TypeScript macOS screencapture

Masaya Kameyama

2026-09-02

Introduction

, playwright . `select` , playwright `page.screenshot` , . DOM .

`select` . , , , IME , . playwright , .

, OS . macOS `screencapture` , TypeScript . E2E `e2e/utils` , playwright .

screencapture

. macOS .

```
#     screenshot.png
screencapture screenshot.png

#
screencapture -x screenshot.png

# 3
screencapture -T 3 screenshot.png

#
screencapture -i screenshot.png

#
screencapture -w screenshot.png

#
screencapture -t jpg screenshot.jpg
```

```
#    2
screencapture -D 2 screenshot.png
```

. , exec . . .

, Node.js child_process, fs, path . .

```
import {exec} from "node:child_process";
import * as fs from "node:fs";
import * as path from "node:path";
import {promisify} from "node:util";

const execAsync = promisify(exec);

/**
 *
 */
export interface ScreenCaptureOptions {
  /**          */
  outputPath: string;
  /**          : true  */
  captureFullScreen?: boolean;
  /**                 : false  */
  interactive?: boolean;
  /**          : false  */
  captureWindow?: boolean;
  /**        : false  */
  playSound?: boolean;
  /**        : 0  */
  delay?: number;
  /**      png, jpg, pdf     : png  */
  format?: "png" | "jpg" | "pdf" | "tiff";
  /**       ID 1:      , 2:         */
  displayId?: number;
}
```

exec .

```
export async function takeScreenshot(options: ScreenCaptureOptions): Promise<string> {
  const {
    outputPath,
    captureFullScreen = true,
    interactive = false,
    captureWindow = false,
    playSound = false,
    delay = 0,
    format = "png",
    displayId,
  } = options;

  //
  const outputDir = path.dirname(outputPath);
  if (!fs.existsSync(outputDir)) {
    fs.mkdirSync(outputDir, {recursive: true});
  }

  const args: string[] = [];

  //
  if (!playSound) {
    args.push("-x");
  }

  if (displayId !== undefined && displayId > 0) {
    args.push("-D", displayId.toString());
  }

  if (delay > 0) {
    args.push("-T", delay.toString());
  }

  args.push("-t", format);

  //
  if (interactive) {
    args.push("-i");
  } else if (captureWindow) {
    args.push("-w");
  } else if (captureFullScreen) {
    //
  }
```

```
  args.push(outputPath);

  const command = `screencapture ${args.join(" ")}`;

  try {
    console.log(`  : ${command}`);
    const {stderr} = await execAsync(command);

    if (stderr) {
      console.warn(` : ${stderr}`);
    }

    //
    if (!fs.existsSync(outputPath)) {
      throw new Error(`             : ${outputPath}`);
    }

    console.log(`          : ${outputPath}`);
    return outputPath;
  } catch (error) {
    throw new Error(`           : ${error}`);
  }
}
```

3 .

- : screencapture . ./result/2026-09-02/ , mkdirSync recursive .
- : . , playSound -x . .
- : screencapture 0 , exec . existsSync .

3 . Esc , . , .

takeScreenshot . , .

```
/**      */
export async function captureScreen(filename: string, outputDir = "./screenshots"): Promise<
  return takeScreenshot({
    outputPath: path.join(outputDir, filename),
    captureFullScreen: true,
    playSound: false,
```

```
  });
}

/**         */
export async function captureInteractive(filename: string, outputDir = "./screenshots"): Pro
  return takeScreenshot({
    outputPath: path.join(outputDir, filename),
    interactive: true,
    playSound: false,
  });
}

/**          */
export async function captureWindow(filename: string, outputDir = "./screenshots"): Promise<
  return takeScreenshot({
    outputPath: path.join(outputDir, filename),
    captureWindow: true,
    playSound: false,
  });
}

/**      */
export async function captureWithDelay(
  filename: string,
  delaySeconds: number,
  outputDir = "./screenshots",
): Promise<string> {
  return takeScreenshot({
    outputPath: path.join(outputDir, filename),
    delay: delaySeconds,
    playSound: false,
  });
}

/**         */
export async function captureFromDisplay(
  filename: string,
  displayId: number,
  outputDir = "./screenshots",
): Promise<string> {
  return takeScreenshot({
    outputPath: path.join(outputDir, filename),
    displayId,
    playSound: false,
```

```
  });
}
```

  takeScreenshot   ,   .   .

---

outputPath	string	-
captureFullScreen	boolean	true
interactive	boolean	false
captureWindow	boolean	false
playSound	boolean	false
delay	number	0
format	'png' | 'jpg' | 'pdf' | 'tiff'	'png'
displayId	number	- ID

---

   interactive → captureWindow → captureFullScreen   .   true   ,
      .

  1 .

```
import {captureScreen} from "./utils/screencapture";

await captureScreen("basic-screenshot.png");
```

      takeScreenshot   .

```
import {type ScreenCaptureOptions, takeScreenshot} from "./utils/screencapture";

const options: ScreenCaptureOptions = {
  outputPath: "./screenshots/detailed-screenshot.jpg",
  format: "jpg",
  delay: 2,
  playSound: false,
};
await takeScreenshot(options);
```

PDF , . , `try` .

```
// PDF
await takeScreenshot({
  outputPath: "./screenshots/screenshot.pdf",
  format: "pdf",
});

//       -D 2
try {
  await captureFromDisplay("display-2-screenshot.png", 2);
} catch (error) {
  console.log("          ,      :", error);
}
```

, . .

```
console.log("             ...");
await captureInteractive("interactive-screenshot.png");

console.log("           ...");
await captureWindow("window-screenshot.png");
```

ts-node .

```
npx ts-node utils/screencapture-examples.ts
```

- **macOS** . `screencapture` macOS , Linux CI . CI E2E playwright `page.screenshot` , .
- . IDE . , .

- . , . `delay` .
- . , playwright . `headless: false` .

playwright , UI . DOM , OS , .

, macOS , , , . , playwright . , , .

E2E README AI . `screencapture` , 2 , .