This document is incomplete. More details will be coming soon.

Systemstate Editor

Currently at Pre-alpha v0.3.1

Systemstate Editor Pre-alpha v0.3.1

This document is incomplete. More details will be coming soon.

Introduction

Thinkers of the world, welcome to the official documentations of Systemstate Editor Pre-alpha v0.3.1. Do you have burst of ideas in your head that drives you to do things? That unique weltanschauung that radiates out from the deep abyss of your soul? Here is a tool that might be able to help you a bit. Look at our logo. Here, the red fire of passion and revolution shall, though the squarish, rigorous frameworks with outlines that contrasts the goods and evils, blacks and whites of this world, rise again with vigor, and shoot out of the other end of the frame with a flamethrower’s flame, brightly and mightily!

After having bold visions, one ought to thought out the meticulous plans and details. This tool is made for exactly such a purpose by being a structure that vessels human thought with the minimal mental overhead and format restrictions.

This document is incomplete. More details will be coming soon.

List of Hotkeys

Hotkeys can speed up the process of using the Systemstate Editor. For example, there are hotkeys to open each of the four tabs seen on the bottom of the Systemstate Editor:

Alt-1	Alt-2 [1]	Alt-3	Alt-4
Console Interface	Search Engine	JSON Viewer	Graphical UI

[1]: There is also Ctrl-Alt which does the same thing.

Then there are hotkeys to navigate the Console Interface:

Up	Down
Select the previous command	Select the next command

Finally, there are hotkeys that aids the workflow of using the Systemstate Editor:

- Enter: Empty the search bar to search for something else.
- Shift-Enter: Load the first search result into the editor.
- Shift: Load the currently selected key into the editor.

Changing Settings via *settings.json*

Under “client”

- “editor”: Toggle which element should the editor open for you when the Systemstate Editor is initialized. By default, it will be set to null. Set this to a UUID to load that element on initialization.

- “tab”: Toggle which tab to open when the Systemstate Editor is initialized. By default, the Console Interface will be opened. Or else, you have these options: “Console” for the Console Interface, “Search” for the Search Engine, “JSON” for the JSON Viewer and “GUI” for the Graphical UI.

- “view”: By default, a split screen is opened when the Systemstate Editor is initialized. However, you can set this to “Graph” and only see the force-directed graph at initialization, “Split” to see both in a the split screen and “Editor” to only see the editor.

- “device”: Toggle whether or not a touch-screen-accessibility button is required to submit queries in the Console Interface or confirm search results in the Search Engine. By default, this button is omitted. Set this to “Mobile” to have the button or “Desktop” to omit it.

You can also set those parameters at the browser’s address bar directly, with the following syntax:

./index.html?*[key]=[value]*

For which, the keys of the four settings above are “initEditor”, “initTab”, “initView” and “initDevice” respectively.

Under “server”

- “port”: Set which port to host the Systemstate Server. Currently, this is the only setting that can only be changed in the settings.json file. This is because both the client and the server have to look up on this file in order to connect with each other, making it impossible to unilaterally change this setting.

- “debug” [1]: Toggle whether to debug or not. If it is set to true, the Systemstate Server will have a read-eval-print loop for the user to debug things.

- “beautify_jsons” [1] [2]: Toggle whether to beautify the JSONs that appear on the Systemstate Server’s output window.

- “stream_capacity” [2]: Toggle how many bytes should the server read from streams each time. Set this too low (say, 4096 bytes) than the server might only receive half of the file that you are uploading to it. Set this too high (say, 1073741824 bytes) than the server might fail to allocate enough RAM to read the stream.

You can also set these parameters by adding flags and key value pairs in the command prompt with the following syntax:

```
systemstate.exe --[flag] --[key]=[value]
```

[1]: Flags are for boolean type settings. If there is a flag, the setting would be set to true, otherwise it would be set to false. Key value pairs are for other settings.

[2]: Where there are underscores in the parameter names found in the *settings.json* file, replace them with hyphens (-) when adding flags and key value pairs in the command prompt.

Console Interface

Client-side Commands

Prefixing commands with “js.” escapes them from being sent to the Systemstate Server. Instead, the corresponding Javascript function will be called.

- "js.add();"
- "js.load();"
- "js.save();"
- "js.remove();"
- "js.help();"
- "js.terminate();"

Server-side Commands

These commands would be sent to the Systemstate Server after being inputted at the Console Interface. Hence, these commands can also be used if the flag “--debug” is added when opening the Systemstate Editor via the command prompt.

- "generate_uuid();"
- "query();"
- "clear();"
- "undefine();"
- "script();"
- "terminate();"
- "import();"
- "export();"
- "expand();"
- "open();"
- "search();"
- "check();"

This document is incomplete. More details will be coming soon.

Conclusion

The Systemstate Editor has the capacity to turn into a great tool for aspiring thinkers to organize their thought in a both unrestricted and formalized way. Written with the infamous trio of HTML, CSS and Javascript in the client-side and PHP in the server-side, it leverages both the flexibility of a web platform and power of a desktop application. In the future, it can be further developed to integrate the various parts of the software closer together via the localhost and via stream wrappers.

Moreover, since the organization of elements are also computer-readable, using the Systemstate Editor would open the doors to possibilities of more advanced automations, or even machine learning. With machine learning on the rise, certainly, it is not impossible for us to cognitively enhance ourselves in the far future with tools that sharpen minds such as with Elon Musk’s vision of the Neuralink.

For the most profound of aspirations though, the Systemstate Editor strives to create an environment where the logical reliability of the left brain can collaborate with the unrestricted creativity of the right brain without the barriers found in other tools. For instance, writing long articles requires a mental overhead to organize thoughts into a sequential (one sentence after another, one paragraph after another) and hierarchical (letters form words, which form sentences, which form paragraphs) structure. This implies right-brain thinking. On the other hand, the left brain’s boundless imagination produces imageries that cannot be easily expressed, communicated and applied to the pre-existing structure of daily life without big paradigm shifts. The Systemstate Editor strives to become the structure that can vessel imagination, hence, doing both.

The End