Keyboard

Keyboard provides an api for managing a virtual keyboard. The high level api is keyboard.type(), which takes raw characters and generates proper `keydown`, `keypress`/`input`, and `keyup` events on your page.

For finer control, you can use keyboard.down(), keyboard.up(), and keyboard.insert_text() to manually fire events as if they were generated from a real keyboard.

An example of holding down `Shift` in order to select and delete some text:

Sync **Async**

```
page.keyboard.type("Hello World!")
page.keyboard.press("ArrowLeft")
page.keyboard.down("Shift")
for i in range(6):
    page.keyboard.press("ArrowLeft")
page.keyboard.up("Shift")
page.keyboard.press("Backspace")
# result text will end up saying "Hello!"
```

An example of pressing uppercase `A`

Sync **Async**

```
page.keyboard.press("Shift+KeyA")
# or
page.keyboard.press("Shift+A")
```

An example to trigger select-all with the keyboard

Sync **Async**

```
# on windows and linux
page.keyboard.press("Control+A")
# on mac_os
page.keyboard.press("Meta+A")
```

Methods

down

Added in: v1.8

Dispatches a `keydown` event.

`key` can specify the intended keyboardEvent.key value or a single character to generate the text for. A superset of the `key` values can be found here. Examples of the keys are:

`F1` - `F12`, `Digit0`- `Digit9`, `KeyA`- `KeyZ`, `Backquote`, `Minus`, `Equal`, `Backslash`, `Backspace`, `Tab`, `Delete`, `Escape`, `ArrowDown`, `End`, `Enter`, `Home`, `Insert`, `PageDown`, `PageUp`, `ArrowRight`, `ArrowUp`, etc.

Following modification shortcuts are also supported: `Shift`, `Control`, `Alt`, `Meta`, `ShiftLeft`.

Holding down `Shift` will type the text that corresponds to the `key` in the upper case.

If `key` is a single character, it is case-sensitive, so the values `a` and `A` will generate different respective texts.

If `key` is a modifier key, `Shift`, `Meta`, `Control`, or `Alt`, subsequent key presses will be sent with that modifier active. To release the modifier key, use keyboard.up().

After the key is pressed once, subsequent calls to keyboard.down() will have repeat set to true. To release the key, use keyboard.up().

NOTE

Modifier keys DO influence `keyboard.down`. Holding down `Shift` will type the text in upper case.

Usage

```
keyboard.down(key)
```

Arguments

- `key` str

  Name of the key to press or a character to generate, such as `ArrowLeft` or `a`.

insert_text

Added in: v1.8

Dispatches only `input` event, does not emit the `keydown`, `keyup` or `keypress` events.

Usage

Sync Async

```
page.keyboard.insert_text("嗨")
```

NOTE

Modifier keys DO NOT effect `keyboard.insertText`. Holding down `Shift` will not type the text in upper case.

Arguments

- `text` str

  Sets input to the specified text value.

press

Added in: v1.8

TIP

In most cases, you should use locator.press() instead.

`key` can specify the intended keyboardEvent.key value or a single character to generate the text for. A superset of the `key` values can be found here. Examples of the keys are:

`F1` - `F12`, `Digit0`- `Digit9`, `KeyA`- `KeyZ`, `Backquote`, `Minus`, `Equal`, `Backslash`, `Backspace`, `Tab`, `Delete`, `Escape`, `ArrowDown`, `End`, `Enter`, `Home`, `Insert`, `PageDown`, `PageUp`, `ArrowRight`, `ArrowUp`, etc.

Following modification shortcuts are also supported: `Shift`, `Control`, `Alt`, `Meta`, `ShiftLeft`.

Holding down `Shift` will type the text that corresponds to the `key` in the upper case.

If `key` is a single character, it is case-sensitive, so the values `a` and `A` will generate different respective texts.

Shortcuts such as `key: "Control+o"` or `key: "Control+Shift+T"` are supported as well. When specified with the modifier, modifier is pressed and being held while the subsequent key is being pressed.

Usage

Sync **Async**

```
page = browser.new_page()
page.goto("https://keycode.info")
page.keyboard.press("a")
page.screenshot(path="a.png")
page.keyboard.press("ArrowLeft")
page.screenshot(path="arrow_left.png")
page.keyboard.press("Shift+O")
page.screenshot(path="o.png")
browser.close()
```

Shortcut for keyboard.down() and keyboard.up().

Arguments

- `key` str

  Name of the key to press or a character to generate, such as `ArrowLeft` or `a`.

- `delay` float *(optional)*

Time to wait between `keydown` and `keyup` in milliseconds. Defaults to 0.

type

Added in: v1.8

CAUTION

In most cases, you should use locator.fill() instead. You only need to press keys one by one if there is special keyboard handling on the page - in this case use locator.press_sequentially().

Sends a `keydown`, `keypress`/`input`, and `keyup` event for each character in the text.

To press a special key, like `Control` or `ArrowDown`, use keyboard.press().

Usage

Sync **Async**

```
page.keyboard.type("Hello") # types instantly
page.keyboard.type("World", delay=100) # types slower, like a user
```

NOTE

Modifier keys DO NOT effect `keyboard.type`. Holding down `Shift` will not type the text in upper case.

NOTE

For characters that are not on a US keyboard, only an `input` event will be sent.

Arguments

- `text` str

  A text to type into a focused element.

- `delay` float *(optional)*

  Time to wait between key presses in milliseconds. Defaults to 0.

up

Added in: v1.8

Dispatches a `keyup` event.

Usage

```
keyboard.up(key)
```

Arguments

- `key` str

  Name of the key to press or a character to generate, such as `ArrowLeft` or `a`.