node-key-sender

1.0.11 • Public • Published a year ago

Readme

Explore BETA

0 Dependencies

4 Dependents

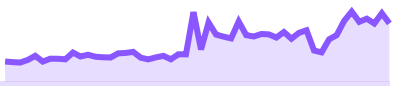
11 Versions

Install

```
> npm i node-key-sender
```

Weekly Downloads

915



Version	License
1.0.11	MIT
Unpacked Size	**Total Files**
84.7 kB	4
Issues	**Pull Requests**
13	2

Homepage

github.com/garimpeiro-it/node-key-sender#readme

Repository

github.com/garimpeiro-it/node-key-sender

Last publish

a year ago

Collaborators

Try on RunKit

Report a vulnerability

node-key-sender

Use this lib to send keyboard events to the operational system.

It uses a jar file (Java), so Java Run Time is required on the operational system you are running your node project (version 8 or above).

Bugs and issues: Please, post any issues to **https://github.com/garimpeiro-it/node-key-sender/issues**.

Main features

- Send raw keyboard key codes to the operational system;
- Send one key;
- Send multiple keys pressed one after the other;
- Send multiple keys pressed together (combination);
- Delay between keys;
- Delay for each pressed key or each combination;
- Possibility to map key codes;

- Case correction for text;
- Multi platform (it will work in all operation systems that Java can run);
- It will send the key to the current focused application in the operational system;
- It is sensitive to the operational system, keyboard driver and phisical keyboard installed on the running platform.

Installation

Install it using npm:

```
npm install --save-dev node-key-sender
```

The command above will install and add the lib into your "package.json" file.

How it works

Each key in your keyboard is mapped with a key code. Although a physical keyboard key may have printed above its surface more than one key (for example ':' and ';'), both generate the same key code. So, do not confuse key codes with ASCII or UNICODE values, they are different things.

To make it clear, lets see an example: In american keyboard, the 16 value is the key code for Shift and the 59 value is the key code for the key ":;". So, in this scenario, to send ':' to the operational system, you should use key code 56. To send ';' you should send 16 + 56 as a combination (pressed together).

In languages that have accents (for example: portuguese and spanish), usually more than one key must be pressed one after another to make the letter with an accent. So 'õ' is the result of sending '~' and 'o'.

While you can send the key codes as numbers, the lib also have labels mapped for most of the keys. So, for key A you may send 'a' or 65. For Shift key you may send 'shift' or 16.

It is possible to change this mapping to convert accents automatically (if you are using a keyboard that supports it). Later in this doc I show how to do that.

Note that key codes may vary according to your running physical keyboard model, keyboard driver and operational system.

Usage

Sending one key:

```
var ks = require('node-key-sender');
ks.sendKey('a');
```

Send multiple keys one after the other:

```
var ks = require('node-key-sender');
ks.sendKeys(['a', 'b', 'c']);
```

Send combination (pressed at the same time):

```
var ks = require('node-key-sender');
ks.sendCombination(['control', 'shift', 'v']);
```

Mapping accents:

```
var accentsMap = {
    'ã': '@514 a',
    'ẽ': '@514 e',
    'ĩ': '@514 i',
    'õ': '@514 o',
    'ũ': '@514 u',
    'Ã': '@514 A',
    'Ẽ': '@514 E',
    'Ĩ': '@514 I',
    'Õ': '@514 O',
    'Ũ': '@514 U',
    'â': 'shift-@514 a',
    'ê': 'shift-@514 e',
    'î': 'shift-@514 i',
    'ô': 'shift-@514 o',
    'û': 'shift-@514 u',
```

```
    'Â': 'shift-@514 A',
    'Ê': 'shift-@514 E',
    'Î': 'shift-@514 I',
    'Ô': 'shift-@514 O',
    'Û': 'shift-@514 U',
    'à': 'shift-@192 a',
    'è': 'shift-@192 e',
    'ì': 'shift-@192 i',
    'ò': 'shift-@192 o',
    'ù': 'shift-@192 u',
    'À': 'shift-@192 A',
    'È': 'shift-@192 E',
    'Ì': 'shift-@192 I',
    'Ò': 'shift-@192 O',
    'Ù': 'shift-@192 U',
    'á': '@192 a',
    'é': '@192 e',
    'í': '@192 i',
    'ó': '@192 o',
    'ú': '@192 u',
    'Á': '@192 A',
    'É': '@192 E',
    'Í': '@192 I',
    'Ó': '@192 O',
    'Ú': '@192 U',
    'ç': '@192 c',
    'Ç': '@192 C',
    'ä': 'shift-@54 a',
    'ë': 'shift-@54 e',
    'ï': 'shift-@54 i',
    'ö': 'shift-@54 o',
    'ü': 'shift-@54 u',
    'Ä': 'shift-@54 A',
    'Ë': 'shift-@54 E',
    'Ï': 'shift-@54 I',
    'Ö': 'shift-@54 O',
    'Ü': 'shift-@54 O'
```

```
};

var ks = require('node-key-sender');
ks.aggregateKeyboardLayout(accentsMap);
ks.sendText("Héllõ Wíth Áççents");
```

Sending batch:

```
var ks = require('node-key-sender');

ks.startBatch()
    .batchTypeKey('N')
    .batchTypeKey('o')
    .batchTypeKey('d')
    .batchTypeKey('e')
    .batchTypeKeys(['N', 'o', 'd', 'e'])
    .batchTypeText('Node')
    .batchTypeKey('N', 1000)
    .batchTypeKey('o', 1000)
    .batchTypeKey('d', 1000)
    .batchTypeKey('e', 1000)
    .sendBatch();
```

Setting global press delay (in milliseconds):

```
ks.setOption('globalDelayPressMillisec', 1000);
```

Setting global delay between keys (in milliseconds):

```
ks.setOption('globalDelayBetweenMillisec', 1000);
```

Setting start delay (in milliseconds):

```
ks.setOption('startDelayMillisec', 1000);
```

Turning off case correction:

```
ks.setOption('caseCorrection', false);
```

List of methods

Economic methods

Use this methods if you want to send just a small amount of keys. Note that the jar file is called each time one of these methods are called:

sendKey(keyCode: string): Promise

Sends one key code.

sendKeys(arrKeyCodes: array): Promise

Sends multiple key codes.

sendLetter(letter: char): Promise

Sends a letter. A letter will be automatically converted to key code according to the keyboard layout configuration. You may set this configuration with `cleanKeyboardLayout`, `setKeyboardLayout` or `aggregateKeyboardLayout`.

sendText(text: string): Promise

Sends a text. A text will have its letters automatically converted to key codes according to the keyboard layout configuration. You may set this configuration with `cleanKeyboardLayout`, `setKeyboardLayout` or `aggregateKeyboardLayout`.

sendCombination(arrKeyCodes: array): Promise

Sends multiple key codes pressed together.

Batch methods

Use this methods to send a large amount of keys. The jar file is executed each time you call `sendBatch`. You should start with `startBatch` and end with `sendBatch`:

startBatch()

Starts a batch.

sendBatch(): Promise

Sends the batch.

batchTypeKey(keyCode: string, waitMillisec: int, batchEvent: int)

Sends a key code. You may pass a delay it will wait until it presses the next key and also the type of event. Type of event is `ks.BATCH_EVENT_KEY_PRESS`, `ks.BATCH_EVENT_KEY_UP` and `ks.BATCH_EVENT_KEY_DOWN`.

batchTypeKeys(arrKeyCodes: array)

Sends a list of key codes, pressed one after the other.

batchTypeCombination(arrKeys: array, waitMillisec: int, batchEvent: int)

Sends a combination - list of key codes that will be pressed together. You may pass a delay it will wait until it presses the next key and also the type of event. Type of event is `ks.BATCH_EVENT_KEY_PRESS`, `ks.BATCH_EVENT_KEY_UP` and `ks.BATCH_EVENT_KEY_DOWN`.

batchTypeText(text: string)

Sends a text. A text will have its letters automatically converted to key codes according to the keyboard layout configuration. You may set this configuration with `cleanKeyboardLayout`, `setKeyboardLayout` or `aggregateKeyboardLayout`.

Keyboard layout methods

Keyboard layout methods affects translation of letter to key code. They affect `sendLetter`, `sendText`, `batchTypeText` and `getKeyCode` methods.

cleanKeyboardLayout(): void

Resets the keyboard layout configuration.

setKeyboardLayout(newKeyMap: object): void

Sets a new keyboard layout. For example:

```
var keyboardLayout = {
    'ç': '@192 c',
    'Ç': '@192 C'
};

var ks = require('node-key-sender');
ks.aggregateKeyboardLayout(keyboardLayout);
ks.sendText("Ç");
```

This keyboard layout converts letter 'Ç' to key codes '@192' and 'C'.

aggregateKeyboardLayout(objKeyMap: object): void

Appends the new mapping to the current mapping.

getKeyboardLayout(): object

Returns the current keyboard layout.

Other methods

getKeyCode(letter: string)

Gets the key code of a letter. A letter will be automatically converted to key code according to the keyboard layout configuration. You may set this configuration with `cleanKeyboardLayout`, `setKeyboardLayout` or `aggregateKeyboardLayout`.

setOption(optionName: string, optionValue: string)

Options that are passed as arguments to the jar. This is the list of valid options names:

- startDelayMillisec (int): Delay in milliseconds it will wait to press the first key.
- globalDelayPressMillisec (int): Global delay in milliseconds it will keep the key pressed.
- globalDelayBetweenMillisec (int): Global delay in milliseconds it will wait until it presses the next key.
- caseCorrection (boolean): When it is on, if you send key 'A' (in upper case), the jar will automatically hold Shift, so resulting key is 'A'.
- extra (string): Use may use it to send raw arguments to the jar file. Example: ' -c 1 -d 1000'.

execute(arrParams: array): Promise

Use this method if you want to directly call the jar file.

Promises

Some methods of this lib returns a promise:

```
ks.sendKey('a').then(
    function(stdout, stderr) {
        // For success
    },
    function(error, stdout, stderr) {
        // For error
    }
);
```

The promise is resolved or rejected right after the jar finishes its execution.

List of methods that returns this promise: `sendCombination`, `sendKey`, `sendKeys`, `sendLetter`, `sendText`, `execute`, `sendBatch`.

List of key codes

We recommend you search for key codes in the Java **java.awt.event.KeyEvent** class doc. The key codes are the constants starting with "VK_". To use it with this lib, just take out these three letters and you can use the rest of the constant name in lowercase. For example, VK_SHIFT constant you use "shift". VK_A constant you use 'a'. The constant numerical value can also be used with an "@" in the beginning. So "@16" for VK_SHIFT and "@65" for VK_A.

Use this website to get an idea of what key code is bound to each key of your current keyboard: **https://www.w3.org/2002/09/tests/keys.html**.

Below, the list of key codes:

Keyboard key	Label key code	Numerical ke

Keyboard key	Label key code	Numerical ke
Enter	"enter"	"@10"
Backspace	"back_space"	"@8"
Tab	"tab"	"@9"
Shift	"shift"	"@16"
Control	"control"	"@17"
Alt	"alt"	"@18"
Pause	"pause"	"@19"
Caps Lock	"caps_lock"	"@20"
Esc	"escape"	"@27"
Space	"space"	"@32"
Page Up	"page_up"	"@33"
Page Down	"page_down"	"@34"
End	"end"	"@35"
Home	"home"	"@36"
Left	"left"	"@37"
Up	"up"	"@38"
Right	"right"	"@39"
Down	"down"	"@40"
Comma	"comma"	"@44"
Minus	"minus"	"@45"
Period	"period"	"@46"

Keyboard key	Label key code	Numerical ke
Slash	"slash"	"@47"
0	"0"	"@48"
1	"1"	"@49"
2	"2"	"@50"
3	"3"	"@51"
4	"4"	"@52"
5	"5"	"@53"
6	"6"	"@54"
7	"7"	"@55"
8	"8"	"@56"
9	"9"	"@57"
Semicolon	"semicolon"	"@59"
Equals	"equals"	"@61"
A	"a"	"@65"
B	"b"	"@66"
C	"c"	"@67"
D	"d"	"@68"
E	"e"	"@69"
F	"f"	"@70"
G	"g"	"@71"
H	"h"	"@72"

Keyboard key	Label key code	Numerical ke
I	"i"	"@73"
J	"j"	"@74"
K	"k"	"@75"
L	"l"	"@76"
M	"m"	"@77"
N	"n"	"@78"
O	"o"	"@79"
P	"p"	"@80"
Q	"q"	"@81"
R	"r"	"@82"
S	"s"	"@83"
T	"t"	"@84"
U	"u"	"@85"
V	"v"	"@86"
W	"w"	"@87"
X	"x"	"@88"
Y	"y"	"@89"
Z	"z"	"@90"
Open bracket	"open_bracket"	"@91"
Numpad 0	"numpad0"	"@96"
Numpad 1	"numpad1"	"@97"

Keyboard key	Label key code	Numerical ke
Numpad 2	"numpad2"	"@98"
Numpad 3	"numpad3"	"@99"
Numpad 4	"numpad4"	"@100"
Numpad 5	"numpad5"	"@101"
Numpad 6	"numpad6"	"@102"
Numpad 7	"numpad7"	"@103"
Numpad 8	"numpad8"	"@104"
Numpad 9	"numpad9"	"@105"
Multiply	"multiply"	"@106"
Add	"add"	"@107"
Subtract	"subtract"	"@109"
Decimal	"decimal"	"@110"
Divide	"divide"	"@111"
Delete	"delete"	"@127"
Num Lock	"num_lock"	"@144"
Scroll Lock	"scroll_lock"	"@145"
F1	"f1"	"@112"
F2	"f2"	"@113"
F3	"f3"	"@114"
F4	"f4"	"@115"
F5	"f5"	"@116"

Keyboard key	Label key code	Numerical ke
F6	"f6"	"@117"
F7	"f7"	"@118"
F8	"f8"	"@119"
F9	"f9"	"@120"
F10	"f10"	"@121"
F11	"f11"	"@122"
F12	"f12"	"@123"
F13	"f13"	"@61440"
F14	"f14"	"@61441"
F15	"f15"	"@61442"
F16	"f16"	"@61443"
F17	"f17"	"@61444"
F18	"f18"	"@61445"
F19	"f19"	"@61446"
F20	"f20"	"@61447"
F21	"f21"	"@61448"
F22	"f22"	"@61449"
F23	"f23"	"@61450"
F24	"f24"	"@61451"
Print Screen	"print_screen"	"@154"
Insert	"insert"	"@155"

Keyboard key	Label key code	Numerical ke
Help	"help"	"@156"
Meta	"meta"	"@157"
Block Quote	"block_quote"	"@192"
Quote	"quote"	"@222"
Numeric Key Pad Up	"kp_up"	"@224"
Numeric Key Pad Down	"kp_down"	"@225"
Numeric Key Pad Left	"kp_left"	"@226"
Numeric Key Pad Right	"kp_right"	"@227"
Grave accent	"dead_grave"	"@128"
Acute accent	"dead_acute"	"@129"
Circumflex accent	"dead_circumflex"	"@130"
Tilde accent	"dead_tilde"	"@131"
Macron accent	"dead_macron"	"@132"
Breve accent	"dead_breve"	"@133"
Above dot accent	"dead_abovedot"	"@134"
Diaeresis accent	"dead_diaeresis"	"@135"
Abovering accent	"dead_abovering"	"@136"
Double acute accent	"dead_doubleacute"	"@137"
Caron accent	"dead_caron"	"@138"
Cedilla accent	"dead_cedilla"	"@139"
Ogonek accent	"dead_ogonek"	"@140"

Keyboard key	Label key code	Numerical ke
Iota accent	"dead_iota"	"@141"
Voiced sound accent	"dead_voiced_sound"	"@142"
Semi voiced sound accent	"dead_semivoiced_sound"	"@143"
Ampersand	"ampersand"	"@150"
Asterisk	"asterisk"	"@151"
Double quote	"quotedbl"	"@152"
Less	"less"	"@153"
Greater	"greater"	"@160"
Brace left	"braceleft"	"@161"
Brace right	"braceright"	"@162"
At	"at"	"@512"
Colon	"colon"	"@513"
Circumflex	"circumflex"	"@514"
Dollar	"dollar"	"@515"
Euro Sign	"euro_sign"	"@516"
Exclamation Mark	"exclamation_mark"	"@517"
Inverted exclamation mark	"inverted_exclamation_mark"	"@518"
Left parenthesis	"left_parenthesis"	"@519"
Right parenthesis	"right_parenthesis"	"@522"
Number sign	"number_sign"	"@520"
Plus	"plus"	"@521"

Keyboard key	Label key code	Numerical ke
Underscore	"underscore"	"@523"
Windows	"windows"	"@524"
Context menu	"context_menu"	"@525"
Japanese PC 106 henkan	"convert"	"@28"
Japanese PC 106 muhenkan	"nonconvert"	"@29"
Japanese PC 106 eisu	"alphanumeric"	"@240"
Japanese PC 106 katakana	"katakana"	"@241"
Japanese PC 106 hiragana	"hiragana"	"@242"
Japanese PC 106 zenkaku	"full_width"	"@243"
Japanese PC 106 hankaku	"half_width"	"@244"
Japanese PC 106 rouman-ji	"roman_characters"	"@245"
Japanese PC 106 zenkouho	"all_candidates"	"@256"
Japanese PC 106 maekouho	"previous_candidate"	"@257"
Japanese PC 106 kanji bangou	"code_input"	"@258"
Japanese PC 106 kana lock	"kana_lock"	"@262"
Japanese PC 106 nihongo	"input_method_on_off"	"@263"
Japanese Solaris kakutei	"accept"	"@30"
Japanese Solaris kana lock	"kana"	"@21"
Japanese kanji	"kanji"	"@25"
Japanese Macintosh katakana	"japanese_katakana"	"@259"
Japanese Macintosh hiragana	"japanese_hiragana"	"@260"

Keyboard key	Label key code	Numerical ke
Japanese Macintosh rouman-ji	"japanese_roman"	"@261"
Sun cut	"cut"	"@65489"
Sun copy	"copy"	"@65485"
Sun paste	"paste"	"@65487"
Sun undo	"undo"	"@65483"
Sun again	"again"	"@65481"
Sun find	"find"	"@65488"
Sun props	"props"	"@65482"
Sun stop	"stop"	"@65480"
Compose	"compose"	"@65312"
Alt GR	"alt_graph"	"@65406"

Note that this is an incomplete list and that the key code may vary according to your physical keyboard, keyboard driver and operational system.

Keywords

nodejs key keyboard input send keystroke os operational system java jar

Help

Documentation

Community

Resources

Advisories

Status

Contact

About

Company

Blog

Careers

Webinars

Press

Newsletter

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy