JavaScript-examiner User Manual Ingredients

Manual

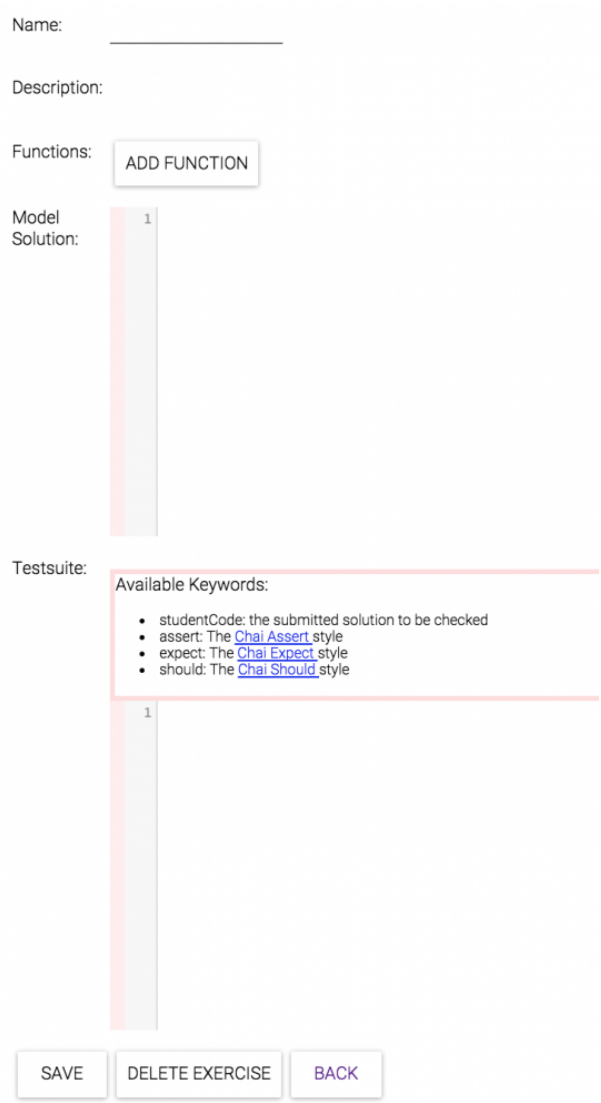
1 — Last update: 2015/05/13

JavaScript-examiner

Table of Contents

Create an exercise (tutor)

Click ‘add’ to create a new exercise.



-Fill in at least the ‘Name’.

-By adding a ‘Model Solution’, the metrics of this solution is compared with the students solution, and the results are presented to the student. The syntax and style formatting rules for the Model Solution are equal to the related rules of a regular submitted solution. Example model solution:

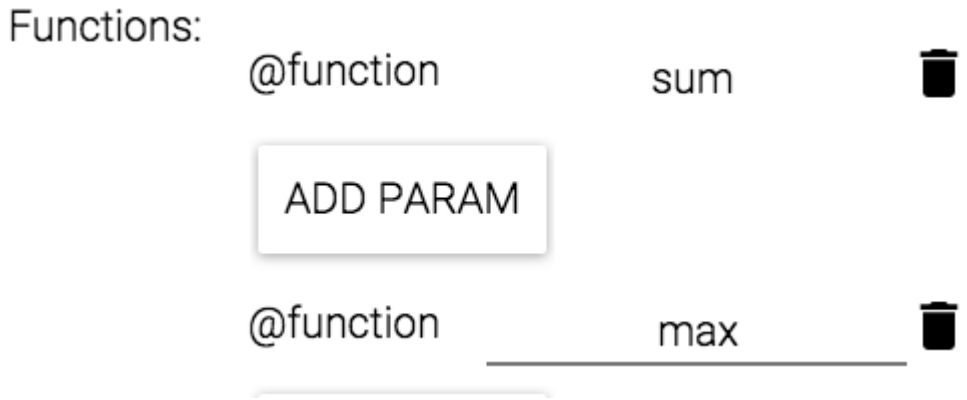
Model Solution:

```
function sum(numarray) {
  var sum = 0;
  numarray.forEach(function(x) {sum = sum + x;});
  return sum;
}

function max(numarray) {
  var max = numarray[0];
  numarray.forEach(function(x) {
    max = (x > max) ? x : max;
  });
  return max;
}

```

-By adding a ‘Testsuite’, the submitted solution will be checked against this solution. In this case, it’s required to define the functions that are tested, by using ‘Add Function’. The function can be called in the testsuite through ‘studentCode.‘. Example test suite:



!https://cdn.manula.com/user/4822/img/large/test-suite.png!

Testsuite:

Available Keywords:

- studentCode: the submitted solution to be checked
- assert: The Chai Assert style
- expect: The Chai Expect style
- should: The Chai Should style

```
1  describe('Tests calculate sum of arrayelements:', function() {
2    describe('Function sum: ', function() {
3      it('should exist', function() {
4        //here the 'studentCode' keyword is used to call the specied function
5        //the keyword 'expect' is used as well
6        expect(studentCode.sum).to.be.a('function');
7      });
8      it('should correctly calculate a non-empty array : ', function() {
9        //here the keyword 'assert' is used
10       assert.strictEqual(6, studentCode.sum([1, 2, 3]), '[1,2,3]');
11     });
12   });
13 });
14 describe('Tests calculate max of arrayelements:', function() {
15   describe('Function max: ', function() {
16     it('should exist', function() {
17       expect(studentCode.max).to.be.a('function');
18     });
19     it('should correctly calculate a non-empty array : ', function() {
20       assert.strictEqual(8, studentCode.max([1, 8, 3]), '[1,8,3]');
21     });
22   });
```

-When saving the exercise, the model solution and testsuite is checked on syntax and style. When these checks pass, the exercise is saved to the database, and in case there is both a solution and testsuite present, the model solution is tested against the test suite. The results of the checks will be presented. Result based on former examples:

Test Results: 4 out of 4 tests passed

Passed tests:

- ✓ Tests calculate sum of arrayelements: Function sum: should exist
- ✓ Tests calculate sum of arrayelements: Function sum: should correctly calculate a non-empty array :
- ✓ Tests calculate max of arrayelements: Function max: should exist
- ✓ Tests calculate max of arrayelements: Function max: should correctly calculate a non-empty array :

SAVE | Exercise Saved. | DELETE EXERCISE | BACK

Make an exercise (user)

1. Select an exercise from the list:

Exercises

- Genereer is wortel van x functies
- Object naar string
- Bereken Bmi
- add-multi
- Bereken faculteit
- Bereken oppervlakte circel
- Genereer add x functies
- plus
- add-multi
- sum and max

2. The selected exercise is the example exercise used here:

sum and max

Write a function that takes an array param and returns the sum of all numbers within this array. Write another function that takes an array param and returns the max number within this array.

SUBMIT YOUR SOLUTION

1

3. Try to solve the exercise, and submit when finished. Based on the submitted solution, feedback is presented:

- In case the syntax is not correct, the error and location of the error will be presented, by pointing the mouse on the exclamation:

```
1 function sum(arr) {
2   sum = 0;
3   return arr[0] + arr[1];
4 }
5 //there is a } to many:
6 }
7
```

Line 6: Unexpected token }

- In case the syntax is correct, but the style does not comply to the defined [JSCS style rules](:“google”:https://github.com/jscs-dev/node-jscs/blob/master/presets/google.json by default) those errors will be shown in the same manner:

```
1 function sum(arr) {
2      //incorrect indentation
3    var sum = 0
4   return arr[0] + arr[1];
5 }
6 //multiple linebreaks
7
8
```

Multiple line break

- In case both syntax and style are ok, the functionality is checked (based on the testsuite defined):

```
1 function sum(arr) {
2   var sum = 0
3   return arr[0] + arr[1];
4 }
5
```

Test results

1 out of 4 tests passed

Passed tests:

✓ Tests calculate sum of arrayelements: Function sum: should exist

Failed tests:

✕ Tests calculate sum of arrayelements: Function sum: should correctly calculate a non-empty array : : [1,2,3]: expected 6 to equal 3

✕ Tests calculate max of arrayelements: Function max: should exist: expected undefined to be a function

✕ Tests calculate max of arrayelements: Function max: should correctly calculate a non-empty array : : undefined is not a function

- In case both syntax, style and functionality are ok, the maintability results are shown. Both the solution result as the
  model solution (if defined) results are shown, for both the full code as per function. The lower the number, the better the result:

Maintainability metrics

	Your code	Model solution
Maintainability score	140.6285606463648	142.84377870035652
Cyclomatic density	18.181818181818183	20

sum

	Your code	Model solution
Cyclomatic density	33.33333333333333	33.33333333333333

<anonymous>

	Your code	Model solution
Cyclomatic density	100	100

max

	Your code	Model solution
Cyclomatic density	25	33.33333333333333

<anonymous>

	Your code	Model solution
Cyclomatic density	200	200

Besides the feedback, the full metrics of both the studentcode and modelsolution are shown:


```
{
 "modelMetrics": {
  "aggregate": {
   "sloc": {
    "logical": 10,
    "physical": 13
   },
   "halstead": {
    "operators": {
     "identifiers": [
      "function",
      "var",
      "=",
      "()",
      "+",
      ".",
      "return",
```

The abstract syntax tree of the presented as well, whenever the syntax check passes:

Abstract Syntax Tree

The abstract syntax tree of your code


```
{
  "end": {
    "file": null,
    "comments_before": [],
    "nlb": true,
    "endpos": 242,
    "endcol": 1
```