LibamtrackWeb Documentation

Arkadiusz Dudzik Piotr Zmilczak

Aug 15, 2023

Contents

CHAPTER 1

Preparing environment

1.1 Requirements

To install and run LibamtrackWeb on your computer you need to have node.js server installed and web browser which supports JavaScript scripts.

Note: (Windows users) Be sure that you have Node Package Manager - npm - set in system path.

1.2 How to install

1. Clone repository to local directory:

```
git clone https://github.com/libamtrack/web.git
```

2. Go into cloned repository:

```
cd ./web
```

3. Install required dependencies:

```
npm install
```

4. Run app on local server:

```
npm run start
```

1.3 Deploying app on GitHub Pages

To deploy LibamtrackWeb app on GitHub Pages:

1. Specify GitHub Pages URL in **package.json**:

```
"homepage": *URL*
```

2. Run command:

```
npm run deploy
```

CHAPTER 2

How to add new function

If you want to add new function to LibamtrackWeb all you need is to follow this steps:

1. Prepare special JavaScript wrapper function to use compiled from C library function. Use JSFunctionGenerator to quickly get wrapper method from C function's signature. More details how to use this tool you can find clicking green button 'show help' on the tool's page.

> **Warning:** During development remember to restart your local server, because auto import plugin generates dependencies to functions during start.

2. Put wrapper function into */src/functionsFromC* directory.
3. Prepare JSON that describes functions. See details in *Function configuration JSON*.
4. Put prepared JSON into directory */src/static/json/<category_name>/*
5. Add all required dictionaries to */src/static/json/dictionaries/*. See details in *Dictionary JSON*.
6. Prepare new category or add function to existing one in *Global Configuration JSON*, which can be found in */src/static/json/GlobalConfig.json*.
7. Rebuild app and test :)

CHAPTER 3

JSONs Configuration

3.1 Global Configuration JSON

Global configuration JSON file looks like below

```
{
    "applicationTitle": "Libamtrack",
    "introText": "libamtrack provides computational routines for the prediction of␣
↪detector response and radiobiological efficiency in heavy charged particle beams.",
    "footerText": "LibatrackWeb ©2022",
    "categories": [],
    "dictionaries": [],
}
```

- **applicationTitle** (*string*) - title of web application
- **introText** (*string*) - short application description displayed in main page
- **footerText** (*string*) - footer - text displayed on end section

3.1.1 Categories

Categories is an array that contains functions grouped in theme categories and optionally customized styles.

```
{
    "categories": [
        {
            "name": "Physics Routines",
            "style": {
                "background": "red",
                "border": "red",
                "color": "black",
                "font-size": "20px"
```

(continues on next page)

(continued from previous page)

```
            },
            "functionsStyle": {
                "color": "red"
            },
            "functions": [
                {
                    "name": "Beta From Energy Single",
                    "jsonConfigPath": "/PhysicsRoutines/BetaFromESingle.json"
                }
            ]
        }
    ]
}
```

- **name** (*string*) - category name which will be displayed in application, (try to use only letters and digits when choosing name, others characters may produce problem during generating route paths)
- **style** (*string*) - customizable CSS for displayed category column on the main page
- **functionsStyle** (*string*) - customizable CSS for each function row in category column

Functions

An array which contains all functions in given category:

- **name** (*string*) - displayed name for function
- **jsonConfigPath** (*string*) - path for JSON file which contains function's details. Format: /<category_name>/<function_name>.json

3.1.2 Dictionaries

An array which contains all dictionary JSONs used by web application

```
{
    "dictionaries": [
        {
            "name": "particles",
            "jsonConfigPath": "/dictionaries/Particles.json"
        }
    ]
}
```

- **name** (*string*) - dictionary name which is used in functions JSON

Note: This name must be equal with the name used in functions configuration JSONs.

- **jsonConfigPath** (*string*) - path to JSON which contains dictionary values: Format: /dictionaries/<dictionary_name>.json

3.2 Function configuration JSON

In this file user can describe all parameters which are needed to generate form for the function.

```
{
    "visibleName": "Mass stopping power with no",
    "functionName": "AT_Mass_Stopping_Power_with_no",
    "description": "Retrieves the electronic mass stopping power in MeV*cm2/g for the␣
↪requested energies and particles for a specified material and data source.",
    "xTitle": "Energy [MeV]",
    "yTitle": "Mass stopping power [MeV*cm2/g]",
    "plot": true,
    "resultUnit": " ",
    "resultPrecision": 12,
    "resultLabel": "Mass stopping power",
    "isMathJaxSupported": true,
    "formItems": [],
    "moreOptions": {},
    "modals": {
        "dataSeries": true,
        "download": true,
        "deleteAll": true
    }
}
```

- **visibleName** (*string*) - name which will be displayed on function page
- **functionName** (*string*) - C/wrapper function name which user will have to generate

Warning: This value MUST BE equal with JavaScript function name user put in src/functionsFromC

- **description** (*string*) - short description what function does, it appears on function page
- **xTitle/yTitle** (*string*) - x/y axis labels
- **plot** (*boolean*) - flag that descrbes whether function returns results on plot (true) or as single value (false)
- **resultUnit** (*string*) - units for functions that return single results
- **resultPrecision** (*int*) - how many significant digits will be displayed in result for single result functions. Default: 12
- **isMathJaxSupported** (*boolean*) - specifies whether to load MathJax script to translate used MathJax expressions in labels/descriptions etc.
- **formItems** (*array*) - form fields described in *Form Items*
- **moreOptions** (*object*) - described in *More options*
- **modals** (*array*) - contains flags describing which elements will be displayed on function page in "Data Series options" section
  - **dataSeries** (*boolean*) - modal window when user can see details about series of data from plot
  - **download** (*boolean*) - modal window for downloading all calculation results
  - **deleteAll** (*boolean*) - button for deleting all calculation results from plot

Note: Parameters "xTitle", "yTitle", "moreOptions", "modals" are ignored when "plot" is set to false

3.2.1 Form Items

An array that contains all inputs, fields etc. needed by function to make calculations. Below, there are described all currently supported types of items.

Common elements:

- **type** (*string*) - specifies form item type
- **parameterName** (*string*) - name of C function parameter that is provided by this item

Warning: "parameterName" MUST BE equal with the name of function argument in C. For example:

```
int test(int a); => "parameterName": "a"
```

- **description** (*string*) - hint that will be displayed when user move cursor on the form item - default: "Insert value"

Entry module

It is a collection of simple form fields that allows to generate a serie of numbers.

```
{
  "type": "entry_module",
  "parameterName": "E_MeV_u",
  "label": "energy [MeV]"
  "startholder": "1",
  "endholder": "1000",
  "intervalType": "points",
  "stepDefaultValue": "1",
  "pointsDefaultNumber": "500",
  "validations": {
        "type": "float",
        "min": "0.0001",
        "max": "10000"
  }
}
```

- **label** (*string*) - string added to "Start" and "End" input labels f.e. for "label": "energy [MeV]" label for first input will be "Start energy [MeV]:" and for second "End energy [MeV]:"
- **startholder** (*float/string*) - initial value for "Start" input
- **endholder** (*float/string*) - initial value for "End" input
- **intervalType** (*string*, ["step", "pointsNo"]) - default value for "Generate" block - default: "step"
- **stepDefaultValue** (*string/float*) - initial value interval type "step" - default: 0.1
- **pointsDefaultNumber** (*string/int*) - initial value for interval type "points" - default: 50
- **validations** (*array*) - array with validation rules for "Start" and "End" inputs
  - **type** (*string*, ["float", "int"]) - number format value - default: "float"
  - **min** (*float/int/string*) - minimal value for inputs
  - **max** (*float/int/string*) - maximum value for inputs

Input

Single input item that allows insert single number.

```
{
  "type": "input",
  "parameterName": "beta",
  "label": "Beta",
  "placeholder": "0.1",
  "defaultValue": "0.1",
  "validations": {
        "type": "float",
        "min": "0.000001",
        "max": "0.999999"
  }
}
```

- **label** (*string*) - name of field visible for user
- **placeholder** (*string*) - value visible when field is empty
- **defaultValue** (*string*) - field initial value
- **validations** (*array*) - field validation rules
  - **type** (*string*, ["float", "int"]) - number format value - default: "float"
  - **min** (*float/int/string*) - input minimal value
  - **max** (*float/int/string*) - input value

Select

Item that allows user to choose some values from list. Lists are provided as *Dictionary JSON*.

```
{
  "type": "select",
  "parameterName": "material_no",
  "label": "Material",
  "values": "materials",
  "defaultValue": 1,
  "description": "Choose material type"
}
```

- **label** (*string*) - name of field visible for user
- **values** (*string*) - dictionary name from *Global Configuration JSON*
- **defaultValue** (*float/int*) - initial value from dictionary object (from property "value")

3.2.2 More options

This fields are responsible for describing types/scales of plot X and Y axis

```
{
    "moreOptions": {
        "visible": true,
        "defaultXAxisType": "log",
```

(continues on next page)

(continued from previous page)

```
        "defaultYAxisType": "linear",
        "plotType": "lines"
    }
}
```

- **visible** (*boolean*) - specifies whether this component is visible on main page or not
- **defaultXAxisType** (*string*, [“log”, “linear”]) - specifies scale of X-axis, if not provided “linear” will be applied
- **defaultYAxisType** (*string*, [“log”, “linear”]) - specifies scale of Y-axis, if not provided “linear” will be applied
- **plotType** (*string*, [“lines”, “points”]) - button that allows to determine if plot will be display as points or line. When not provided button will not be displayed and plot type will be “line”

3.3 Dictionary JSON

Dictionary JSON is an array which contains dictionary values as objects. Each dictionary is listed in web application in order provided in this file.

```
[
  {
      "name": "H",
      "value": 1001
  }
]
```

- **name** - dictionary name - it will be displayed in application
- **value** - value e.g. number which will be used in calculations

Note: Couple name-value should be unique in one dictionary.