@push.rocks/smartreport

A library for creating and displaying reports on the web.

- readme.md for @push.rocks/smartreport

readme.md for @push.rocks/smartreport

create reports and display them on the web

Install

To install `@push.rocks/smartreport`, you need to run the following command using npm (Node Package Manager):

```
npm install @push.rocks/smartreport --save
```

Ensure you have Node.js and npm installed on your machine before you install the package.

Usage

This documentation provides a comprehensive guide on how to utilize `@push.rocks/smartreport` to create and display reports on the web using TypeScript. Whether you're creating a simple project report or an elaborate web analytics display, `@push.rocks/smartreport` offers the tools required to craft, manipulate, and showcase your data elegantly.

Prerequisites

Before you begin, ensure that the following are installed:

- Node.js (10.x.x or higher)
- TypeScript (3.x or higher)

Getting Started

First, import `Smartreport`, `ReportSection`, and `ReportItem` from `@push.rocks/smartreport`:

```
import { Smartreport, ReportSection, ReportItem } from '@push.rocks/smartreport';
```

Setting up Your Report

To set up a new report:

```
const myReport = new Smartreport();
```

Adding Report Sections

A report is comprised of various sections that can be individually added to the report instance. Each `ReportSection` can have a heading or name to distinguish what it represents.

```
const salesReportSection = new ReportSection('Sales Data');
myReport.addReportSection(salesReportSection);
```

Adding Items to a Section

Within each `ReportSection`, you can add multiple `ReportItem`s which represent the data or content of the section:

```
const salesItem1 = new ReportItem(/* Include your data here */);
// Add data and configurations to salesItem1 as required
salesReportSection.addReportItem(salesItem1);
```

Repeat the process to add more sections and items as needed by your report.

Customizing `ReportItem`

The `ReportItem` class can be extended or used to encapsulate data points, charts, or any form of data representation. Custom attributes and methods can be added to represent and process your data effectively. Currently, customization requires direct interaction with the class:

```
class CustomReportItem extends ReportItem {
  constructor(public data: any) {
    super();
    // Add custom initialization and methods here
  }
```

```
}

const customItem = new CustomReportItem({metric: 'visitors', count: 1000});
salesReportSection.addReportItem(customItem);
```

This example demonstrates how to subclass `ReportItem` for custom data encapsulation. Implement data processing and presentation logic within your custom classes as needed.

Displaying the Report

To display your report on the web, you will need to convert it into a format suitable for web consumption, such as HTML or JSON. This library focuses on the creation and structural organization of report data, so transforming the report into a web-friendly format and displaying it would require additional steps, potentially using other libraries or custom code to parse and render the data.

For example, to generate HTML:

```
// This function is purely illustrative
function reportToHtml(report: Smartreport): string {
  let htmlOutput = `<h1>My Report</h1>`;
  report.reportSections.forEach(section => {
    htmlOutput += `<h2>${section.name}</h2>`;
    section.reportItems.forEach(item => {
      // Implement custom rendering based on the item's data and type
      htmlOutput += `<p>Item: ${/* item data representation */}</p>`;
    });
  });
  return htmlOutput;
}

const myReportHtml = reportToHtml(myReport);
// Display `myReportHtml` in your web application
```

Note: The function `reportToHtml` is for illustration purposes and must be implemented based on your project's requirements.

Conclusion

`@push.rocks/smartreport` provides the foundational tools to construct structured, data-driven reports. While creating and organizing report data is straightforward with `Smartreport`,

transforming and displaying this data on the web requires additional handling suited to your application's architecture and presentation needs.

Remember, the Transformation and presentation layer for displaying the report on the web is up to you to implement. Use your preferred libraries and frameworks to bring your report data to life in the user interface.

License and Legal Information

Trademarks

Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.