### What is Sharepoint?

A Web-based collaborative platform that integrates with MS Office. Briefly, it is a document management and storage system(企业级协作和内容管理)

* Sharepoint Server
* Sharepoint Online(趋势！ 多租户的平台)

One of the most popular web parts in SharePoint Online is the Script Editor

SharePoint的网 站归为两类

* 一类是要进行协作的团队网站，所有成员都可以参与内容创作、协作等；
* 一 类是通信网站（或者准确地说是沟通网站，英文是Communication Site），它一般用来在企 业内部发布内容，大部分用户只是查看

SharePoint Online 不支持服务器场和沙箱解决方案，但是继续支持用户直接在浏览器中定制 和“开发”页面（可以写少量的脚本，改样式），以及通过SharePoint Designer进行定制（网页 的高级定制，工作流定制等），与此同时，它还支持下面两种开发模式

1. SharePoint Add-in开发，允许开发人员独立开发一个Web应用，然后用iframe的方式嵌入到SharePoint的页面或者网站中去.。
2. SharePoint Framework 开发，允许开发人员使用全新的客户端开发手段，定制Web Part 和Extension。

SharePoint Add-in的开发（在2013年提出来）可用于SharePoint Server，也可以用于SharePoint Online.

Sharepoint Framework(16年提出的)

经过一段时间的发展，要有全新的一套框架来重 新定义SharePoint的开发。具体而言，希望能用更加原生的Web开发技术来实.

通过编程访问SharePoint的资源，例如列表，文档库等, 需要Sharepoint Online 的REST API (<https://msdn.microsoft.com/en-us/library/office/jj860569.aspx>), 现在也支持在Microsoft Graph中直接访问(但有限，<https://developer.microsoft.com/zh-cn/graph/docs/api-reference/v1.0/resources/sharepoint>)

SharePoint Framework

The SharePoint Framework (SPFx) is a page and web part model that provides full support for client-side SharePoint development。

SharePoint Framework 适用于 SharePoint Online，并即将可用于本地 SharePoint (SharePoint 2016 Feature Pack 2)。

Sharepoint Framework 是2016年发布的新开发框架。原因就是因为SharePoint本身是在不断发展，需要有全新的一套框架来重新定义SharePoint 的开发——具体而言，希望能用更加原生的Web开发技术来实现，并且与SharePoint有更加自然的融合。

特性：

1. **在当前用户的上下文和浏览器的连接中运行**。没有像SharePoint Add-in那样使用IFrame，也不是将JavaScript 直接嵌入到页面中（安全风险较高，也可能受制于用户浏览器的设置而失效）。
2. 控件直接在页面 DOM 中呈现。
3. 控件支持响应式呈现，以适应不同尺寸的界面。
4. 允许开发人员更好地访问生命周期 - 其中包括呈现 - 加载、序列化和反序列化、配置更改等等
5. 它未指定框架。可以使用喜欢的任何 JavaScript 框架：React、Handlebars、Knockout、Angular 等
6. 工具链基于 npm、ypeScript、Yeoman、webpack 和 gulp 等常见开放源代码客户端开发工具。
7. 提供可靠的性能表现，相比较SharePoint Add-in来说有了极大的提升。
8. 最终用户可以在所有网站上使用租户管理员（或其代理）批准的 SPFx 客户端解决方案，其中包括自助式团队、组或个人网站。
9. Web 部件可添加到经典页面和新式页面，同时支持SharePoint Online和SharePoint Server。

适用范围：

* 客户端Web部件，你可以用Javascript实现所有的界面，并将其应用到任何的SharePoint页面中去。
* 扩展程序（Extensions），包括修改页面逻辑的 ApplicationCustomizers，为字段提供定制的 FieldCustomizers ，还有为列表或者文档库添加自定义菜单和命令的 CommandSets

Visual Studio Code轻量级的、跨平台的工具来进行SPFx开发，而且因为SharePointFramework的框架无关性，可以使用Javascript框架（例如React、Handlebars、Knockout、Angular等）开发。如果你对C#很熟悉，你的经验可以得到复用，TypeScript is a nice transition into the JavaScript world. 而且，Visual Studio Code内置了对Typescript的支持。

Development Environment

SharePoint client-side development tools include a Yeoman generator for creating new web parts.

Install NodeJS

Install a Code editor: Visual Studio Code

Install Yeoman and gulp

配置环境变量，给出Yeoman的命令yo 所在的位置例如：

C:\Users\song\_\AppData\Roaming\npm

Install Yeoman SharePoint geneator

参考：<https://docs.microsoft.com/en-us/sharepoint/dev/spfx/set-up-your-development-environment>

问题请参考：<https://blog.csdn.net/changjn/article/details/42677941>

<https://www.tuicool.com/articles/VJJZza#c-6389>

Web **parts 开发**

<https://Sppnp.sharepoint.com/sites/Group/_layouts/15/workbench.aspx>

SharePoint client-side solution is **HTML/TypeScript based**,

1. Run Yeoman SharePoint Generator, 建立一个project.

Yo @microsoft/sharepoint

Preview 效果build and run it on **a local web server**：

1. The client-side toolchain uses HTTPS endpoint by default.

**install the developer certificate** and preview your web part:

gulp trust-dev-cert

**rebuild project :**

gulp serve

  这时create a local, node-based HTTPS server on localhost:4321

(gulp serve 这个命令在本地启动一个Workbench文件，用来调试刚才创建的这个WebPart)

支持在本地直接进行调试，不要求你安装SharePoint Server，也不需要你真的拥有SharePoint

Online的环境

**Gulp**: A task runner to handle build process tasks such as

* Bundling and minifying JavaScript and CSS files.
* Running tools to call the bundling and minification tasks before each build.
* Compiling SASS files to CSS.
* Compiling TypeScript files to JavaScript.

**在命令行中>code . 打开VS Code 的WebParts**

**Visual Studio Code** provides built-in support for gulp and other task runners. Select Ctrl+Shift+B on Windows or Cmd+Shift+B on Mac to debug and preview your web part.

**SharePoint Workbench** is a developer design surface that enables you to quickly preview and test web parts without deploying them in SharePoint.

假设我已经有了一个SharePoint Online的网站，我想在它里面直接去调试这个WebPart，我可以简单的这样做：

1. 打开任何一个SharePoint Online站点，将其地址复制下来，例如 https://microsoftapc.sharepoint.com/teams/Samplesiteforares
2. 在上面的地址后面追加一段地址 /\_layouts/15/workbench.aspx，在浏览器中访问这个地址后，你将看到一个跟刚才那个本地调试界面很类似的页面

部署SPFx WebPart

1. Any client-side web part should extend the **BaseClientSideWebPart** class

因为，**BaseClientSideWebPart** implements the minimal functionality that is required to build a web part. It provides many parameters to validate and access read-only properties such as :

* **displayMode**
* web part properties
* web part context
* web part **instanceId**,
* the web part **domElement**
* and much more.

To learn more about how to work with the property pane and property pane field types, see [Make your SharePoint client-side web part configurable](https://docs.microsoft.com/en-us/sharepoint/dev/spfx/web-parts/basics/integrate-with-property-pane).

### 在property pane中加入新的功能

第一步：import

@microsoft/sp-webpart-base

加入properties to the property pane: a checkbox

第二步：maps the fields to typed objects

因为增加了新的properties，我们需要在**IHelloWorldWebPartProps interface 中加入这些新的properties的属性。**

**第三步：**

**在getPropertyPaneConfiguration** method中adds the new property pane fields and maps them to their respective typed objects.

第四步：

在render() method中，加入properties to the web part properties, you can now access the properties in the same way you accessed the **description** property.

第五步：

在manifest.json中设置默认值

.manifest.json 中defines the web part metadata such as version, id, display name, icon, and description

Connect Web part to Sharepoint

**通常在Sharepoint中我们可以获得如下数据：**

* Web title
* Web absolute URL
* Web server-relative URL
* User sign-in name

**用this.context.pageContext 可以调用以上资源**

**本地测试：**

**Get the page context**

**在render中加入**

<p class="${ styles.description }">Loading from ${escape(this.context.pageContext.web.title)}</p>

**保存 + gulp serve**

**在<https://localhost:4321/temp/workbench.html>**

**在Sharepoint上测试：**

**保证本地gulp serve运行**

**https://mygame.sharepoint.com/\_layouts/workbench.aspx**

[**用LucasSong@mygame.onmicrosoft.com**](mailto:用LucasSong@mygame.onmicrosoft.com)**登陆**

**添加web part时，可以看到hello world**

**(虽然是online 测试, 但本地gulp serve向应)**

案例 List model

**本地测试：**

需要建一个mock store

import { ISPList } from './HelloWorldWebPart';

**因为在HelloWorldWebPart中有很多exports, 所有必须特指其中的 ISPList，用{ }**

**而且不需要给出后缀 “.ts”,在HelloWorldWebPart。因为它是default module**

案例中给出的代码It exports the **MockHttpClient** class as a default module so that it can be imported in other files.

**线上测试：**

需要用class **spHttpClient** to execute REST API requests against SharePoint.

所以需要import 从@microsoft/sp-http下

import {

SPHttpClient,

SPHttpClientResponse

} from '@microsoft/sp-http';

this.context.spHttpClient.get

发送GET 请求

**EnvironmentType 用来识别web parts运行环境**