# Използвани стандартни конвенции

1. За именуване на променливи и функции е използвана CamelCase конвенцията, както и описателни имена, даващи точна представа за какво се използва въпросната променлива или функция.

Пример:

***var expextedResult;***

1. За именуването на константи се използват само главни букви и долна черта за отделяне на думите съставящи името на константата.
2. Отварящата скоба на функции, цикли и условни оператори се на мира на същия ред, където е декларирана съответната функция, цикъл или условен оператор.
3. Затварящата скоба на функции и цикли се намира на нов ред в края на съответната функция или цикъл.
4. При условните оператори else се пише на същия ред където е затварящата скоба на кореспондиращия му if.

Пример:

***if (isDependent) {***

*for (i = 0; i < valuesArray.length; i += chunk) {*

*tempArray = valuesArray.slice(i, i + chunk);*

*chunkedArray.push(tempArray);*

*}*

***} else {***

*for (i = 0; i < valuesArray.length; i += chunk) {*

*tempArray = valuesArray.slice(i, i + chunk);*

*tempArray.unshift(1);*

*chunkedArray.push(tempArray);*

*}*

*}*

1. Try-catch блокове – отварящата скоба на try блока се на мира на същия ред където се дефинира начало на блока. Затварящата скоба се пише на нов ред в края на блока и след него следва ключовата дума catch, с която се дава начало на catch блока.
2. Кодът е подравнен и форматиран според изискванията за четимост, като всеки отделен блок от код е отместен с една табулация навътре спрямо предходния му (parent) ред (декларация на функция, условен оператор, цикъл, try-catch блок или клас/структура).

Пример:

*return new Promise(*

*(resolve, reject) => {*

*var fieldCounter = 0;*

*for(var option in args) {*

*++fieldCounter;*

*}*

*if(fieldCounter != 4) {*

*return reject('Invalid number of data set fields.');*

*}*

*if (rows < 1 || columns < 1) {*

*return reject('Invalid values for rows/columns.');*

*}*

*if (valuesArray.length != rows \* columns) {*

*return reject('Invalid number of values in array.');*

*}*

*valuesArray.forEach(function (element) {*

*if (isNaN(element)) {*

*return reject('Invalid type of value in array.');*

*}*

*}, this);*

*var chunkedArray = [];*

*var chunk = columns;*

*if (isDependent) {*

*for (i = 0; i < valuesArray.length; i += chunk) {*

*tempArray = valuesArray.slice(i, i + chunk);*

*chunkedArray.push(tempArray);*

*}*

*} else {*

*for (i = 0; i < valuesArray.length; i += chunk) {*

*tempArray = valuesArray.slice(i, i + chunk);*

*tempArray.unshift(1);*

*chunkedArray.push(tempArray);*

*}*

*}*

*return resolve(math.matrix(chunkedArray));*

*});*

*};*