minor change was made where the 3rd parameter could previously only be a “token” value retrieved by other YASQE functions unlike the twin function of “getNextNonWsToken.” To implement changes and provide a more uniform use, it now matches where it can also be a character position number. (Only useful when implementing very specific functionality)
- Variables Autocompleter:
    - Rather than requiring prefix to start with “?” or “$” and match exactly based on prefix substrings, variables will now be suggested based on partial matches, but only when the cursor is in the “subject” position or in the “object” position while it is not a class assignment.

YASQE Additions:

- Assign Subject Class:
    - Provides single level attempt to scan current triple statement block to recover the assigned class of the current statement. If found, the token object used for several functions on an autocompleter will have an entry named “subjectClass” which represents the string of the found class, “undefined” otherwise.
        - If subject is not a variable: subject itself is assigned to “subjectClass”
        - If subject is a variable: All lines are scanned to find assignment of matching variable to a class using “a,” “rdf:type” and full URI
- Suggestion History:
    - Prior to displaying the suggestions, the list is passed through a sorting function that uses previously confirmed suggestions to influence the list. Any items that have been selected and inserted into the query will be added to the history, with later entries appearing higher and older ones being lower.
        - If an entry is not in the history: Will be unmoved and below entries in the history
        - If an entry is in the history: Will be moved to the top based on the order of other entries that appear in the history.

- Partial Match Filter:
    - Due to the repeated use of checking if the current written text is contained in any suggestion to be displayed to the user, a default filter is located at “yasqe.autocompleters.partialMatchFilter(list, token)”
        - list: The list of possible, unfiltered strings to suggest to the user
            - If it is an object: Keys on the object will be inspected for matches to “token”. Keys must be strings on entire object.
            - If it is an array: Values in the array will be inspected for matches to “token”. Values must be strings in entire array. Can also handle two string arrays described by “Modified Displayed Suggestions” in “YASQE Changes”
        - token: The string to use for checking matches. Is used as a lowercase, contains method of partial match. If an empty string is provided, will result in all valid entries in “list” to be added.
        - Returns an array of valid suggestions.
- Triples Function:
    - Previously, each autocompleter would need to create their own method of discovering their current position on the document and manually extract usable data. Now, a function “yasqe.getTriples(forSuggestion, useBuffer)” exists to minimize the manual implementation.
        - forSuggestion: If “true,” will specifically use more checks to verify if the cursor is indeed within the bounds of a triple statement block, otherwise if “false” will ignore the checks and attempt to extract some form of data. Recommended to set this to “true” for implementing autocompleters.
        - useBuffer: Due to the size of the function, setting this to “true” will use previously generated data if it exists in order to not redo the same large amount of work. If “false,” the function will then force the regeneration of returnable data.
        - Returns the following object with entries if all checks are valid, “undefined” otherwise:
            - cursor: Array that specifies the current location of the cursor within the “data” block
                - [0]: The line in the triple block / first layer index in “data”
                - [1]: The word on a line in the triple block / second layer index in “data”
            - data: Array of arrays that represent the triple statement block decomposed into lines and words for each line
                - First layer: All lines of the triple block, found by the termination characters of “.” and “;”
                - Second layer: All words for each specific line of the triple block

- Class Designators:
    - In various parts of autocompleters and core functionality, there are times where detecting if a class assignment is required. Possible strings of class assignment predicates (“a”, “rdf:type”, “https://www.w3.org/1999/02/22-rdf-syntax-ns#type”) will be stored inside “yasqe.autocompleters.classDesignators” in the form of an object key lookup. Any entries should be strings and should equal “true” on the object for fast searching.

Provided Implementations:

- Local Definition Autocompleter:
    - Uses loaded strings to define a simple chain of suggestions in the appropriate “subject,” “predicate” and “object” positions based on the location of the cursor in the current query triple. Also anticipates assignment of classes.
    - Uses the ID of “local_definitions” for YASQE to enable, disable, etc.
    - Uses provided “Partial Match Filter” in “YASQE Additions” to recover potential suggestions from provided strings.
    - Is not loaded by default in case it is not wanted, file can be loaded to register autocompleter.
    - Provides three functions on the “yasqe.autocompleters” object:
        - addLocalDefinition(subject, predicate, object): Adds a suggestion for the chain of “subject,” “predicate,” “object.” All parameters are not required, however data will only be correctly inserted if there are no undefined parameters prior to the last one used, ie: “addLocalDefinition(“sub”, null, “obj”)” would not be valid. All used parameters must be strings. “object” can also be a two string array as described by ”Modified Displayed Suggestions” in “YASQE Changes.”
        - getLocalDefinition(subject, predicate): Gets the structure for the designated layer. “predicate” is not required. “object” is not used as multiple entries reside in the array that makes up a predicates “object” values.
        - removeLocalDefinition(subject, predicate, object): Removes a definition based on a previously added chain. Removing any chained entry will remove itself and anything after it, ie: “removeLocalDefinition(“sub”, “pred”)” will remove “pred” and all of it’s objects from the “sub” subject. Parameters do not need to be exact items entered to “addLocalDefinition,” only their contents have to match.