

dtplib

Portable Printer interface

Remarks: js interface needs to be used with the dtplib plugin, otherwise the js interface can not print normally

- Compatible hardware environments: Windows, Kylin, UOS, Deepin, Ubuntu, etc.
- Supported software environments: Browser, NodeJS, VUE, React and so on.

Download

```
# npm download
npm install dtplib
# yarn download
yarn add dtplib
```

Usage

Install dtplib Print Assistant

1. Window version of the installation: in windows, the printer driver has been integrated dtplib Print Assistant, install the driver directly.
2. Linux desktop version (deb version):

```
# Installation of dtplib service
under linux
sudo apt update
sudo apt install libcurl4
sudo dpkg -i dtplib-2.1.xxxx-xxx.deb
# uninstallation of dtplib service under linux
sudo dpkg -r com.dothantech.dtplib
```

3. Linux server version (rpm): not supported yet

Using in Browser

1. Check if the local Print Assistant is available (the dtpweb print interface requires a local print assistant to print);
2. Introduce dtpweb.js file;
3. Obtain the interface instance via dtpweb.DTPWeb.getInstance();
4. Start label drawing operations;

eg:

```
<!-- Import the dtpweb.js file where appropriate -->
<script type="text/javascript" src="../../lib/dtpweb.js"></script>
<script>
  /**
   * @type {import("../lib/dtpweb").DTPWeb}
   */
  var api = dtpweb.DTPWeb.getInstance();

  window.onload = function () {
    // check if the interface is available
    api.checkPlugin(function (success) {if
      (!success) {
        api = undefined;alert("No Printer
        Plugin Detected! ");
      }
    });
  };
</script>
```

Use Interface in NodeJS/Vue

1. Import dtpweb modules

```
import { DTPWeb } from 'dtpweb'
```

2. Get interface instance api

```
mounted() {
  const api = DTPWeb.getInstance();
  api.checkPlugin((success) => {
    if (!success) {
      api = undefined;alert("No Print Service
      Detected");
    }
  })
}
```

Quickstart

Browser interface quickstart

```
// Interface initialization
<script type="text/javascript" src="../../lib_sync/dtpweb.js"></script>
<script>
var api = dtpweb.DTPWeb.getInstance();
window.onload = function() {
    api.checkPlugin((success) => { if
        (!success) {
            api = undefined;
            console.log("No DothanTech Printer Assistant Detected!");
        }
    });
}

async function printText() {
    const printerName = "DT60 Label Printer"; const
    labelWidth = 40;
    const labelHeight = 30;
    const text = "https://www.detonger.com" const
    fontHeight = 3;
    // Check if the interface is available
    if (!api) return;
    // 1. Connect the target printer before printing
    if (!(await api.openPrinter({printerName})))return;
    // 2. Create a print job in specific size
    await api.startJob({width: labelWidth, height: labelHeight});
    // 3. Draw Text on the label
    await api.drawText({text, width: labelWidth, height: labelHeight, fontHeight});
    // 4. End drawing operation and start printing
    await api.commitJob();
    // 5. Close the printer
    await api.closePrinter();
}
</script>
```

Quick start with the vue environment

- Install the dtpweb interface package

```
npm install dtpweb
```

Or

```
yarn add dtpweb
```

```
// Import Interface
```

```
import {DTPWeb} from "dtpweb"
```

```
// Initialization Interface
```

```
mounted() {
```

```
  this.api = DTPWeb.getInstance();this.api.checkPlugin((success)
```

```
  => {
```

```
    if (!success) {
```

```
      this.api = undefined;
```

```
      console.log("No Detected DothanTech Printer Helper!");
```

```
    }
```

```
  });
```

```
}
```

```
// Print test
```

```
methods: {
```

```
  async printQRCode() { const
```

```
    labelWidth = 30;
```

```
    const labelHeight = 30; const
```

```
    margin = 5;
```

```
    const qrcodeWidth = labelWidth - margin * 2; const
```

```
    printerName = "DT60 Label Printer";
```

```
    // Check if the interface is available
```

```
    if (!this.api) return;
```

```
    // 1. Connect the target printer before printing
```

```
    if (!(await this.api.closePrinter({printerName}))) return;
```

```
    // 2. Create a print job in specific size
```

```
    await this.api.startJob({});
```

```
    // 3. Drawing QR codes on print jobs
```

```
    await this.api.draw2DQRCode({x: margin, y: margin, width: qrcodeWidth});
```

```
    // 4. Ends drawing the print job and starts printing
```

```
    await this.api.commitJob();
```

```
    // Close the printer after printing
```

```
    await this.api.closePrinter();
```

```
  }
```

```
}
```

DTPWeb Interface Introduction

```
/**
 * PC JavaScript Version asynchronous wrapper for the LPAPI interface, based on the dtpweb print
 * interface.
 */
declare class DTPWeb {
  /**
   * Interface Initialization Configuration.
   */
  init(options?: LPA_InitOptions): void;
  /**
   * Check if the plugin is available.
   */
  checkPlugin(callback?: (success: boolean) => void): Promise<boolean>;
  /**
   * Check if the specified port is available.
   */
  checkPort(port?: number): Promise<boolean>;
  /**
   * Open the specified printer.
   */
  openPrinter(printer?: string | LPA_Device | undefined): Promise<boolean>;
  /**
   * Get the name of the connected printer.
   */
  getPrinterName(): Promise<string>;
  /**
   * Check if the printer is opened.
   */
  isPrinterOpened(): Promise<boolean>;
  /**
   * Close the printer.
   */
  closePrinter(): Promise<boolean>;
  /**
   * Get the printer list.
   */
  getPrinters(options: LPA_PrinterOptions): Promise<LPA_Device[]>;
  /**
   * Get print related parameters.
   */
  getParam(id: LPA_ParamID): Promise<number>;
  /**
   * Set Print Parameters;
   */
  setParam(options: LPA_PrintParamOptions): Promise<boolean>;
  /**
   * Get the Paper Type of a Connected Printer.
   */
}
```

```

getGapType(): Promise<LPA_GapType>;
/**
 * Set the Paper Type of a connected printer.
 */
setGapType(value: LPA_GapType): Promise<boolean>;
/**
 * Get the Print Darkness of a connected printer.
 */
getPrintDarkness(): Promise<number>;
/**
 * Set the Print Darkness of a connected printer.
 */
setPrintDarkness(value: number): Promise<boolean>;
/**
 * Get the Print Speed of a connected printer.
 */
getPrintSpeed(): Promise<number>;
/**
 * Set the Print Speed of a connected printer.
 */
setPrintSpeed(value: number): Promise<boolean>;
/**
 * Get the Print DPI. (Valid when printer is connected successfully) .
 */
getPrinterDPI(): Promise<number>;
/**
 * Start print jobs
 *
 * When starting a print job, this function automatically opens the first LPAPI supported printer installed on the
 * current system if no printer is connected.
 * All existing print data will be discarded if there are unprinted jobs.
 */
startJob(options: LPA_JobOptions): Promise<boolean>;
/**
 * Commit print job for real printing.
 */
commitJob(options?: PrintOptions): Promise<boolean>;
/**
 * Get the Page information for the print job that just completed.
 */
getPageInfo(): Promise<LPA_PageInfo>;
/**
 * Get the Page Image information for the print job that just completed.
 */
getPageImage(options: LPA_PageImageOptions): Promise<LPA_PageImage>;
/**
 * Start printing page
 */
startPage(): Promise<boolean>;
/**
 * End printing page
 */

```

```

endPage(): Promise<boolean>;
/**
 * Draw Text
 */
drawText(options: DrawTextOptions): Promise<boolean>;
/**
 * Print 1D Barcode
 */
draw1DBarcode(options: DrawBarcodeOptions): Promise<boolean>;
/**
 * Print 2D QR Code
 */
draw2DQRCode(options: DrawQrcodeOptions): Promise<boolean>;
/**
 * Print Pdf417 Code
 */
draw2DPdf417(options: DrawPdf417Options): Promise<boolean>;
/**
 * Draw Rectangular Frame
 *
 * @param {DrawRectOptions} options Related options for drawing rectangular frame
 */
drawRectangle(options: DrawRectOptions): Promise<boolean>;
/**
 * Draw Ellipse Frame
 */
drawEllipse(options: DrawRectOptions): Promise<boolean>;
/**
 * Draw Circle
 */
drawCircle(options: DrawCircleOptions): Promise<boolean>;
/**
 * Draw Line
 *
 * @param {DrawLineOptions} options Related options for drawing line
 */
drawLine(options: DrawLineOptions): Promise<boolean>;
/**
 * Print a specified URL image
 *
 * @param {DrawImageUrlOptions} options Related options for drawing URL image
 */
drawImage(options: DrawImageUrlOptions): Promise<boolean>;
/**
 * Draw Image Objects
 *
 * @param {DrawImageDataOptions} options Related options for drawing Image Objects
 */
drawImageD(options: DrawImageDataOptions): Promise<boolean>;
/**
 * Print a specified bitmap object

```

```
*  
* @param {PrintImageOptions} options Related options for printing image  
*/  
    printImage(options: PrintImageOptions): Promise<boolean>;  
}
```

Revision History

v2.1.20221212

- The relevant code upgrade of the underlying dtpweb service Bluetooth connection.
- Realized the function of json data printing.

v2.1.5

- Synchronized update of relevant interface documentation according to the latest version of the interface.

v2.1.4

- Interface optimization

v2.1.3

- Optimize and improve the interface documentation.

v2.1.2

- Solved the problem that base64 images could not be printed when calling the interface to print in nodejs.

v2.1.1

- Improvement of the LAN printing function;
- Improvement and update of interface usage and annotations.

v2.1.0

- Version 2.1 released.