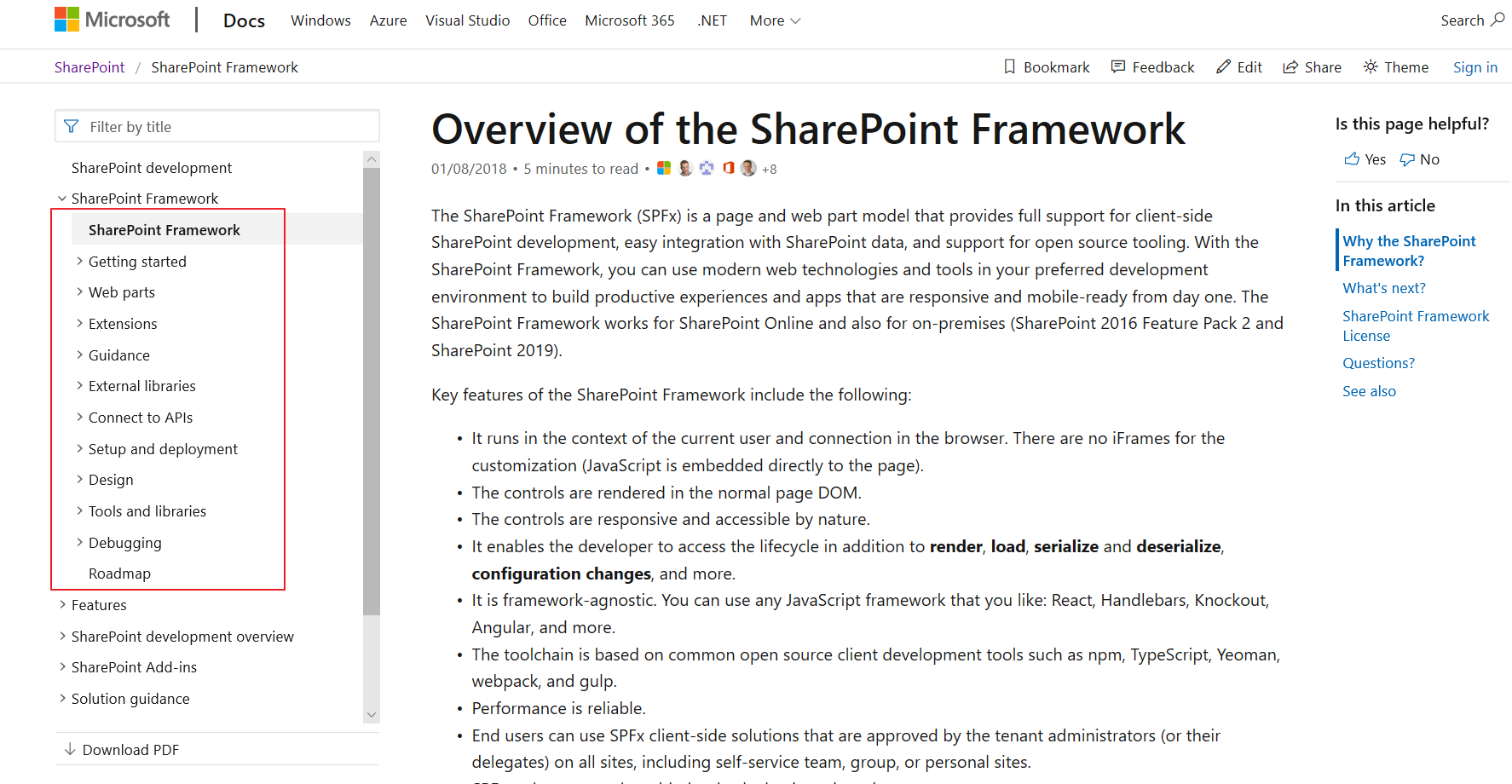
SPFx(SharePoint Framework)

SharePoint 2019 부터는 SharePoint Online의 Modern UI를 사용합니다. 기존의 팜 솔루션을 통한 화면 변경 및 핸들링은 여전히 가능하지만, Modern UI 상태에서는 마스터페이지나 Context Menu 같은 부분은 팜 솔루션으로는 변경이 되지 않습니다.

SPFx를 통해 Modern UI에 접근할 수 있습니다. SPFx의 특징은 다음과 같습니다.

* 현재 사용자의 컨텍스트에서 실행되고 브라우저에서 연결됩니다. 사용자 화를위한 iFrame이 없습니다 (자바 스크립트는 페이지에 직접 임베드 됨).
* 컨트롤은 일반 페이지 DOM에서 렌더링됩니다.
* 컨트롤은 기본적으로 반응 형이며 액세스 가능합니다.
* 이를 통해 개발자는 **렌더링**, **로드**, **직렬화** 및 **역 직렬화**, **구성 변경** 등을 비롯하여 수명주기에 액세스 할 수 있습니다.
* 프레임 워크에 구애받지 않습니다. React, Handlebars, Knockout, Angular 등 원하는 JavaScript 프레임 워크를 사용할 수 있습니다.
* 툴체인은 npm, TypeScript, Yeoman, webpack 및 gulp와 같은 일반적인 오픈 소스 클라이언트 개발 도구를 기반으로 합니다.
* 성능이 안정적입니다.
* 최종 사용자는 셀프 서비스 팀, 그룹 또는 개인 사이트를 포함한 모든 사이트에서 테넌트 관리자 (또는 대리인)가 승인 한 SPFx 클라이언트 쪽 솔루션을 사용할 수 있습니다.
* SPFx 웹 파트는 클래식 페이지와 최신 페이지에 모두 추가할 수 있습니다.

참고: <https://docs.microsoft.com/en-us/sharepoint/dev/spfx/sharepoint-framework-overview>  


# SPFx 환경설정

## Office 365 Developer Program

Office 365 tenant 가 없을 경우, Office 365 Developer Program을 통해 90일의 Office 365를 테스트할 수 있습니다.

참고: <https://developer.microsoft.com/en-us/office/dev-program>

참고: <https://docs.microsoft.com/ko-kr/office/developer-program/office-365-developer-program>

## Office 365 tenant 설정하기

SPFx로 빌드된 앱을 사용하기 이해서는 app catalog Site가 생성되어 있어야 합니다.

1. 브라우저에 다음 URL을 입력하여 **SharePoint 관리 센터** 로 이동하십시오. tenant 접두사를 Office 365 tenant 접두사로 바꿉니다. (ex: https://yourtenantprefix-admin.sharepoint.com)
2. 왼쪽 사이드 바에서 **추가 기능을** 선택하십시오.
3. **앱** 섹션을 찾아 **열기를** 선택하십시오.
4. **앱** 페이지에서 **앱 카탈로그를** 선택하십시오.
5. **확인**을 선택하여 새 앱 카탈로그 사이트를 만듭니다.
6. 다음 페이지에서 다음 세부 사항을 입력하십시오.
   * **제목**: **앱 카탈로그를** 입력하십시오.
   * **웹 사이트 주소 *접미사***: 앱 카탈로그의 선호 접미사를 입력하십시오. 예를 들어, **apps**입니다.
   * **관리자**: 사용자 이름을 입력한 다음 확인 버튼을 선택하여 사용자 이름을 확인하십시오.
7. 앱 카탈로그 사이트를 만들려면 **확인**을 선택하십시오.

참고: <https://docs.microsoft.com/en-us/sharepoint/dev/spfx/set-up-your-developer-tenant#create-app-catalog-site>

## SPFx 개발도구 설치하기

Visual Studio 또는 자체 사용자 지정 개발 환경을 사용하여 SharePoint Framework 솔루션을 구축할 수 있습니다. Mac, PC 또는 Linux를 사용할 수 있습니다.

* + 1. NodeJS 설치  
       <https://nodejs.org/dist/latest-v10.x/node-v10.19.0-x64.msi>
    2. Code Editor 설치  
       Visual Studio Code, Atom, Webstorm과 같은 Code editor 가 있으나, 여기서는 Visual Studio Code를 설치하는 것으로 진행합니다.  
       <https://code.visualstudio.com/>
    3. Yeoman and gulp 설치  
       SPFx는 Yeoman 과 gulp를 이용하여 기본 템플릿을 구성하고 배포합니다.  
       다음 명령을 입력하여 Yeoman 및 gulp를 설치합니다.  
       Command 창에서 진행합니다.

|  |
| --- |
| npm install -g yo gulp |

* + 1. Yeoman Sharepoint generator 설치  
       Command 창에서 진행합니다.

|  |
| --- |
| npm install -g @microsoft/generator-sharepoint |

* + 1. Modern web browser설치  
       개발 테스트를 위해서는 Edge, Chrome, Firefox 와 같은 브라우저가 필요합니다. 개발 테스트를 위한 Local workbench는 IE에서 동작하지 않습니다.
    2. Self-signed developer 인증  
       개발 환경에서 사용자 지정 솔루션을 테스트할 때 사용되는 SharePoint Framework의 로컬 웹 서버는 기본적으로 HTTPS를 사용합니다. 이것은 개발 자체 서명 SSL 인증서를 사용하여 구현됩니다. 자체 서명 SSL 인증서는 개발자 환경에서 신뢰하지 않습니다. 먼저 인증서를 신뢰하도록 개발 환경을 구성해야 합니다.  
       Command 창에서 진행합니다.

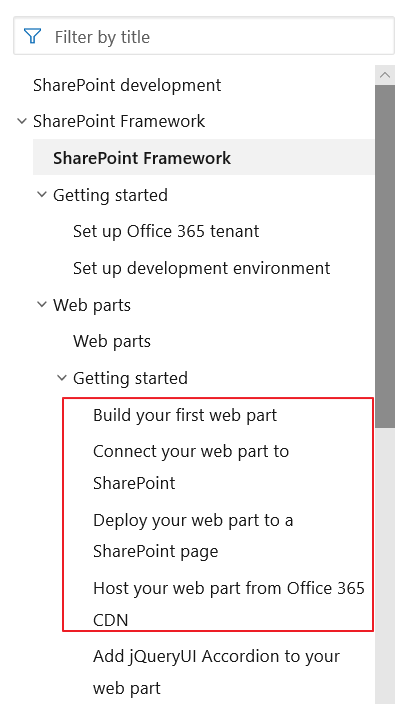
|  |
| --- |
| gulp trust-dev-cert |

SPFx 환경설정에 대한 내용은 아래 참고 주소의 내용을 정리하였습니다.

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

.

# SPFx Tutorial

SPFx를 익혀보는데 좋은 예제는 Web Parts를 만드는 과정을 따라해 보는 것을 추천합니다. 표시한 과정을 모두 진행하면, 프로젝트 생성, 빌드, SharePoint Online에 배포하는 과정을 익힐 수 있습니다.  


참고: <https://docs.microsoft.com/en-us/sharepoint/dev/spfx/web-parts/get-started/build-a-hello-world-web-part>

<https://docs.microsoft.com/en-us/sharepoint/dev/spfx/web-parts/get-started/connect-to-sharepoint>

<https://docs.microsoft.com/en-us/sharepoint/dev/spfx/web-parts/get-started/serve-your-web-part-in-a-sharepoint-page>

<https://docs.microsoft.com/en-us/sharepoint/dev/spfx/web-parts/get-started/hosting-webpart-from-office-365-cdn>

# 배포 시 유의사항

SharePoint online에 배포된 앱을 다시 배포할 경우, 버전정보를 변경하지 않으면 배포된 사이트에서는 업데이트 사항이 반영되지 않으므로, 추가 배포할 경우에는 버전을 증가시켜 배포하여야 합니다.

## config> package-solution.json 수정

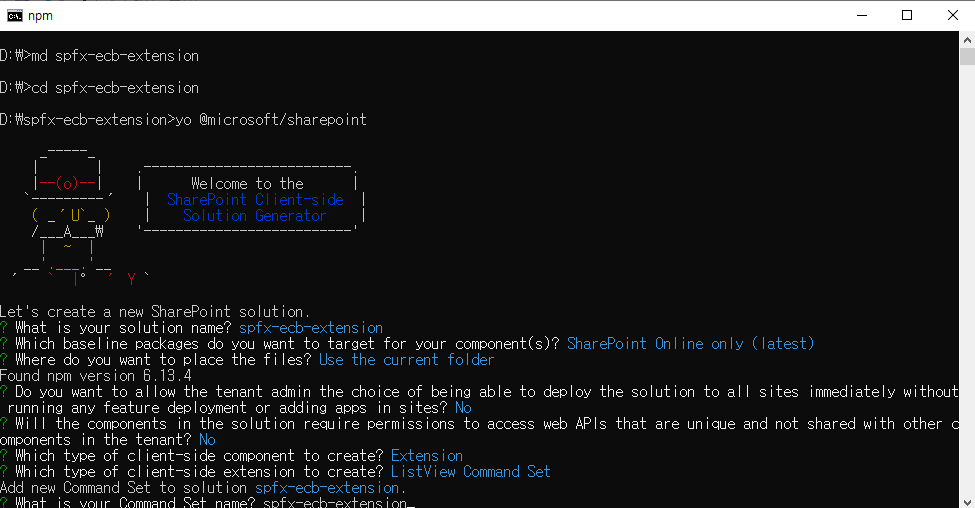
## AppCatalogs에 빌드한 앱 업로드

## 배포된 사이트(ex: 팀사이트)에서 앱 세부정보 선택

## 앱의 새 버전 확인 및 설치 클릭하여 새로운 버전으로 앱 업데이트

# Context Menu 예제

SPFx Tutorial을 진행한 후 이 예제를 진행하는 것을 권장 드립니다. 이 예제는 **Migrating from Edit Control Block (ECB) menu item to SharePoint Framework Extensions** (<https://docs.microsoft.com/en-us/sharepoint/dev/spfx/extensions/guidance/migrate-from-ecb-to-spfx-extensions>)를 참조한 예제입니다. 이 예제는 CustomAction을 추가하여 줍니다.

* 1. 솔루션 생성  
     솔루션 생성은 Command를 이용하여 생성합니다.  
     

|  |
| --- |
| D:\>md spfx-ecb-extension  D:\>cd spfx-ecb-extension  D:\spfx-ecb-extension>yo @microsoft/sharepoint  ? What is your solution name? spfx-ecb-extension  ? Which baseline packages do you want to target for your component(s)? SharePoint Online only (latest)  ? Where do you want to place the files? Use the current folder  ? Do you want to allow the tenant admin the choice of being able to deploy the solution to all sites immediately without  running any feature deployment or adding apps in sites? No  ? Will the components in the solution require permissions to access web APIs that are unique and not shared with other components in the tenant? No  ? Which type of client-side component to create? Extension  ? Which type of client-side extension to create? ListView Command Set  Add new Command Set to solution spfx-ecb-extension.  ? What is your Command Set name? spfx-ecb-extension  ? What is your Command Set description? spfx-ecb-extension description |

* 1. jQuery 사용을 위해 다음의 명령어를 Command 에서 실행합니다.

|  |
| --- |
| npm install jquery --save  npm install --save @types/jquery |

* 1. 솔루션을 다 받고 나면 code . 을 입력하여 Visual Studio Code실행합니다.
  2. Location 옵션에 따라, ContextMenu가 보여지는 위치를 선택할 수 있습니다.  
     **ClientSideExtension.ListViewCommandSet.ContextMenu:** ContextMenu만 설정  
     **ClientSideExtension.ListViewCommandSet.CommandBar:** CommandBar 만 설정  
     **ClientSideExtension.ListViewCommandSet:** ContextMenu 및 CommandBar 둘다 설정
  3. sharepoint\assets\ClientSideInstance.xml 파일변경

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Elements xmlns="http://schemas.microsoft.com/sharepoint/">      <ClientSideComponentInstance          Title="SpfxEcbExtensionT"          Location="ClientSideExtension.ListViewCommandSet"          ListTemplateId="101"          Properties="{&quot;sampleTextOne&quot;:&quot;One item is selected in the list.&quot;}"          ComponentId="49cfbd6f-c2e9-4072-8121-896a88f37bdc" />  </Elements> |

* 1. sharepoint\assets\elements.xml 파일변경

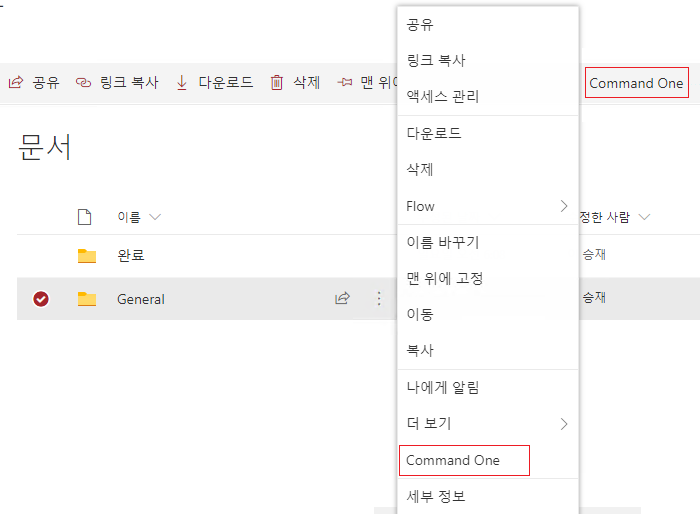
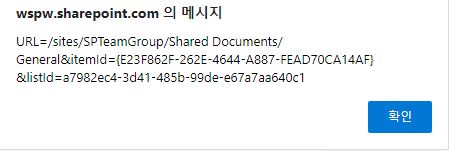
|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Elements xmlns="http://schemas.microsoft.com/sharepoint/">      <CustomAction          Title="SpfxEcbExtensionT"          RegistrationId="101"          RegistrationType="List"          Location="ClientSideExtension.ListViewCommandSet"          ClientSideComponentId="49cfbd6f-c2e9-4072-8121-896a88f37bdc"          ClientSideComponentProperties="{&quot;sampleTextOne&quot;:&quot;One item is selected in the list.&quot;}">      </CustomAction>  </Elements> |

* 1. src\extensions\spfxEcbExtensionT\SpfxEcbExtensionTCommandSet.manifest.json 파일변경

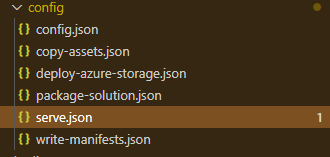
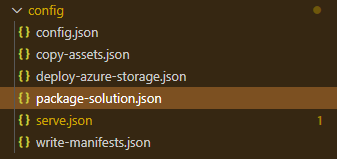
|  |
| --- |
| {    "$schema": "https://developer.microsoft.com/json-schemas/spfx/command-set-extension-manifest.schema.json",    "id": "49cfbd6f-c2e9-4072-8121-896a88f37bdc",    "alias": "SpfxEcbExtensionTCommandSet",    "componentType": "Extension",    "extensionType": "ListViewCommandSet",    // The "\*" signifies that the version should be taken from the package.json    "version": "\*",    "manifestVersion": 2,    // If true, the component can only be installed on sites where Custom Script is allowed.    // Components that allow authors to embed arbitrary script code should set this to true.    // https://support.office.com/en-us/article/Turn-scripting-capabilities-on-or-off-1f2c515f-5d7e-448a-9fd7-835da935584f    "requiresCustomScript": false,    "items": {      "COMMAND\_1": {        "title": { "default": "Command One" },        "iconImageUrl": "icons/request.png",        "type": "command"      }      }  } |

* 1. src\extensions\spfxEcbExtensionT\SpfxEcbExtensionTCommandSet.ts 파일변경

|  |
| --- |
| import { Guid } from '@microsoft/sp-core-library';  import { override } from '@microsoft/decorators';  import { Log } from '@microsoft/sp-core-library';  import {    BaseListViewCommandSet,    Command,    IListViewCommandSetListViewUpdatedParameters,    IListViewCommandSetExecuteEventParameters  } from '@microsoft/sp-listview-extensibility';  import { Dialog } from '@microsoft/sp-dialog';  //  // jquery 사용을 위해서는 Command 에서 다음의 명령을 실행  // npm install jquery --save  // npm install --save @types/jquery  //  import \* as $ from "jquery";  import \* as strings from 'SpfxEcbExtensionTCommandSetStrings';  /\*\*   \* If your command set uses the ClientSideComponentProperties JSON input,   \* it will be deserialized into the BaseExtension.properties object.   \* You can define an interface to describe it.   \*/  export interface ICustomEcbCommandSetProperties {    // This is an example; replace with your own properties    Command1\_Text: string;  }  const LOG\_SOURCE: string = 'CustomEcbCommandSet';  export default class CustomEcbCommandSet extends BaseListViewCommandSet<ICustomEcbCommandSetProperties> {    @override    public onInit(): Promise<void> {      Log.info(LOG\_SOURCE, 'Initialized CustomEcbCommandSet');      return Promise.resolve();    }    @override    public onListViewUpdated(event: IListViewCommandSetListViewUpdatedParameters): void {      const compareOneCommand: Command = this.tryGetCommand('COMMAND\_1');      if (compareOneCommand) {        // This command should be hidden unless exactly one row is selected.        compareOneCommand.visible = event.selectedRows.length === 1;      }    }    @override    public onExecute(event: IListViewCommandSetExecuteEventParameters): void {      switch (event.itemId) {        case 'COMMAND\_1':          const itemUrl: string = event.selectedRows[0].getValueByName("FileRef");          const itemId: number = event.selectedRows[0].getValueByName("UniqueId");          const listId: Guid = this.context.pageContext.list.id;  $(document).ready(function (){alert(`URL=${itemUrl}&itemId=${itemId}&listId=${listId}`)});          //Dialog.alert(`URL=${itemUrl}&itemId=${itemId}&listId=${listId}`);          break;        default:          throw new Error('Unknown command');      }    }  } |

* 1. 빌드  
     gulp bundle --ship
  2. 패키지  
     gulp package-solution --ship
  3. 패키징된 파일은 sharepoint\solution 경로 하위에 생성됩니다.  
     spfx-ecb-extension.sppkg
  4. AppCatalogs에 패키징된 sppkg 파일을 업로드 한 후, 테스트할 사이트로 가서 앱 설치를 합니다.
  5. 항목 하나를 선택하면 추가된 항목(Command One)을 확인할 수 있습니다. 상단부분이 **CommandBar**이고 Context 가 **ContextMenu**입니다.  
     
  6. Command One을 클릭하면 항목의 정보를 확인할 수 있습니다.  
     

# 참고사항

* Config 정보  
  개발테스트 시 사용하는 gulp serve 의 설정값은 serve.json 을 읽습니다.  
    
  배포파일 패키징 시 사용하는 gulp package-solution 의 설정값은 package-solution.json 을 읽습니다.  
  
* SharePoint Framework Client-Side Extension Samples  
  SPFx 샘플들이 업로드 되어 있는 사이트입니다.   
  <https://sharepoint.github.io/sp-dev-fx-extensions/samples/type/>