```
/**
 * @description - This file is intended to serve the standard commenting guide for JAVASCRIPT using ECMAScript 5 (ES5).
 * Before executing this script we expect some packages & modules are already installed & running on your Linux/Windows Environment.
 * The soul purpose of this file is to guide how the various syntax are to be explained in a fashion, So as the anonymous reader
 * of this file will have clear understanding of all the i/o behavioral and logical statements that are implemented.
 *
 * JAVASCRIPT STYLE GUIDE - We have specified `JAVASCRIPT STYLE GUIDE` where ever it is required to follow, just to explain
 * how the standards need to be maintained. While doing commenting we expect you to write the comments as and when the code is written.
 *
 * DEPENDENCIES & PACKAGES - Expect the nodejs & npm installed
 */

/**
 * JAVASCRIPT STYLE GUIDE - SOURCE FILE HEADER STYLE
 * NOTE - A SINGLE BLANK LINE SEPARATES THE TWO BLOCKS i.e. BETWEEN HEADER, PACKAGE,
 *        REQUIRE STATEMENTS, CLASS DECLARATION, etc
 */

/******************************************************************************
 *  Execution       :   1. default node         cmd> node PrimeChecker.js
 *                      2. if nodemon installed cmd> nodemon PrimeChecker.js
 *
 *  Purpose         : Determines whether a number `input number` is prime or not.
 *
 *  @description
 *
 *  @file           : PrimeChecker.js
 *  @overview       : Prime checker module to check if number is prime or not.
 *  @module         : PrimeChecker - This is optional if expeclictly its an npm or local package
 *  @author         : BridgeLabz <admin@bridgelabz.com>
 *  @version        : 1.0
 *  @since          : 06-08-2017
 *
 ******************************************************************************/

/**
 * @description JAVASCRIPT STYLE GUIDE - Modern JavaScript practice should always evoke the "Use Strict"; pragma.
 * The `use strict` directive is new in JavaScript 1.8.5 (ECMAScript version 5).
 */
"use strict";

/**
 * JAVASCRIPT STYLE GUIDE - CLASS INSTANCE VARIABLE DECLARATION STYLE
 * NOTES - ALL CLASS VARIABLE SHOULD BE DECLARED IN THE TOP AFTER STATIC DECLARATION IF ANY.
 *       - CLASS MEMBER VARIABLE NAME STARTS WITH 'm' TO INDICATE CLASS VARIABLE. FOLLOWED
 *         BY THE VARIABLE NAME IN CAMELCASE
 *       - FOR EVERY CLASS VARIABLE THERE NEEDS TO BE A COMMENT DESCRIBING THE NEED FOR THE VARIABLE
 */

/**
 * @description - ECMAScript 5 (ES5)
 * JAVASCRIPT STYLE GUIDE - REQUIRE STATEMENTS
 * NOTE : NO WILDCARD IMPORT i.e. import * ;
 *        NO LINE WRAPPING - APPEARS IN ONE LINE
 *        IMPORT ONLY NECESSARY VARIABLES, FUNCTIONS, OBJECTS, ARRAYS, etc.
 */

/**
 * @description Dependencies require to be installed before the execution of this file.
 * @var {Class} prompt class instance of the prompt
 */
var prompt = require("prompt");

/**
 * @description JAVASCRIPT STYLE GUIDE - GLOBAL IMMUTABLE CONSTANT DECLARATION STYLE
 * NOTES - ALL CONSTANT VARIABLE SHOULD BE DECLARED IN THE TOP & SHOULD BE LOADED WHERE THEY ARE REQUIRED.
 *         FOR EVERY CONSTANT VARIABLE THERE NEEDS TO COMMENT
 *       - CONSTANT VARIABLE ARE DECLARED AS ALL CAPS SEPARATED BY _ IF MULTIPLE WORD
 */

/**
 * @description Constant Variable is declared to use to define the message for Prime Number Message.
 * @var {string} PRIME_NUMBER_MESSAGE
 */
var PRIME_NUMBER_MESSAGE = " is a prime number";

/**
 * @description Constant Variable is declared to use to define message if the number is not Prime Number.
 * @var {string} NOT_PRIME_NUMBER_MESSAGE
 */
var NOT_PRIME_NUMBER_MESSAGE = " is not a prime number";

/**
 * JAVASCRIPT STYLE GUIDE - CLASS DECLARATION STYLE
 * NOTE - ORDER OF DECLARATION - STATIC VARIABLE, CLASS VARIABLES, DEFAULT CONSTRUCTOR,
 *        PARAMETERIZED CONSTRUCTOR, FUNCTIONS, TO STRING FUNCTION, STATIC FUNCTIONS AND
 *        FINALLY MAIN FUNCTION.
 *      - EACH TIME A NEW BLOCK OR BLOCK LIKE CONSTRUCT IS OPENED, THE INDENT INCREASES BY THREE
 *        SPACES.
 *      - COLUMN LIMIT 100 CHARACTERS PER LINE
 *      - ONE STATEMENT PER LINE
 */

/**
  * @description Class PrimeChecker
  *
  * @class PrimeCherker
  * @extends {Number}
  */
var PrimeCherker = function(number) {

    /**
     * @description mIsPrime variable stores if the number is prime or not.
```

```javascript
     */
    this.mIsPrime = false;

    /**
     * @description mInputNumber variable stores the number entered by the user as the input.
     */
    this.mInputNumber = 0;

    /**
     *
     */
    if (typeof number !== "undefined") {
        this.exec(number);
    }
};

/**
* JAVASCRIPT STYLE GUIDE - CLASS STATIC FUNCTION DECLARATION STYLE
* NOTES - CLASS FUNCTION IS DECLARED IMMEDIATELY AFTER CONSTRUCTOR
*       - FOR EVERY FUNCTION THERE NEEDS TO BE A COMMENT DESCRIBING THE PURPOSE
*       - FUNCTION DECLARATION STARTS WITH A SMALL CASE AND THEN CAMEL CASE FOR EVERY WORD IN
*         THE FUNCTION
*/

PrimeCherker.isPrime = function(number) {

    /**
     * JAVASCRIPT STYLE GUIDE - EVERY BLOCK NEEDS TO HAVE THE COMMENT
     * NOTES - LEAVE SINGLE BLANK LINE AFTER EVERY LOOP
     *       - NO NEED FOR CURLY BRACES {} IF THE CONDITION HAS ONLY ONE STATEMENT
     */

    /**
     * @description Default set the isPrime to true.
     * @var {boolean} isPrime Boolean default true
     */
    var isPrime = true;

    /**
     * @description Any number less than 2 is prime number
     */
    if (number < 2) return isPrime;

    /**
     * @description try all possible factors of n
     * if n has a factor, then it has one less than or equal to sqrt(n),
     * so for efficiency we only need to check factor <= sqrt(n) or
     * equivalently factor*factor <= n
     */
    for (var factor = 2; factor * factor <= number; factor++) {

        /**
        * @description if factor divides evenly into n, n is not prime, so break out of loop
        */
        if (number % factor == 0) {
            isPrime = false;
            break;
        }
    } // End of for loop

    return isPrime;
};

/**
 * JAVASCRIPT STYLE GUIDE - CLASS FUNCTION DECLARATION STYLE
 * NOTES - CLASS FUNCTION IS DECLARED IMMEDIATELY AFTER CLASS DECLARATION
 *       - FOR EVERY FUNCTION THERE NEEDS TO BE A COMMENT DESCRIBING THE PURPOSE
 *       - FUNCTION DECLARATION STARTS WITH A SMALL CASE AND THEN CAMEL CASE FOR EVERY WORD IN
 *         THE FUNCTION
 */

/**
 * @description Prototype property adding the property functions, Define all the functions here.
 *
 * @method setNumber() - Set the mInputNmber variable
 * @method getNumber() - Get the mInputNumber value
 * @method toString()  - Concatenating number & string text
 * @method isPrime()   - Check if mInputNumber is prime or not indirectly calling the static function isPrime
 * @method exec()      - Exec act returning function and returns the result.
 */
PrimeCherker.prototype = {

    /**
     * @description Setter for setting the number `input number`
     * @param {Number} number A valid input number is expected
     */
    setNumber: function(number) {
        this.mInputNumber = number;
    },

    /**
     * @description Getter for getting the number
     * @function getNumber Get Number saved in the instance
     * @return {Number} expected to return the value which is saved in mInputNumber which is accessible inside the classs only.
     */
    getNumber: function() {
        return this.mInputNumber;
    },

    /**
     * @description Ideally toString is not required here as concatenating the number & string is easily done `+` operator
     * Convert the message to string.
     * @function toString Convert the string
     * @return {String} Convert the mixed variable to string.
     */
    toString: function() {

        /**
        * @description Initialize the message variable
        * @var {string} message
        */
        var message = "";

        /**
        * @description Based on boolean value of mIsPrime we need to return the result.
        */
        return (
            this.mInputNumber +
            (this.mIsPrime ? PRIME_NUMBER_MESSAGE : NOT_PRIME_NUMBER_MESSAGE)
        );
    },

    /**
     * @description Checking the prime number if the number is prime or not.
     * @return {boolean} Returns boolean true/false if the number is prime.
     */
    isPrime: function() {

        /**
        * @description Invoke the static function
        */
        return PrimeCherker.isPrime(this.mInputNumber);
    },

    /**
     * @description  The main function is written to test PrimeChecker class
     *
     */
    exec: function(number = 0) {

        /**
        * @description DEclaring & setting the isPrime variable.
        * @var {boolean} isPrime data type Boolean
        */
        var isPrime = false;

        /**
        * @description Verify the number if its valid INT Number
        */
        try {

            /**
             * @description Check if the number is defined
             */
            if (typeof number === "undefined") {

                /**
                 * @description Not a valid input number
                 */
                throw "No valid input provided.";
            }

            /**
             * @description Check if the number valid number
             */
            if (isNaN(number)) {

                /**
                 * @description concatenating the variable & text
                 */
                throw `${number} is not a valid number.`;
            }

            if(typeof(number, 'Number') && !!(number % 1)){
                throw `${number} is not a valid integer number.`;
            }
            /**
             * @description Call the `setNumber(number)` to set instance member `mInputNumber`
             */
            this.setNumber(number);
        } catch (e) {

            /**
             * @description Handling the exception error thrown
             */
            console.log("PLEASE ENTER VALID INPUT: " + e);
            return;
        }

        /**
         * @description Method 1 - using static function executing the prime checker
         */
        isPrime = PrimeCherker.isPrime(number);
        console.log("PRINTING RESULT USING PrimeChecker STATIC isPrime FUNCTION");
        if (isPrime) {
            console.log(number + PRIME_NUMBER_MESSAGE);
        } else {
            console.log(number + NOT_PRIME_NUMBER_MESSAGE);
        }

        /**
         * @description Method 2 - using instance function of Prime Checker
         */
        isPrime = this.isPrime();
        console.log("PRINTING RESULT USING PrimeChecker INSTANCE FUNCTION");
        if (isPrime) {
            console.log(this.mInputNumber + PRIME_NUMBER_MESSAGE);
        } else {
            console.log(this.mInputNumber + NOT_PRIME_NUMBER_MESSAGE);
        }
        return;
    }
};

/**
 * @description IIFE (Immediately Invoked Function Expression) this anonymous function is executed right after it's created,
 * not after it is parsed. And the function inside get called.
 */
(function() {

    /**
     * @description Call the prompt start function. We wanted the program to be available immediately
     * @func call start();
     */
    prompt.start();

    /**
     * @description
     * @func get Calling the get function and passing the callback function
     */
    prompt.get(["input number"], function(err, result) {
        /**
         * @description Invoke the PrimeChecker();;
         */
        var primeChecker = new PrimeCherker(result["input number"]);
    });
})();
```

Details
General Info
Type
Javascript
Size
11 KB
Location
JavaScript Comments
Modified
5:55 PM Mar 7
Created
4:01 PM Aug 11, 2017
Opened by me
10:41 AM Mar 8
Sharing
BridgeLabz Solutions LLP
Can View
D
Dilip More
Owner
Description
No description
Download Permission
Viewers cannot download