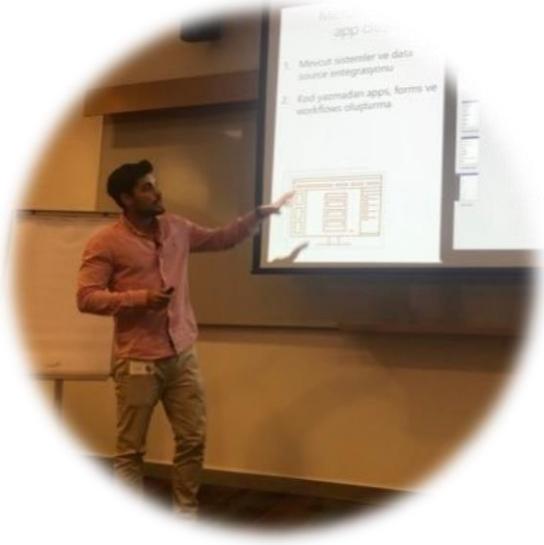


SharePoint Framework Development

TypeScript, React JS ve PnP JS Kullanarak SharePoint Framework Development

SERDAR KETENCI



Senior Software Engineer

Hakkında

Serdar Ketenci, SharePoint ve Office 365 'te çözümler geliştirmekte olan, ASP.NET Core, Node JS, Django alt yapısını kullanarak Web API, MVC alanlarında da çalışmaları bulunan kıdemli bir bilgisayar mühendisidir. Özel ilgili alanı ise Python ile Data Science & AI üzerindedir.

SharePoint çözümler geliştirmekle birlikte 2017 ve 2018 yıllarında İstanbulda düzenlenen SharePoint Saturday etkinliklerinde hem organizatör hem de konuşmacı olarak yer almıştır. 2019 yılında üçüncüsü düzenlenecek olan SharePoint Saturday etkinliğinde de yine organizatör ve konuşmacı olarak yer alacaktır.

İçindekiler

Genel Bakış	6
SharePoint Framework SharePoint Ortam Kurulumu.....	7
App Catalog Oluşturma (SharePoint Online).....	8
App Catalog Oluşturma (SharePoint On-Premises)	9
Developer Site Oluşturma (SharePoint Online).....	11
Developer Site Oluşturma (SharePoint On-Premises)	12
SharePoint Framework Development Ortam Kurulumu.....	13
Node JS Kurulumu	14
Yeoman ve Gulp Kurulumu.....	18
Yeoman SharePoint Generator Kurulumu.....	18
Code Editor Kurulumu	19
Opsiyonel Araçlar (Tools).....	19
SharePoint Framework (SPFx) WebPart Genel Bakış	19
Neden TypeScript ve React?	19
Şimdi Yeni Bir WebPart Oluşturalım.....	20
Yeni WebPart Oluşturmak İçin.....	20
Client-Side WebPart Önizleme	22
Client-Side WebPart Önizleme ve Test Etme	25
SharePoint Framework (SPFx) WebPart Proje Yapısı	27
Klasör Yapısı	28
Proje İçerisindeki Önemli Dosyalar	30
WebPart Class	30
WebPart Render Method	31
WebPart Property Pane Konfigürasyonu	33
WebPart Manifest	38
SharePoint İçerisinde WebPart Önizleme	39
SharePoint Framework Projesine İkinci WebPart Nasıl Eklenir?	40
SharePoint Framework WebPart SharePoint Entegrasyonu 1	41
Loading Indicator.....	41

Error Indicator	42
Lodash Utility Library.....	43
Page Display Modes	44
Page Context	47
Environment Type	48
Logging.....	50
SPComponentLoader.....	50
SharePoint Framework WebPart SharePoint Entegrasyonu 2	51
Model Tanımlama	51
Servis Tanımlama	52
SharePoint Framework WebPart Güncelleme	56
Localization Güncelleme.....	58
React Component Güncelleme	59
React Component İçerisine SharePoint Framework WebPart Context Aktarılmalı Mıdır?	64
Kolaylık Ama Gerçekten Sağlıklı Bir Yöntem Mi?.....	64
Peki Neden Kötü Fikir?	65
Peki Ne Yapmalıyız?.....	66
SharePoint Framework WebPart Office Fabric UI React Kullanımı	66
Hedef	67
Demo	67
SharePoint Framework WebPart PnP Kütüphanesi	70
@pnp/sp Kurulumu	70
@pnp/sp import.....	70
SharePoint Framework Extensions	72
SharePoint Framework Extensions Application Customizer	74
Proje Oluşturma	75
Application Customizer Örnek Kod	77
Application Customizer Debug	78
SharePoint Framework Extensions Field Customizer	81
Proje Oluşturma	81

Field Customizer Debug.....	83
SharePoint Framework Extensions Command Set.....	85
Proje Oluşturma	86
ListView Command Set Debug.....	88
serve.config İle İlgili Detaylar	89
SharePoint Framework Deployment Office 365 CDN	90
Office 365 CDN Aktifleştirme	90
Solution Settings	91
SharePoint Framework Deploy Süreci	92
SharePoint Framework Uygulama Yükleme	93
SharePoint Framework Deployment SharePoint Library	96
CDN Yerine Kullanacağınız Doküman Kütüphanesi Oluşturma	96
SharePoint Library İçin Solution Package Konfigürasyonu	97
CDN Path Güncelleme	97
SharePoint Framework Deploy Süreci	98
SharePoint Framework Uygulama Yükleme	100
SharePoint Framework Deployment Azure CDN	102
Azure Storage Account Konfigürasyonu.....	102
BLOB Container Oluşturma.....	103
Storage Account Access Key	104
CDN Profile ve Endpoint	105
Solution Package Konfigürasyonu.....	106
Azure Storage Account Konfigürasyonu.....	107
CDN Path Güncelleme	108
SharePoint Framework Deploy Süreci	108
SharePoint Framework Uygulama Yükleme	110

Genel Bakış

SharePoint Framework (SPFx) client-side SharePoint development yapmamıza olanak sağlayan bir platformdur. SPFx, SharePoint ile hızlı bir şekilde entegre olabilmekte ve open source library konusunda esnek development yapabilmeniz için çalışma ortamınızda SharePoint yüklü olmasına ihtiyaç duymamaktadır. Kısacası; SharePoint developerları bir çok külfetten kurtaran bir alt yapı sunmaktadır.

SPFx alt yapısını, *SharePoint Online*, *SharePoint 2019* ve *SharePoint 2016 Feature Pack 2* versiyonlarında kullanabilirsiniz.

SPFx 'in temel özellikleri;

- ✓ SharePoint App olarak çalışabilmektedir. Yani önceden geliştirmekte olduğumuz SharePoint Hosted App gibi Iframe olarak çalışmamaktadır.
- ✓ Browser DOM içerisinde render edilebilmektedir. Dolayısıyla müdahale etmeyi kolaylaştırmaktadır.
- ✓ Client-Side development 'a imkan vermesinden dolayı JavaScript Framework ile çalışma yapılmaktadır. (React, Handlebars, Knockout, Angular, Vue gibi.)
- ✓ Open Source Client Development tool 'u kullanabilirsiniz. (npm, TypeScript, Yeoman, Webpack, Gulp gibi.)
- ✓ SharePoint App mantığında çalışmasından dolayı geliştirilen çözüm son kullanıcılara, tenant admin yöneticileri (veya delegeleri) tarafından dağıtıldıktan (onaylandıktan) sonra kullanılabilir.
- ✓ Classic veya Modern sayfalar ile uyumlu çalışabilmektedir.

Yukarıda gördüğünüz gibi bir çok özelliği bulunmaktadır. Peki bu duruma nasıl geldik? İsterseniz şimdi de biraz SharePoint development yöntemlerindeki değişime bir göz atalım.

SharePoint 2001 'de lanse edildiğinden beri bir çok development yöntemi ile geliştirmeler yaptık. SharePoint 2007 ile *CAML Query*, *SharePoint Object Model*, *XML*, *XSLT*, *Silverlight*, *Web User Control*, *Visual WebPart* gibi kavramlarla karşılaştık ve programlama dili olarak VB ve C# kullanarak geliştirmeler yaptık.

Sonrasında SharePoint 2010 çıktı ve *Ajax* desteği ile SharePoint 2007'nin desteklemediği browserları destekledi. (Sadece multiple browser support değil, aslında External Content Type, Term Store, Visual Studio 2010 uyumlu hale gelerek template kullanabildik.) Yine C# kullanarak development sürecimizi devam ettirdik.

SharePoint 2013 ile birlikte *JSLink* (Client-Side Rendering), *SharePoint App Model* gibi kavramlarla tanıştık. *SharePoint-Hosted App* ile tamamen javascript yazarak uygulama oluşturmaya, *Provider-Hosted App* ile MVC projelerimizi SharePoint alt yapısıyla entegre edebilmemize olanak sağlandı. Fakat SharePoint App ile geliştirme yapmamızın bazı dezavantajları da oluştu; Client-Side WebPart ’ı site içerisinde yüklediğimizde IFrame olarak çalışması gibi.

Aynı zamanda diğer bir geliştirme yöntemi olarak *Rest API* üzerinden Client-Side development yapabildik. Kisaca özetlemek gerekirse, tüm bu süreçler devam ederken SharePoint Online ile tanıştık. Microsoft cloud üzerinde C# ile development ’ı kendi sunucusu üzerinde ölçekleyememesinden dolayı SharePoint Online tarafında C# ’a desteğini kaldırıldı. Sonrasında geliştirmiş olduğumuz modülleri her iki platform üzerinde desteklemek adına (On-Premises & Online) *Rest API* üzerinden development yaptık (Javascript Injection).

Bu development yöntemi aslında *JavaScript Injection* olmasından dolayı, sayfa içerisinde Script Editör kullanarak ya da MasterPage / Page Layout içerisinde ekleyerek kodumuzu çağırmak zorunda bırakmasından dolayı WebPart oluşturma konusunda tam anlamıyla dezavantajlar yarattı.

Bu gibi sebeplerden dolayı *SharePoint App* ve *JavaScript Injection* özellikleri kullanılarak, günümüz dünyasına ayak uydurmak adına SharePoint Framework SPFx ’i geliştirdi.

SharePoint Framework SharePoint Ortam Kurulumu

SharePoint App ’ı sitemiz üzerinde kullanabilmek için kendi ortamımızı hazırlamamız gerekmektedir.

Eğer herhangi bir ortamınız mevcut değil ise, [Office 365 Developer Program](#) ’ına kayıt olup, Office 365 developer aboneliği edinebilirsiniz.

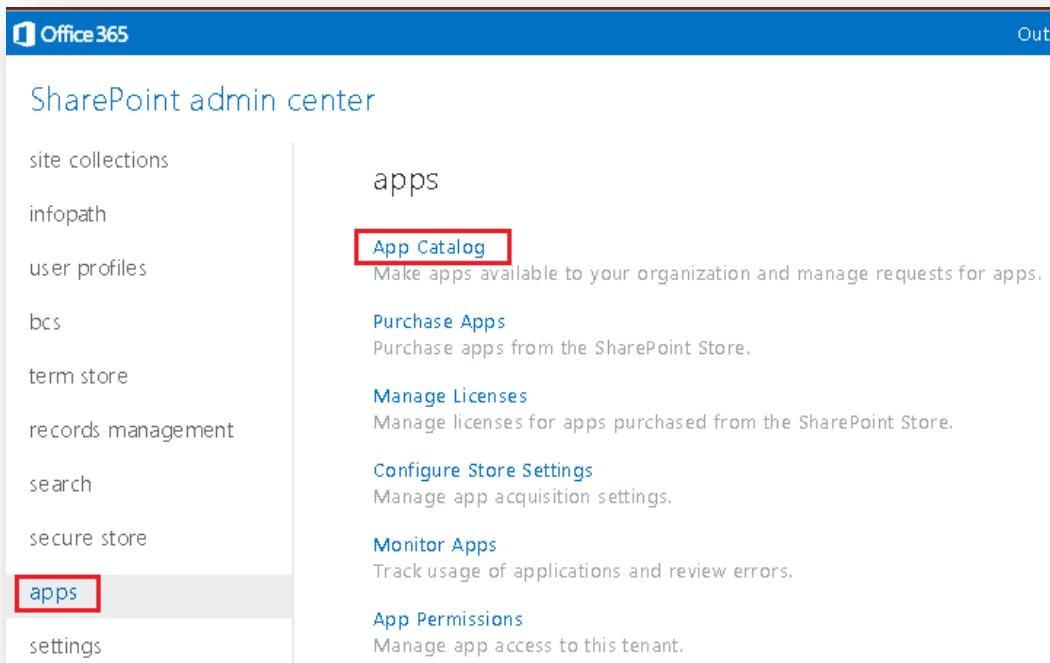
SPFx ile geliştirmiş olduğunuz uygulamaları yüklemek ve dağıtmak için App Catalog ihtiyacınız bulunmaktadır. Şimdi SharePoint Online ve On-Premises ortamları için neler yapmamız gerekiyor onlara göz atalım. Eğer App Catalog mevcut ise bu adımı geçebilirsiniz.

App Catalog Oluşturma (SharePoint Online)

- ✓ Aşağıdaki linkte yer alan “*SharePoint Admin Center*” adresine giriniz.
“yourtenantprefix” yazan kısmı kendi prefixiniz ile değiştirmeyi unutmayın.

<https://yourtenantprefix-admin.sharepoint.com>

- ✓ Sol menüde yer alan “*Apps (Uygulamalar)*” seçeneğine tıklayınız.

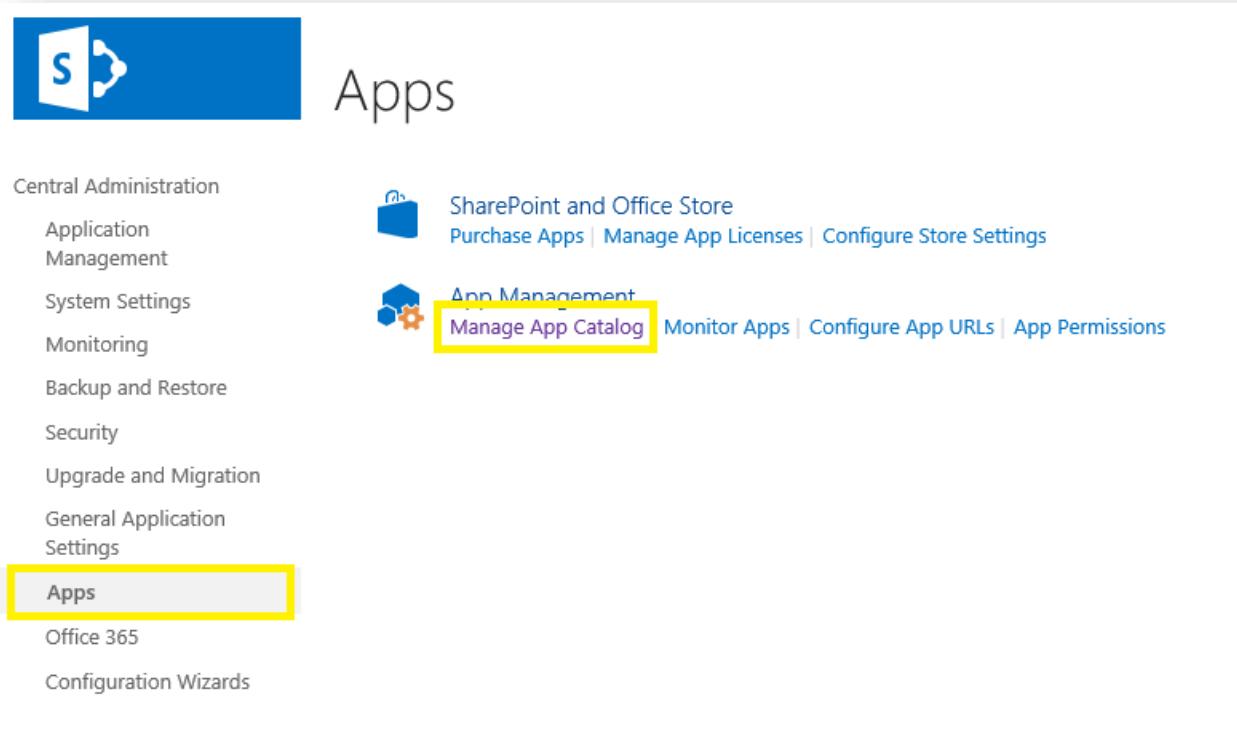


- ✓ Karşınıza gelen sayfa üzerinde yer alan “*App Catalog (Uygulama Kataloğu)*” seçeneğine tıklayınız.
- ✓ App Catalog oluşturmak için “OK (Tamam)” butonuna tıklayınız.
- ✓ Bir sonraki sayfada aşağıdaki bilgileri doldurunuz (Site collection oluşturmak için gerekli bilgilerdir.)
 - **Title (Başlık):** Oluşturulacak site adıdır.
 - **Web site address suffix (Web Site Adresi):** Oluşturulacak site collection url adıdır.
 - **Administrator (Yönetici):** Oluşturulacak site collection admin bilgisidir.
- ✓ Yukarıdaki bilgileri girdikten sonra “OK (Tamam)” butonuna tıklayınız.

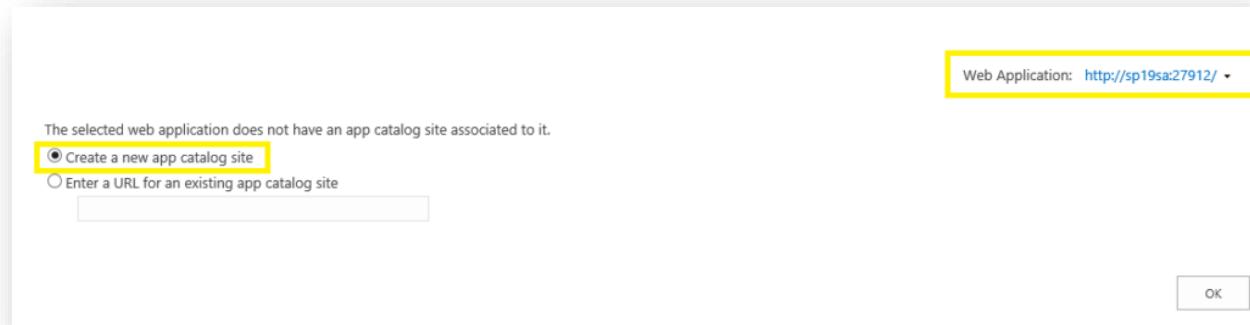
App Catalog ile ilgili tüm işlemlerimiz bu kadardır. Site collection ‘ın oluşması biraz zaman alabilir (1-2 dk). Takip etmek için “SharePoint Admin Center” içerisinde sol menüde yer alan “Site Collections (Site Koleksiyonu)” alanını görüntüleyebilirsiniz.

App Catalog Oluşturma (SharePoint On-Premises)

- ✓ SharePoint Central Administration (SharePoint Yönetim Paneli) açınız.
- ✓ Açılan sayfa üzerinde “Apps (Uygulamalar)” içerisindeki “Manage App Catalog (Uygulama Kataloğunu Yönet)” seçeneğine tıklayınız.



- ✓ Karşınıza gelen sayfa üzerinde ilgili Web Application ‘ı seçip, “Create a new app catalog site” seçeneğini işaretleyerek (default olarak işaretli gelmektedir.) “Ok (Tamam)” butonuna tıklayınız.



- ✓ Bir sonraki sayfada aşağıdaki bilgileri (Site collection oluşturmak için gerekli bilgilerdir) doldurmanız gerekmektedir.
 - **Title (Başlık):** Oluşturulacak site adıdır.
 - **Web site address suffix (Web Site Adresi):** Oluşturulacak site collection url adıdır.
 - **Administrator (Yönetici):** Oluşturulacak site collection admin bilgisidir.

Web Application: <http://sp19sa:27912/>

Title:
App Catalog

Description:

URL:
<http://sp19sa:27912/sites/AppCatalog>

English ▾

User name:
Serdar Ketenci

- ✓ Yukarıdaki bilgilerinizi girdikten sonra “OK (Tamam)” butonuna tıklayınız.

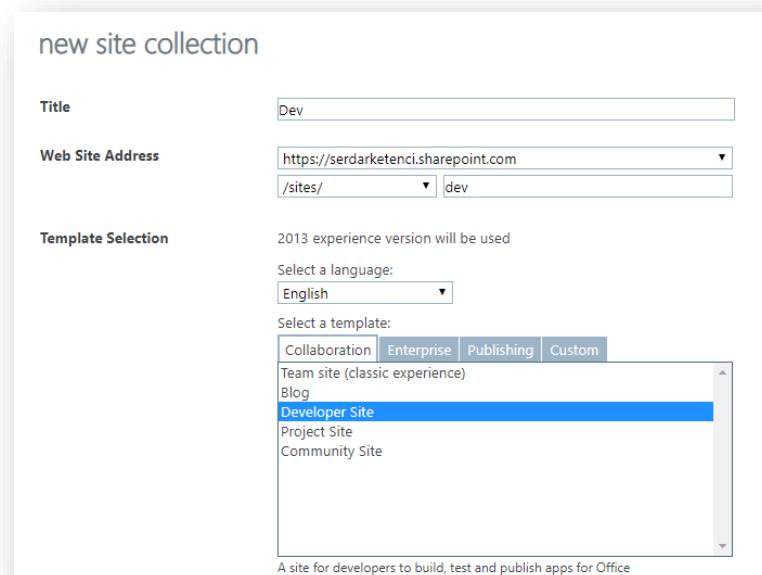
Developer Site Oluşturma (SharePoint Online)

Yaptığınız çalışmaları test etmek için site collection ve site ihtiyacınız bulunmaktadır. Herhangi bir site template açıp, onun üzerinde geliştirmelerinizi yapabiliriz (Workbench ve testler için). Fakat best practices olarak “Developer Site” üzerinden gidilmesi App ’i deploy yaparken direkt site üzerinde denemek yerine, öncesinde developer site üzerine yükleyerek test işlemlerini tamamladıktan sonra ilgili site üzerine yüklemeniz daha sağlıklı olacaktır.

- ✓ Aşağıdaki linkte yer alan “SharePoint Admin Center” adresine giriniz.
“yourtenantprefix” yazan kısmı kendi prefixiniz ile değiştirmeyi unutmayın.

<https://yourtenantprefix-admin.sharepoint.com>

- ✓ Karşınıza gelen sayfa üzerinde (Site Collections) “New (Yeni)” → “Private Site Collection (Özel Site Koleksiyonu)” seçeneğine tıklayınız.
- ✓ Bir sonraki sayfada aşağıdaki bilgileri (Site collection oluşturmak için gerekli bilgilerdir) doldurmanız gerekmektedir.
 - **Title (Başlık):** Oluşturulacak site adıdır.
 - **Web Site Address Suffix (Web Site Adresi):** Oluşturulacak site collection url adıdır.
 - **Template Selection (Şablon Seçimi):** Developer Site (Geliştirici Sitesi) seçiniz.
 - **Administrator (Yönetici):** Oluşturulacak site collection admin bilgisidir.



- ✓ Yukarıdaki bilgileri girdikten sonra “OK (Tamam)” butonuna tıklayınız.

Developer Site Oluşturma (SharePoint On-Premises)

- ✓ SharePoint Central Administration (SharePoint Yönetim Paneli) açınız.
- ✓ “Application Management (Uygulama Yönetimi)” → “Create site collections (Site koleksiyonu oluştur)” linkine tıklayınız.



- ✓ Bir sonraki sayfada aşağıdaki bilgileri (Site collection oluşturmak için gereklili bilgilerdir) doldurmanız gerekmektedir.
 - **Title (Başlık):** Oluşturulacak site adıdır.
 - **Web Site Address Suffix (Web Site Adresi):** Oluşturulacak site collection url adıdır.
 - **Template Selection (Şablon Seçimi):** Developer Site (Geliştirici Sitesi) seçiniz.
 - **Administrator (Yönetici):** Oluşturulacak site collection admin bilgisidir.

Web Application: <http://sp19sa:27912/>

Title: Dev

Description:

URL: http://sp19sa:27912/sites/ Dev

Select a language: English

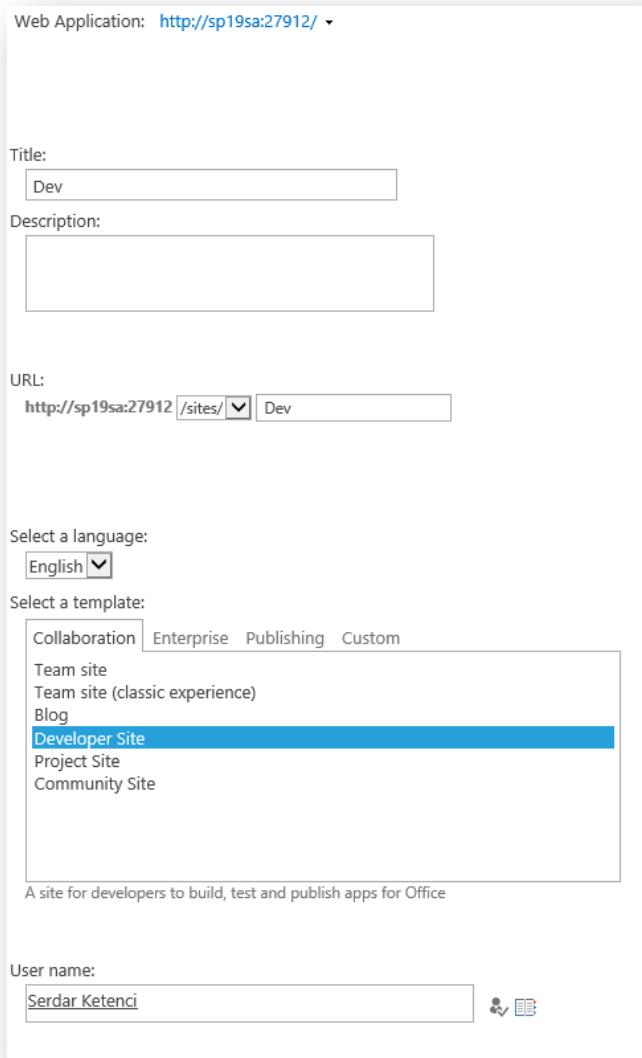
Select a template:

Collaboration Enterprise Publishing Custom

Team site
Team site (classic experience)
Blog
Developer Site
Project Site
Community Site

A site for developers to build, test and publish apps for Office

User name: Serdar Ketenci



- ✓ Yukarıdaki bilgileri girdikten sonra “OK (Tamam)” butonuna tıklayınız.

SharePoint Framework Development Ortam Kurulumu

SharePoint Framework Development ortamını oluşturmanız için aşağıdaki bileşenlerin bilgisayarlarınızda yüklü olması gerekmektedir.

- ✓ Node JS
- ✓ Yeoman ve Gulp
- ✓ Yeoman SharePoint Generator
- ✓ Code Editor (Visual Studio / Webstorm vs.)
- ✓ Postman ve Fiddler (Opsiyonel)

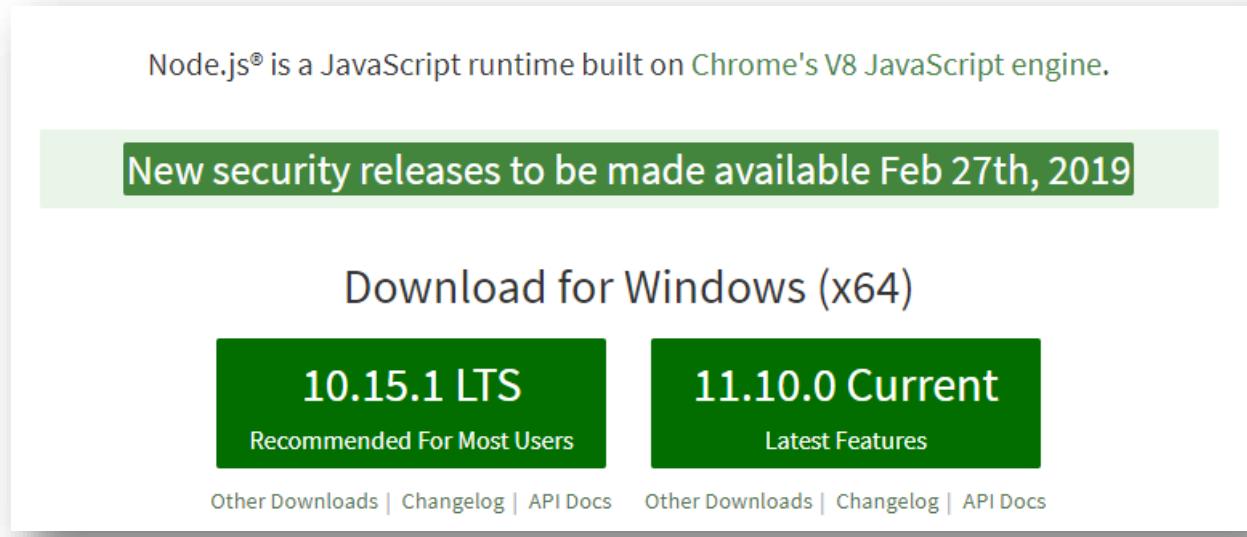
Node JS Kurulumu



Joyent tarafından 2009 yılında geliştirilmeye başlanmış bir Javascript Runtime platformudur. V8 JavaScript motoru üzerinde çalışmaktadır. V8 Google tarafından geliştirilen, Chrome web browser 'ının da üzerinde çalıştığı C, C++ ve JavaScript dilleri ile kodlanan bir enginedir. Tek amacı JavaScript kodunu makine koduna çevirmektir. Node Package Manager (npm) aynı zamanda dünyadaki en büyük open source kütüphane ekosistemidir. Modülleri paketlemek ve deploy etmek için Gulp ve Yeoman kullanarak ilerleyebilirsiniz.

İlk adım olarak Node JS LTS kurulumunu gerçekleştirelim;

- ✓ [Linke](#) tıklayarak Node JS web sitesini açınız.

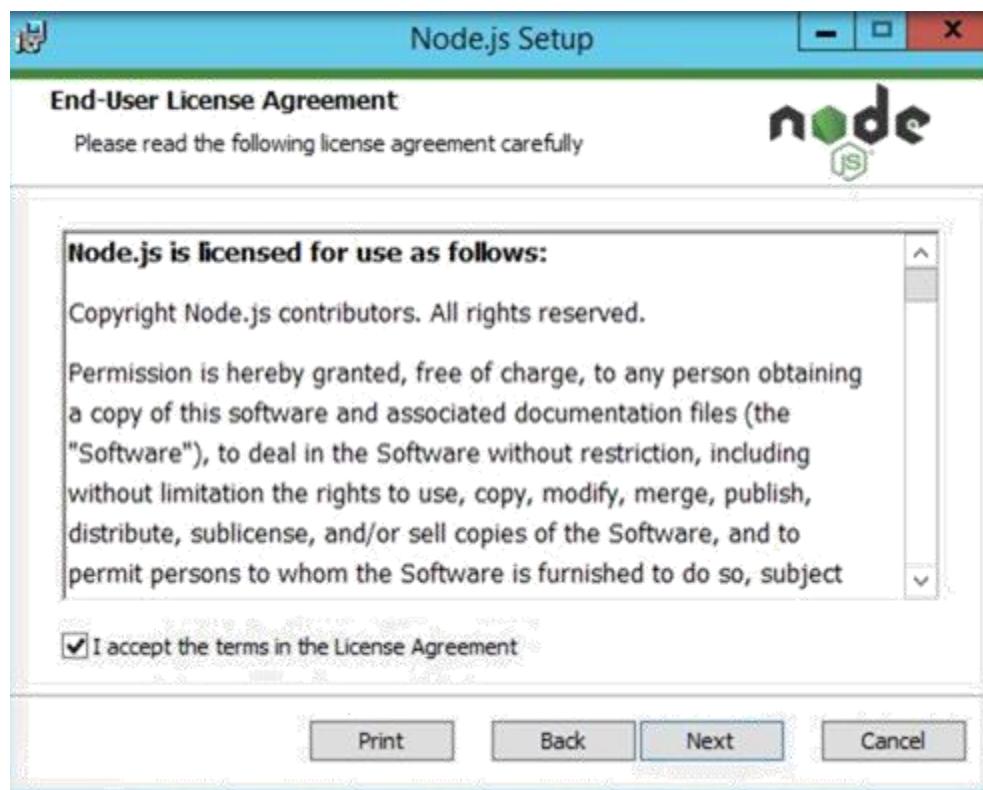


The screenshot shows the official Node.js website's download section. At the top, there is a statement: "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." Below this, a green banner announces "New security releases to be made available Feb 27th, 2019". Two large green buttons are prominently displayed: "10.15.1 LTS" (labeled "Recommended For Most Users") and "11.10.0 Current" (labeled "Latest Features"). At the bottom of the page, there are links for "Other Downloads | Changelog | API Docs" for both the LTS and Current versions.

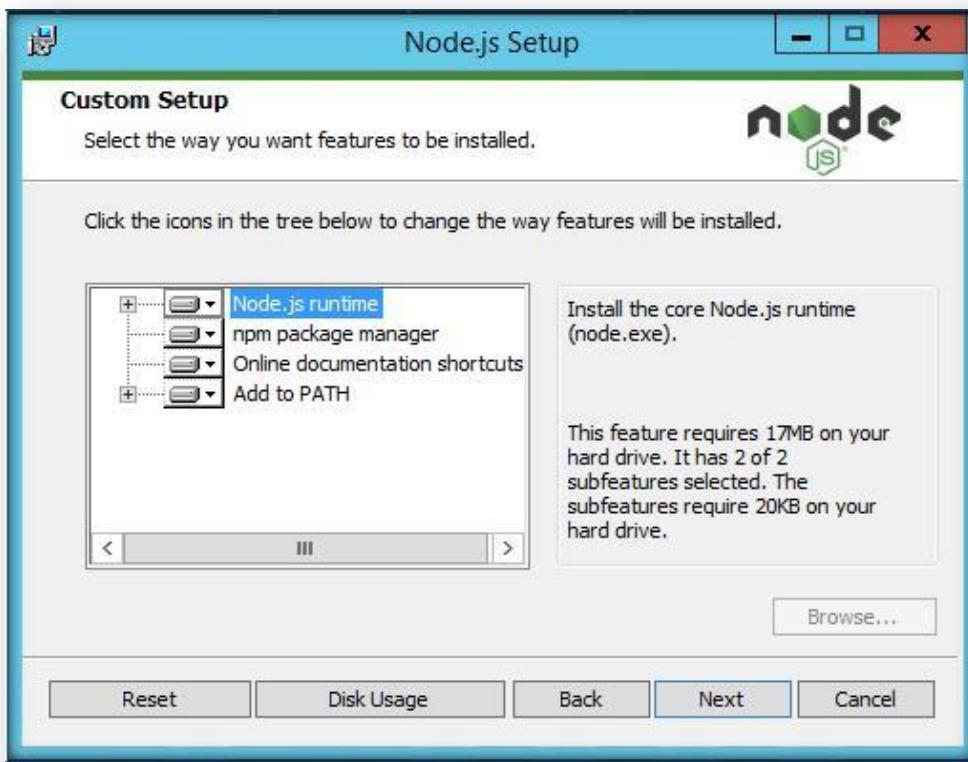
- ✓ LTS sürümünü indirdikten sonra, dosyayı çalıştırınız ve "Next" butonuna tıklayınız.



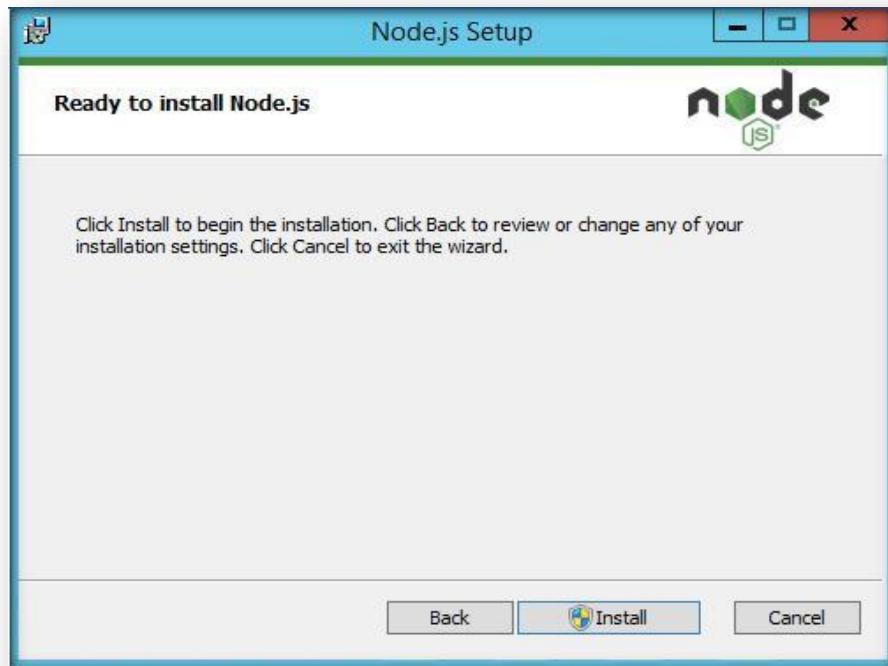
- ✓ "I accept the terms in the License Agreement" checkbox'ını işaretleyerek "Next" butonuna tıklayınız.



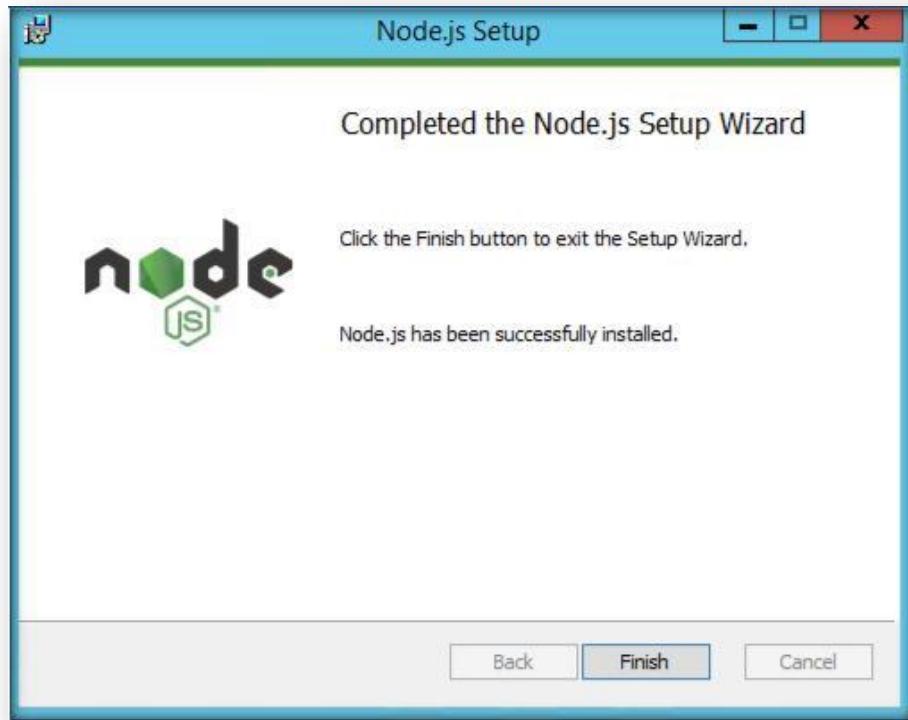
- ✓ "Next" butonuna tıklayınız.



✓ “Install” butonuna tıklayınız.



- ✓ Yükleme işlemi tamamlandığında “Finish” butonuna tıklayarak işlemleri sonlandırınız.



- ✓ Tüm kurulum işlemlerinizi yaptıktan sonra bilgisayarınızı yeniden başlatınız.
- ✓ Bilgisayarınızı yeniden başlattıktan sonra Node JS command prompt açtığınızda aşağıdaki gibi bir görüntü oluşacaktır.

A screenshot of a command prompt window titled "Node.js command prompt". The window shows the message "Your environment has been set up for using Node.js 10.10.0 (x64) and npm." and the command prompt line "C:\Users\sketenci>".

- ✓ “**npm -v**” komutu ile versiyonu görüntüleyebilirsiniz.

A screenshot of a command prompt window titled "Select Node.js command prompt". The window shows the message "Your environment has been set up for using Node.js 10.10.0 (x64) and npm.", the command "C:\Users\sketenci>npm -v", and the output "6.4.1".

Yeoman ve Gulp Kurulumu



Yeoman web uygulaması geliştirirken kullanılan kütüphane (bootstrap, jquery vb.), araç (gulp, bower vb.) ve dosya yapısı benzerlik göstermektedir. Bu nedenle her proje için aynı yapıyı kurmak zaman alacağından, Microsoft yeni framework üzerinde Yeoman 'ı kullanarak bize hazır bir yapı (scaffolding) sunmaktadır.

Gulp kısaca bir “javascript görev çalıştırıcısıdır (javascript task runner)”. Gulp bize aşağıdaki avantajları sağlamaktadır,

- ✓ Script ve Style dosyalarınızı küçültür,
- ✓ Dosyalarınızı birleştirir,
- ✓ Ön Belleği Temizler,
- ✓ Test ve optimizasyon kullanmak için (SPFx ile oluşturduğumuz projelerimizi paketler, SharePoint 'e yükler vs.)

Lokalinize global olarak Yeoman ve Gulp yüklemek için aşağıdaki kod bloğunu çalıştırınız.

```
npm install -g yo gulp
```

Yeoman SharePoint Generator Kurulumu

Yeoman SharePoint web part generator hızlıca SharePoint Client-Side development için gerekli ortamı hazırlamanıza yardımcı olmaktadır.

SharePoint Framework (SPFx), Yeoman generator global olarak kurmak için aşağıdaki kod bloğunu çalıştırınız.

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

SharePoint Framework (SPFx), Yeoman generator farklı sürümlerini yüklemek ve farklı projeleriniz arasında geçiş yapmak isterseniz, lokal olarak kurmanız gerekmektedir. Lokal olarak yüklemek için aşağıdaki kod bloğunu çalıştırın.

```
npm install @microsoft/generator-sharepoint --save-dev
```

Code Editor Kurulumu

Client-Side development yapabilmeniz için herhangi bir code editor ya da IDE kullanabilirsiniz.

- ✓ Visual Studio Code
- ✓ Atom
- ✓ Webstorm

Opsiyonel Araçlar (Tools)

- ✓ Fiddler
- ✓ Postman
- ✓ Cmder for Windows
- ✓ Oh My Zsh for Mac
- ✓ Git source control tools

SharePoint Framework (SPFx) WebPart Genel Bakış

SharePoint Framework Angular, Vue, React, Handlebars, Knockout gibi bir çok JavaScript Framework ve library ile entegre çalışabilmektedir.

Neden TypeScript ve React?

TypeScript, object oriented yazmaya aşina olanlar yani statik ve type safe olmasından dolayı tercih edilmektedir. Projelere daha standart getirilmesini sağlamaktadır. (Bu demek değildir ki JavaScript ile OOP yapamıyoruz, EcmaScript 6'da class özelliği eklendi ve prototype-based inheritance, super calls, instance, static method, constructors destek vermeye başladı.)

React, 2013 yılında Facebook tarafından release olmasından itibaren virtual dom, esnek bir kütüphane olması ile popüleritesini artırdı ve facebook, AirBnb, Uber, Netflix gibi firmalar tarafından tercih edilmesi de en güçlü yanlarından birisi oldu.

Şimdi Yeni Bir WebPart Oluşturalım

Yeni WebPart Oluşturmak İçin

- 1) İlk olarak çalıştığınız lokasyon üzerinde aşağıdaki kodu çalıştırarak yeni klasör oluşturunuz. (**md** bulunduğu konum üzerinde klasör oluşturmanızı sağlar.)

```
md helloworld-webpart
```

- 2) Oluşturduğunuz proje dizinine gitmek için aşağıdaki kodu çalıştırınız. (**cd** komutu belirttiğiniz dizine gitmenizi sağlar.)

```
cd helloworld-webpart
```

- 3) Yeoman SharePoint Generator kullanarak yeni bir solution oluşturunuz.

```
yo @microsoft/sharepoint
```

```
C:\Examples>md helloworld-webpart  
C:\Examples>cd helloworld-webpart  
C:\Examples\helloworld-webpart>yo @microsoft/sharepoint
```

- 4) Sizden solution oluşturmak için aşağıdaki bilgileri isteyecektir;
 - “What is your solution name?” default olarak “helloworld-webpart” yazmaktadır, “enter” tuşuna basıp kullanabilir ya da farklı bir solution name belirtebilirsiniz.
 - “Which baseline packages do you want to target for your component(s)? (Use arrow keys)” son versiyon üzerinden devam ediniz.
 - “Where do you want to place the files? (Use arrow keys)” oluşturulacak dosyaların hangi kısma atılacağını belirtiniz. “Use the current folder” deyip, var olan klasör üzerine oluşturmayı tercih ediniz.

- “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? (y/N)” her siteye uygulamanızı eklemek yerine, tüm sitelerde otomatik olarak dağıtılmış bir şekilde kullanmak istiyorsanız, “y” deyip, “enter” tuşuna basınız. Default olarak “No” yazmaktadır, herhangi bir şey yazmadan “enter” dediğinizde “No” olarak değeri atayacaktır.
- “Which type of client-side component to create? (Use arrow keys)” client -side component olarak seçiminizi “WebPart” olarak yapınız.

```

Let's create a new SharePoint solution.
? What is your solution name? helloworld-webpart
? 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
Found npm version 6.4.1
? 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? Yes
? Which type of client-side component to create? WebPart

```

5) Sonraki parametreler WebPart için gerekli olan bilgilerdir;

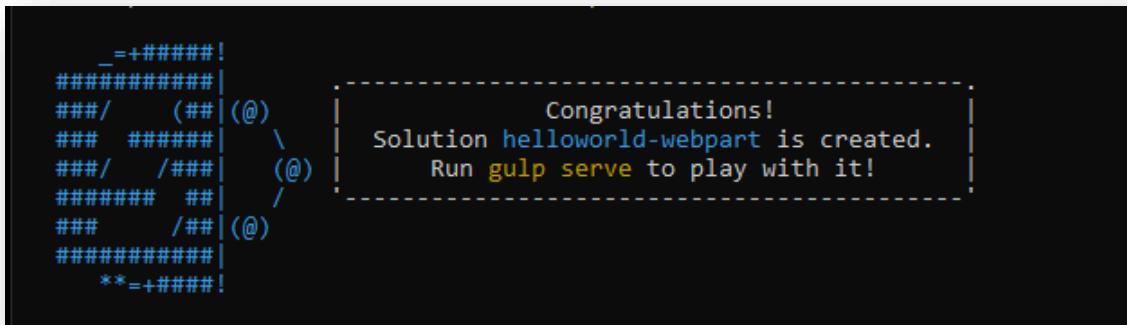
- “What is your Web part name?” WebPart ismini yazınız.
- “What is your Web part description?” WebPart için açıklama giriniz.
- “Which framework would you like to use?” kullanmak istediğiniz framework seçiniz, “React” deyip ilerleyiniz.

```

Add new Web part to solution helloworld-webpart.
? What is your Web part name? HelloWorld
? What is your Web part description? HelloWorld description
? Which framework would you like to use? React

```

Tüm parametrelerimizi girdikten sonra Yeoman gerekli olan dependencies yükler ve gerekli alt yapıyı hazırlar. Bu işlem 1-2 dakika sürebilir.



Client-Side WebPart Önizleme

Önizleme yapabilmek için local web server üzerinden build ve run etmektedir. Default olarak HTTPS protokolü üzerinden haberleşmektektir. Web server ile ilgili config dosyaları proje içerisinde “config” → “serve.json” içerisinde yer almaktadır. İhtiyacınıza göre konfigüre edebilirsiniz.

```
1  {
2   "$schema": "https://developer.microsoft.com/json-schemas/core-build/serve.schema.json",
3   "port": 4321,
4   "https": true,
5   "initialPage": "https://localhost:5432/workbench",
6   "api": [
7     {
8       "port": 5432,
9       "entryPath": "node_modules/@microsoft/sp-webpart-workbench/lib/api/"
10    }
11 ]
```

Developer sertifikasının SADECE geliştirme ortamınıza bir kez yüklenmesi gereklidir, daha önce uyguladıysanız bu adımı atlayabilirsiniz.

Aynı dizinde olduğunuzdan emin olduktan sonra aşağıdaki kod bloğunu çalıştırınız.

```
gulp trust-dev-cert
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Examples\helloworld-webpart> gulp trust-dev-cert
Build target: DEBUG
[13:10:12] Using gulpfile C:\Examples\helloworld-webpart\gulpfile.js
[13:10:12] Starting gulp
[13:10:12] Starting 'trust-dev-cert'...
[13:10:12] Starting subtask 'configure-sp-build-rig'...
[13:10:12] Finished subtask 'configure-sp-build-rig' after 7.8 ms
[13:10:12] Starting subtask 'trust-cert'...
[13:10:12] Finished subtask 'trust-cert' after 100 ms
[13:10:12] Finished 'trust-dev-cert' after 111 ms
[13:10:13] -----[ Finished ]-----
[13:10:13] Project helloworld-webpart version:0.0.1
[13:10:13] Build tools version:3.8.8
[13:10:13] Node version:v10.10.0
[13:10:13] Total duration:4.73 s
PS C:\Examples\helloworld-webpart>
```

Developer sertifikasını yükledikten sonra aşağıda kodu çalıştırarak WebPart’ı önizleyebiliriz.

```
gulp serve
```

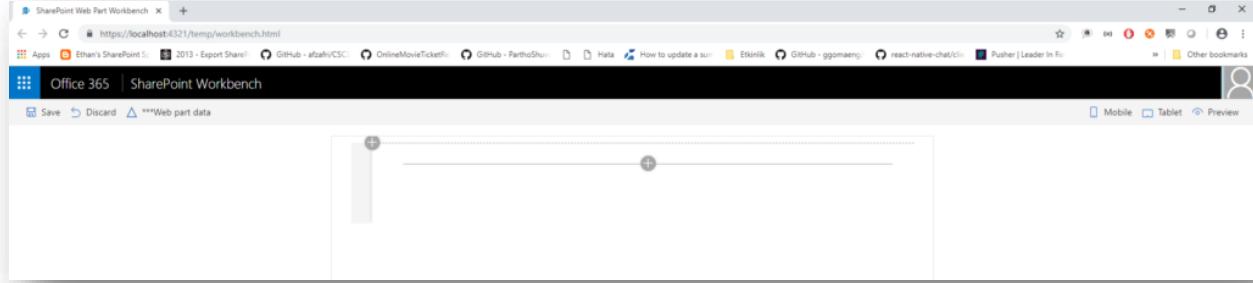
Bu komut, localhost: 4321 ’de HTTPS sunucusu oluşturmak için gulp task yürütür ve WebPartlarınızı dev ortamınızdan önizlemek için varsayılan tarayıcınız üzerinde açılmasını sağlar.

```
PS C:\Examples\helloworld-webpart> gulp serve
Build target: DEBUG
[13:14:16] Using gulpfile C:\Examples\helloworld-webpart\gulpfile.js
[13:14:16] Starting gulp
[13:14:16] Starting 'serve'...
[13:14:16] Starting subtask 'configure-sp-build-rig'...
[13:14:16] Finished subtask 'configure-sp-build-rig' after 5.04 ms
[13:14:16] Starting subtask 'spfx-serve'...
[13:14:17] Starting server...
Starting api server on port 5432.
Registering api: /workbench
Registering api: /*
[13:14:17] Finished subtask 'spfx-serve' after 343 ms
[13:14:17] Starting subtask 'pre-copy'...
[13:14:17] Finished subtask 'pre-copy' after 10 ms
[13:14:17] Starting subtask 'copy-static-assets'...
[13:14:17] Starting subtask 'sass'...
[13:14:18] Server started https://localhost:4321
[13:14:18] LiveReload started on port 35729
[13:14:18] Running server
[13:14:18] Opening https://localhost:5432/workbench using the default OS app
[13:14:19] Finished subtask 'copy-static-assets' after 2.45 s
  Request: [::1] '/workbench'
[13:14:19] Finished subtask 'sass' after 2.62 s
[13:14:19] Starting subtask 'tslint'...
[13:14:19] [tslint] tslint version: 5.9.1
[13:14:22] Starting subtask 'tsc'...
[13:14:22] [tsc] typescript version: 2.4.2
  Request: '/temp/workbench.html'
  Request: '/temp/manifests.js'
  Request: '/temp/workbench-packages/@microsoft_sp-loader/dist/sp-loader-assembly_default.js'
  Request: '/temp/workbench-packages/@microsoft_sp-webpart-workbench/lib/api/workbenchInit.js'
  Request: '/temp/workbench-packages/@microsoft_sp-webpart-workbench/lib/api/assets/server-icon.png'
[13:14:25] Finished subtask 'tsc' after 3.55 s
[13:14:25] Starting subtask 'ts-npm-lint'...
[13:14:25] Finished subtask 'ts-npm-lint' after 76 ms
[13:14:25] Starting subtask 'api-extractor'...
[13:14:25] Finished subtask 'api-extractor' after 1.41 ms
[13:14:26] Finished subtask 'tslint' after 6.57 s
[13:14:26] Starting subtask 'post-copy'...
[13:14:26] Finished subtask 'post-copy' after 592 µs
[13:14:26] Starting subtask 'collectLocalizedResources'...
[13:14:26] Finished subtask 'collectLocalizedResources' after 4.7 ms
[13:14:26] Starting subtask 'configure-webpack'...
[13:14:27] Finished subtask 'configure-webpack' after 1.17 s
[13:14:27] Starting subtask 'webpack'...
[13:14:28] Finished subtask 'webpack' after 844 ms
[13:14:28] Starting subtask 'configure-webpack-external-bundling'...
[13:14:28] Finished subtask 'configure-webpack-external-bundling' after 832 µs
[13:14:28] Starting subtask 'copy-assets'...
[13:14:28] Finished subtask 'copy-assets' after 50 ms
[13:14:28] Starting subtask 'write-manifests'...
```

SharePoint client-side development tools gulp task 'ler ile aşağıdaki adımları gerçekleştirir;

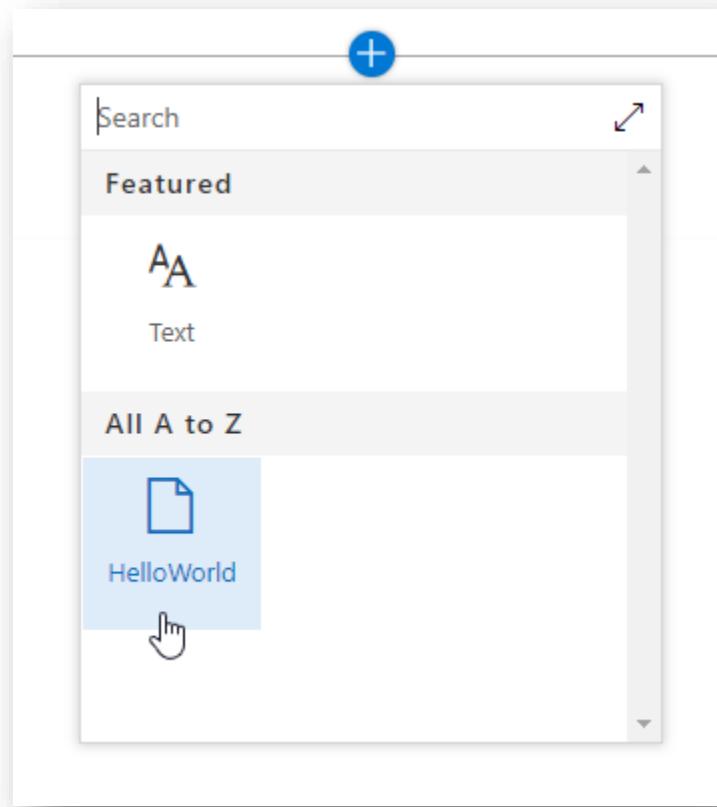
- ✓ JavaScript ve CSS dosyalarının bundle ve minify edilmesi,
- ✓ Bundling ve minification işlemleri için gerekli tool 'ların çağırılması ve çalıştırılması
- ✓ SASS dosyalarının CSS 'e compile edilmesi,
- ✓ TypeScript dosyalarının JavaScript 'e compile edilmesi

“gulp-serve” komutu ile karşınıza gelen SharePoint Workbench ekranıdır. Deploy yapmadan geliştirmiş olduğunuz WebPart ’ınızı hızlı bir şekilde önizlemenizi ve test etmenize olanak sağlamaktadır.

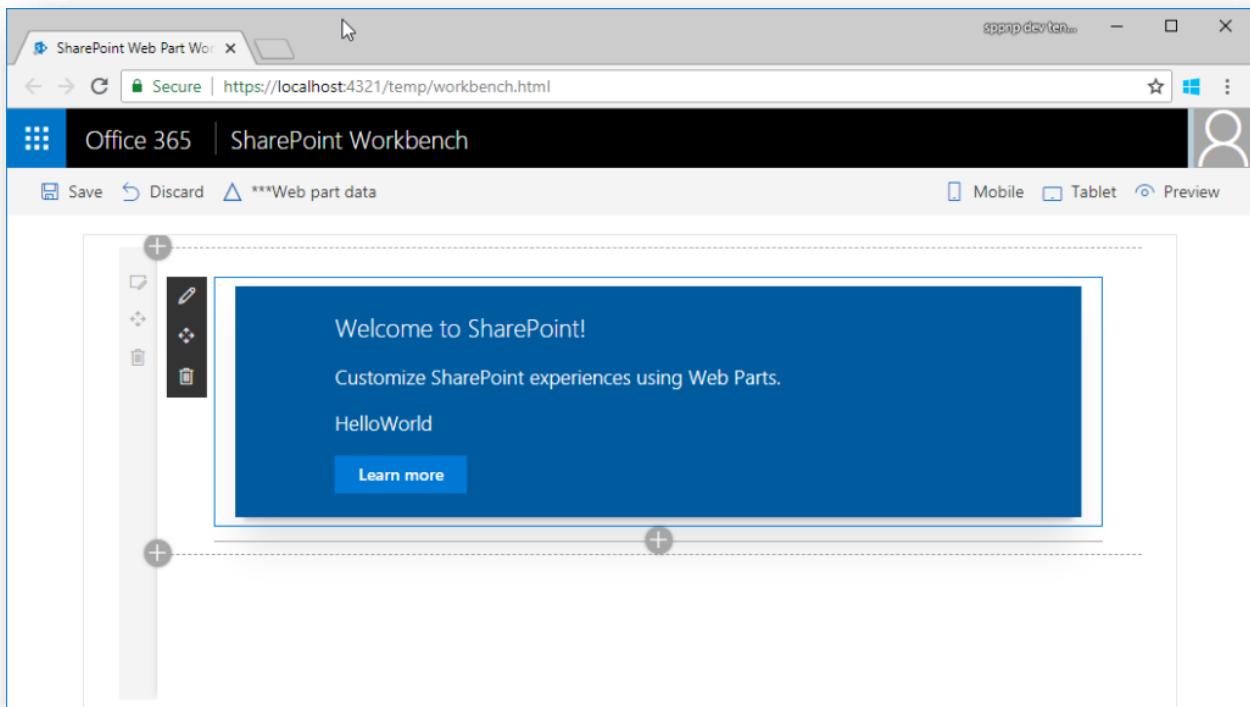


Client-Side WebPart Önizleme ve Test Etme

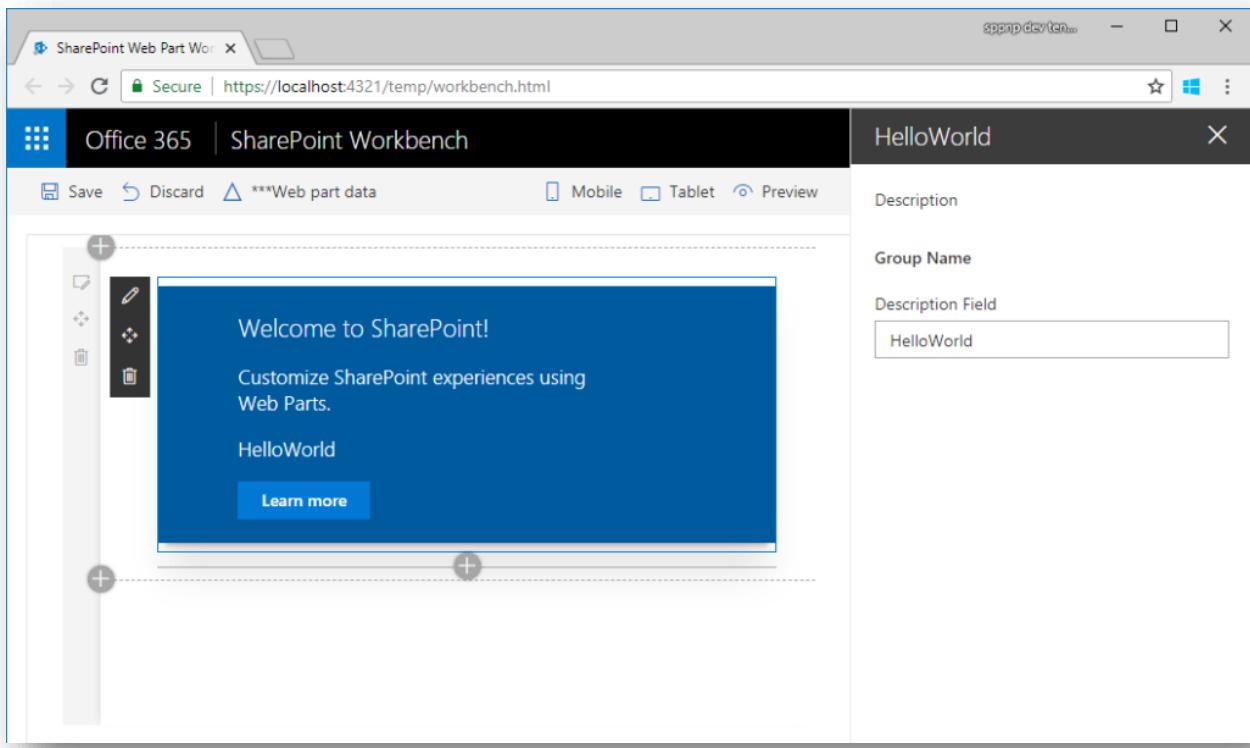
1. Karşınıza gelen sayfa üzerinde WebPart eklemek için “+” butonuna tıklayınız,



2. Sonrasında “HelloWorld” WebPart ’ını seçiniz.



3. Sayfanıza ilgili WebPart’ı ekledik. Şimdi property düzenlemek için kalem ikonuna tıklayınız ve sağ tarafta açılan panel üzerinde değişiklik yaparak test işleminizi gerçekleştiriniz. (Property Pane kısmı ile WebPartınızı özelleştirebilir ve gerekli parametrelerinizi dışarıdan alarak daha dinamik modüller geliştirmenize olanak sağlamaktadır. Property değiştirdiğinizde ve WebPart üzerinde kullandığınızda anlık olarak yansıyor olacaktır.)



SharePoint Framework (SPFx) WebPart Proje Yapısı

Yeoman SharePoint Generator kullanarak oluşturduğunuz WebPart projesini, kullandığınız IDE ile açınız. Folder structure olarak aşağıdaki gibi bir yapı karşınıza gelecektir.

```
▶ HELLOWORLD-WEBPART
  ▶ .vscode
  ▶ config
  ▶ dist
  ▶ lib
  ▶ node_modules
  ▶ src
  ▶ temp
  ⚙ .editorconfig
  ◇ .gitignore
  { } .yo-rc.json
  JS gulpfile.js
  { } package-lock.json
  { } package.json
  ⓘ README.md
  { } tsconfig.json
  { } tslint.json
```

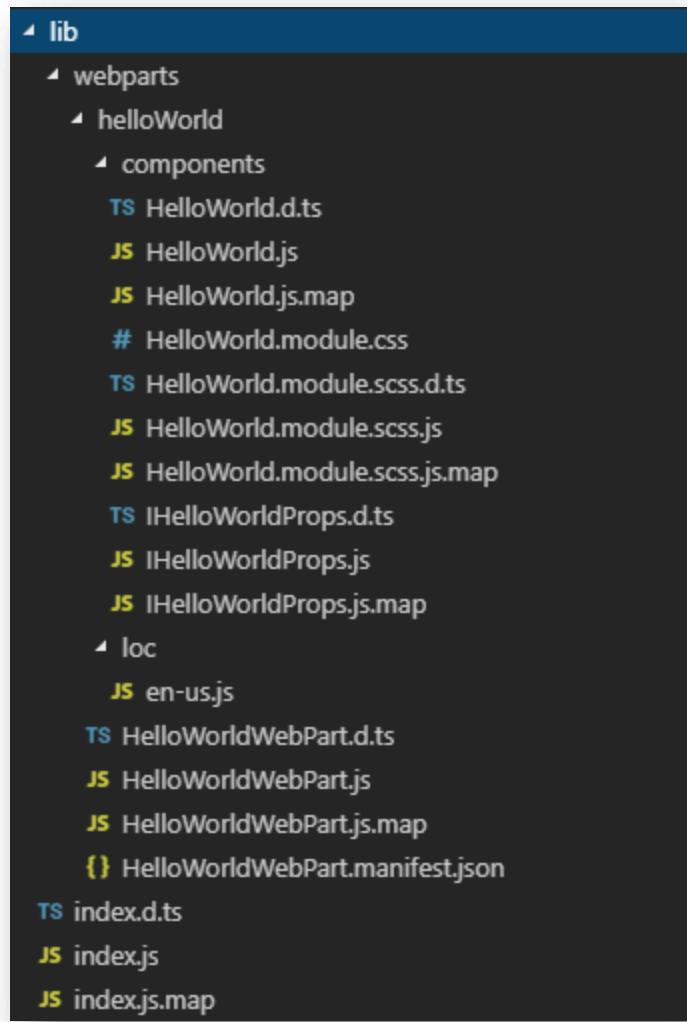
Klasör Yapısı

src:

- ✓ Oluşturduğunuz WebPart 'lar bu klasör içerisinde saklanır.

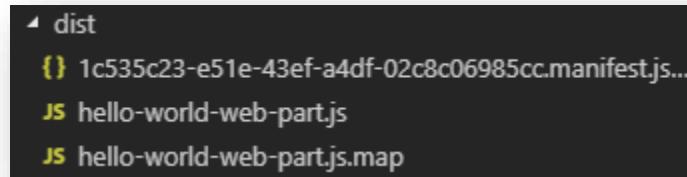
lib:

- ✓ JavaScript dosyalarına derlenmiş (compile) tüm TypeScript dosyaları bu klasörde saklanır.
- ✓ Paketlenip dağıtılabilen işlenmiş kodları içerir.



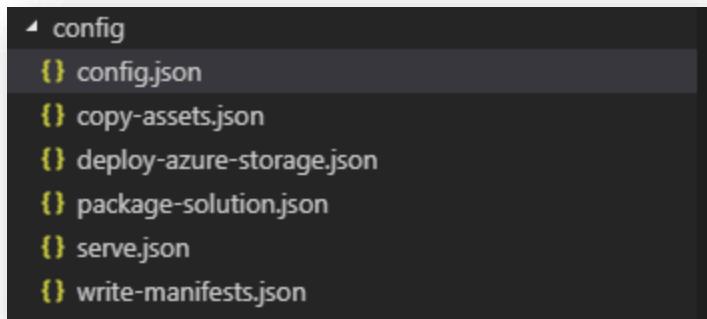
dist:

- ✓ Son dağıtılmış dosyaları içerir.
- ✓ Aynı zamanda WebPart 'ların bundle edilmiş son halleri mevcuttur.



config:

- ✓ Configuration için gerekli olan JSON dosyaları.
- ✓ Bundle edilecek dosyaların configuration bilgileri.



node_modules:

- ✓ Node.js tarafından indirilen modülleri içerir (gulp, pnp, sharepoint framework [Microsoft]...).

typings:

- ✓ Çeşitli kütüphaneler için TypeScript definition dosyalarını içerir.
- ✓ Typings ise JavaScript dosyalarının intellisense desteği sunmaktadır.

sharepoint:

- ✓ “gulp package solution” komutundan sonra oluşan “.spapp” paketini içerir.

[Proje İçerisindeki Önemli Dosyalar](#)

config.json

- ✓ Configuration dosyasıdır. Proje içerisindeki web part bilgilerini, harici referansları (jQuery vs.) ve localization için gerekli olan bilgileri dosyada saklar.

deploy-azure-storage.json

- ✓ Client-Side WebPart Azure CDN 'ye dağıtırken kullanılır. Azure Storage hesabınıza ait bilgileri içerir.

package-solution.json

- ✓ Bu dosya solution 'a ait konfigürasyon bilgisini içerir.

gulpfile.js

- ✓ Çalıştırılacak gulp task 'ını içerir.

package.json

- ✓ Solution içerisinde yer alan library 'i içerir.

tsconfig.json

- ✓ TypeScript compile edilirken gerekli olan konfigürasyon bilgilerini içerir.

[WebPart Class](#)

“HelloWorldWebPart.ts” solution içerisinde yer alan “src/webparts/helloWorld” için entry point olarak belirtilen dosyadır. İlgili dosyayı açtığınızda

“BaseClientSideWebPart” extend olduğunu göreceksiniz. Extend olmasından dolayı ilgili WebPart, base WebPart içerisinde yer alan property ve functionları tetiklemekte ve mirastan devrelan özellikleri kullanmanıza olanak sağlamaktadır.

BaseClientSideWebPart, web part oluşturmak için gereken en düşük functionality’ı uygular. Bu sınıf ayrıca *displayMode*, *web part property*, *web part context*, *web part instanceId*, *web part domElement* ve daha fazlasını içeren read only properties validate etmekte ve erişmenize olanak sağlamaktadır.

Aşağıda yer alan “IHelloWorldWebPartProps” interface ‘in “BaseClientSideWebPart” içerisinde tanımlanmaktadır.

WebPart class oluşturulmadan önce interface tanımlıyoruz. Dolayısıyla web part içerisinde kullanacağınız property’lerinizi önceden tanımlıyoruz. (Extend ettiğimiz class içerisinde kullanacağımız property lerimizi belirtiyoruz.) Aynı zamanda bu kısımda belirtmiş olduğumuz property’ler (“description” gibi) property pane üzerinden kullanılmaktadır.

```
export interface IHelloWorldWebPartProps {
  description: string;
}

export default class HelloWorldWebPart extends BaseClientSideWebPart<IHelloWorldWebPartProps> {
```

```
export interface IHelloWorldWebPartProps {
  description: string;
}
```

WebPart Render Method

Base class içerisinde “protected abstract” olarak tanımlanmış olan “render()” methodu extend ettiğimiz web part içerisinde tetiklenmekte ve “ReactDOM.render(element, this.domElement)” ile “components” içerisinde yer alan “HelloWorld.tsx” tetiklemektedir.

```
protected abstract render(): void;  
/**  
 * This API should be called by web parts that perform Async rendering. Those web part are required to override  
 * the isRenderAsync API and return true. One such example is web parts that render content in an IFrame. The  
 * web part initiates the IFrame rendering in the `render()` API but the actual rendering is complete only after  
 * the iframe loading completes.  
 */
```

BaseClientSideWebPart

```
public render(): void {  
    const element: React.ReactElement<IHelloWorldProps > = React.createElement(  
        HelloWorld,  
        {  
            description: this.properties.description  
        }  
    );  
  
    ReactDOM.render(element, this.domElement);  
}
```

HelloWorldWebPart.ts

```
public render(): void {  
    const element: React.ReactElement<IHelloWorldProps > = React.createElement(  
        HelloWorld,  
        {  
            description: this.properties.description  
        }  
    );  
  
    ReactDOM.render(element, this.domElement);  
}
```

```

1 import * as React from 'react';
2 import styles from './HelloWorld.module.scss';
3 import { IHelloWorldProps } from './IHelloWorldProps';
4 import { escape } from '@microsoft/sp-lodash-subset';
5
6 export default class HelloWorld extends React.Component<IHelloWorldProps, {}> {
7   public render(): React.ReactElement<IHelloWorldProps> {
8     return (
9       <div className={ styles.helloWorld }>
10         <div className={ styles.container }>
11           <div className={ styles.row }>
12             <div className={ styles.column }>
13               <span className={ styles.title }>Welcome to SharePoint!</span>
14               <p className={ styles.subtitle }>Customize SharePoint experiences using Web Parts.</p>
15               <p className={ styles.description }>{escape(this.props.description)}</p>
16               <a href="https://aka.ms/spfx" className={ styles.button }>
17                 <span className={ styles.label }>Learn more</span>
18               </a>
19             </div>
20           </div>
21         </div>
22       );
23     }
24   }
25 }

```

WebPart Property Pane Konfigürasyonu

Property pane alanı HelloWorldWebPart içeresine tanımlı olarak gelmekte ve “getPropertyPaneConfiguration()” ile web part ‘inizin alacağı parametrelerin tanımlandığı kısımdır. Burada tanımlamış olduğumuz property ise “components” içerisinde yer alan “HelloWorld.tsx” içerisindeki “this.props.description” olarak kullanabilmekteyiz.

Property pane içeresine: check box, drop-down list, ve toggle property ekleyelim.

1. HelloWorldWebPart.ts dosyasını açıp, en üst satırında yer alan “@microsoft/sp-webpart-base” alanını aşağıdaki kod bloğu ile güncelleyiniz.

```

import {
  BaseClientSideWebPart,
  IPropertyPaneConfiguration,
  PropertyPaneTextField,
  PropertyPaneCheckbox,
  PropertyPaneDropdown,
  PropertyPaneToggle
} from '@microsoft/sp-webpart-base';

```

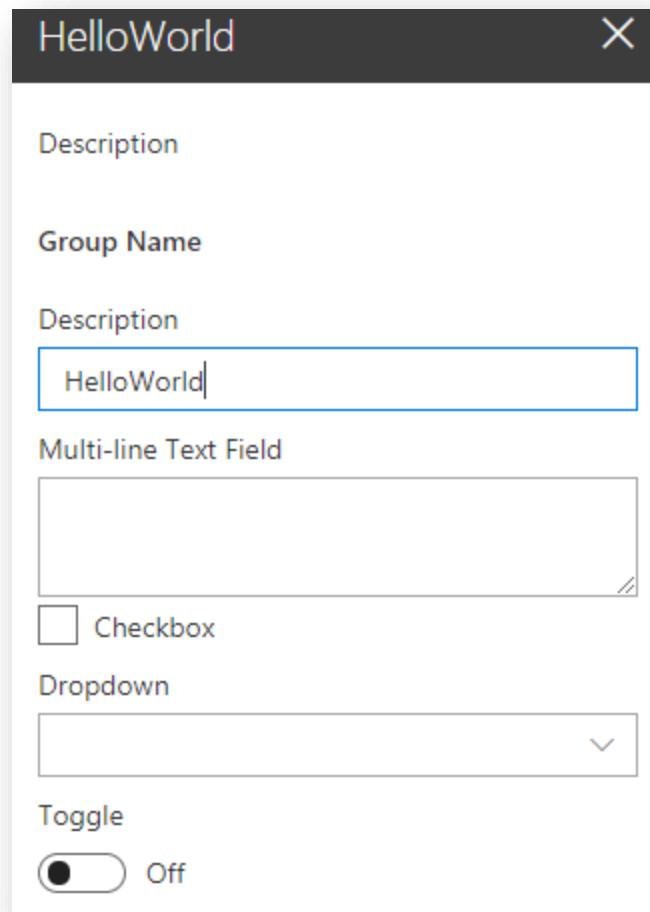
2. Web part içeresine eşleme yapabilmeniz için “IHelloWorldWebPartProps” içeresine property tanımlayalın. (Aşağıdaki kod bloğu ile güncelleyiniz.)

```
export interface IHelloWorldWebPartProps {  
    description: string;  
    test: string;  
    test1: boolean;  
    test2: string;  
    test3: boolean;  
}
```

3. Property pane alanında “IHelloWorldWebPartProps” içerisinde tanımladığımız property kullanabilmek için aşağıdaki gibi güncelleyiniz.

```
protected getPropertyPaneConfiguration(): IPaneConfiguration {  
    return {  
        pages: [  
            {  
                header: {  
                    description: strings.PropertyPaneDescription  
                },  
                groups: [  
                    {  
                        groupName: strings.BasicGroupName,  
                        groupFields: [  
                            PropertyPaneTextField('description', {  
                                label: 'Description'  
                            }),  
                            PropertyPaneTextField('test', {  
                                label: 'Multi-line Text Field',  
                                multiline: true  
                            }),  
                            PropertyPaneCheckbox('test1', {  
                                text: 'Checkbox'  
                            }),  
                            PropertyPaneDropdown('test2', {  
                                label: 'Dropdown'  
                            })  
                        ]  
                    }  
                ]  
            }  
        ]  
    }  
}
```

```
        label: 'Dropdown',
        options: [
            { key: '1', text: 'One' },
            { key: '2', text: 'Two' },
            { key: '3', text: 'Three' },
            { key: '4', text: 'Four' }
        ]
    }),
PropertyPaneToggle('test3', {
    label: 'Toggle',
    onText: 'On',
    offText: 'Off'
})
]
}
]
]
}
];
}
```



4. Gerekli işlemlerimizi yaptık ama “components” içerisinde kullanabilmek için react component içerisinde yer alan props ile eşleme yapmanız gerekmektedir. İlk olarak “components” içerisindeki “IHelloWorldProps.ts” dosyasını açınız ve aşağıdaki gibi güncelleyiniz.

```
export interface IHelloWorldProps {  
  description: string;  
  test: string;  
  test1: boolean;  
  test2: string;  
  test3: boolean;  
}
```

5. Kaydettiğinizde build ederken hata alıyor olacaksınız. Bunun sebebi ise, “HelloWorldWebPart.ts” içerisinde yer alan render() methodu ile ReactDOM render ettiğinden bahsetmiştık. Component içerisinde tanımladığımız property nullable olmamasından dolayı ilgili property setlenmesini beklemektedir.

```
export default class HelloWorldWebPart extends BaseClientSideWebPart<IHelloWorldWebPartProps> {

    public render(): void {
        const element: React.ReactElement<IHelloWorldProps> = React.createElement(
            HelloWorld,
            {
                description: this.properties.description,
            }
        );

        ReactDOM.render(element, this.domElement);
    }
}
```

Aşağıdaki kod bloğu ile güncelleyiniz.

```
public render(): void {
    const element: React.ReactElement<IHelloWorldProps> = React.createElement(
        HelloWorld,
        {
            description: this.properties.description,
            test: this.properties.test,
            test1: this.properties.test1,
            test2: this.properties.test2,
            test3: this.properties.test3,
        }
    );
    ReactDOM.render(element, this.domElement);
}
```

İlgili dosyayı kaydettiğinizde build succeeded olacaktır.

6. components içerisinde yer alan “HelloWorld.tsx” dosyasını açınız ve render kısmını aşağıdaki gibi güncelleyiniz.

```
<p className={ styles.description }>{escape(this.props.description)} {this.props.test}</p>
```



7. WebPart property default değer atamak isterseniz, "HelloWorldWebPart.manifest.json" dosyasını açınız ve aşağıdaki gibi güncelleyiniz (terminal üzerinde web projesini durdurup, tekrardan "gulp serve" deyiniz.)

```
"properties": {  
    "description": "HelloWorld",  
    "test": "Multi-line text field",  
    "test1": true,  
    "test2": "2",  
    "test3": true  
}
```

WebPart Manifest

"HelloWorldWebPart.manifest.json" dosyası, web part 'a ait meta data tutmaktadır (versiyon, id, display name, icon, description gibi) ve her web part içerisinde tanımlı olmalıdır.

Yaptığınız değişiklikleri tekrardan uygulamak ve doğruluğundan emin olmak için aşağıdaki komutu çalıştırınız.

```
gulp serve
```

SharePoint İçerisinde WebPart Önizleme

SharePoint Workbench ile geliştirdiğimiz web part 'ımızı önizleyip, test edebildik fakat SharePoint içerisindeki veri almamız gerekiyorunda origin 'e takılıyor olacağız ve gerçek datalar ile işlem yapamayacağımız. Microsoft bu kısım için çözüm sunmakta ve SharePoint ortamı ile web part 'larımıza entegre etmeyi sağlamaktadır. (SharePoint ortamında önizlemek ve test etmek için projenizi paketlemenize ve yüklemenize gerek olmamaktadır.)

Şimdi neler yapmamız gerekiyor, bir göz atalım.

1. Aşağıdaki url 'i kendi SharePoint ortamınızın adresi ile güncelleseyerek tarayıcınız üzerinde açınız.

```
https://your-sharepoint-tenant.sharepoint.com/\_layouts/workbench.aspx
```

SPFx developer certificate yüklü olmalıdır. Yüklü değilse durdurup "gulp trust-dev-cert" komutunu çalıştırınız. Sonrasında ise "gulp serve" ile yeniden ayağa kaldırınız.

2. Sonrasında Workbench ortamında yaptığımız gibi web partınızı ekleyiniz.



Fark ettiğiniz gibi yukarıdaki ekran görüntüsü ile workbench içerisindeki renkler farklıdır. Bunun nedeni ise, web part renklerini Office UI Fabric Core site üzerinde varsayılan olarak tanımlı olan site içerisinde almasından kaynaklıdır.

SharePoint Framework Projesine İkinci WebPart Nasıl Eklenir?

İkinci bir WebPart eklemek için projenizin path 'i içerisinde aşağıdaki komutu çalıştırırmak, yani Yeoman generator tekrardan çalıştırmanız yeterli olacaktır.

```
yo @microsoft/sharepoint
```

```
C:\Examples\helloworld-webpart>yo @microsoft/sharepoint

  _--(o)--_
  ( _~U~_ )
  /  A   \ \
  |    ~   |
  -  *  -  Y  -
              Welcome to the
              SharePoint Client-side
              Solution Generator

? Which type of client-side component to create? (Use arrow keys)
> WebPart
  Extension
```

Komutu çalıştırdıktan sonra solution oluşturulduğunu algılayıp, direkt olarak size hangi client-side component oluşturmak istediğiniz soruyor olacaktır. “WebPart” seçip “enter” tuşuna basınız.

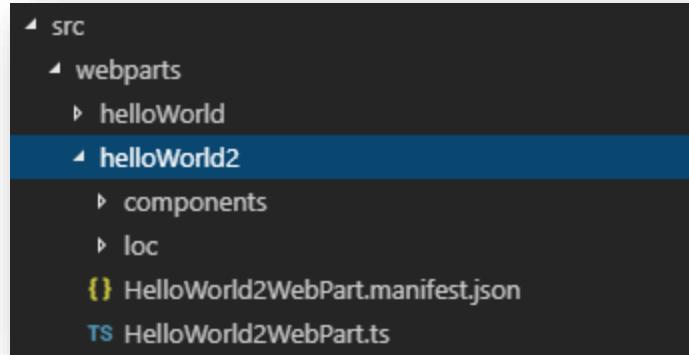
Sonrasında önceki başlıkta bahsettiğim WebPart parametrelerini yazıp, ikinci WebPart oluşturunuz.

```

? Which type of client-side component to create? WebPart
Add new Web part to solution helloworld-webpart.
? What is your Web part name? HelloWorld2
? What is your Web part description? HelloWorld2 description
? Which framework would you like to use? React
  force package.json
  force config\config.json
  force config\serve.json
  force config\package-solution.json
  create src\webparts\helloWorld2\components\IHelloWorld2Props.ts
  create src\webparts\helloWorld2\components\HelloWorld2.module.scss
  create src\webparts\helloWorld2\components\HelloWorld2.tsx
  create src\webparts\helloWorld2\HelloWorld2WebPart.ts
  create src\webparts\helloWorld2\loc\en-us.js
  create src\webparts\helloWorld2\loc\mystrings.d.ts
  create src\webparts\helloWorld2\HelloWorld2WebPart.manifest.json
npm WARN ajv-keywords@3.2.0 requires a peer of ajv@^6.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
audited 757244 packages in 24.839s
found 287 vulnerabilities (117 low, 6 moderate, 164 high)
  run `npm audit fix` to fix them, or `npm audit` for details

      _+#####
      #####|#
###/  (#|(@)  [-----| Congratulations!
###  #####| \  | Solution helloworld-webpart is created.
###/  /###| (@) | Run gulp serve to play with it!
###### #| /
###  /##| (@)
##########
**=+#####

```



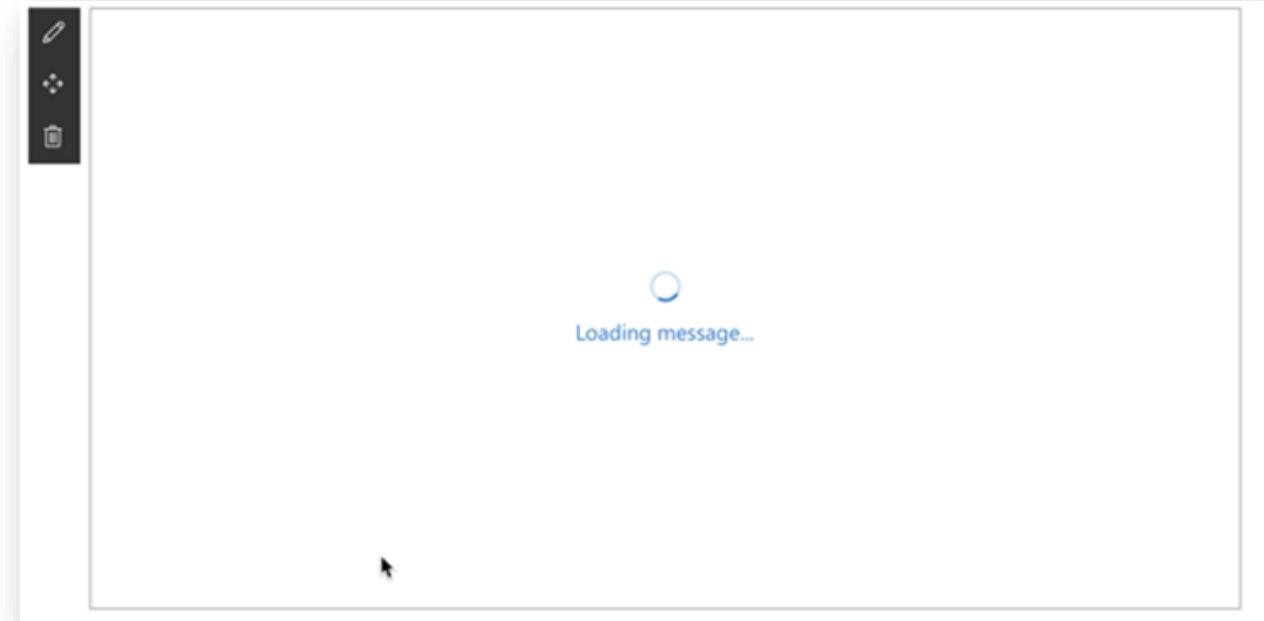
SharePoint Framework WebPart SharePoint Entegrasyonu 1

Loading Indicator

Asenkron olarak SharePoint listesinden ya da başka bir web servisten aldığımız dataları gösterirken “loading” işlemlerine ihtiyaç duyuyoruz. Bunun için context içerisindeki “this.context.statusRenderer” kullanabilirsiniz.

- ✓ **Loading ibaresini görüntülemek için:** displayLoadingIndicator ()
- ✓ **Loading ibaresini temizlemek için:** clearLoadingIndicator ()

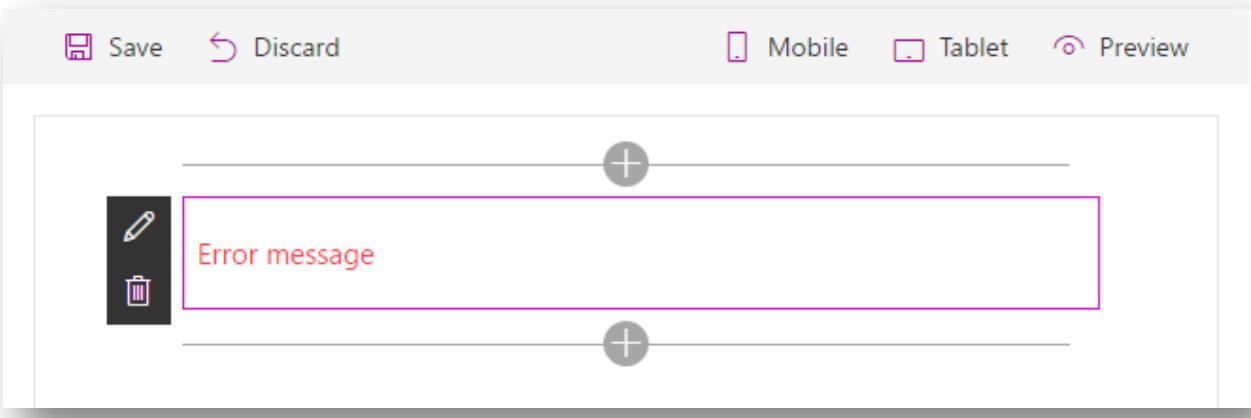
```
this.context.statusRenderer.clearLoadingIndicator(this.domElement);  
this.context.statusRenderer.displayLoadingIndicator(this.domElement, "message");
```



Error Indicator

- ✓ **Hata göstermek için:** renderError
- ✓ **Gösterilen hatayı temizlemek için:** clearError

```
this.context.statusRenderer.renderError(this.domElement, err);  
this.context.statusRenderer.clearError(this.domElement);
```



Lodash Utility Library

Arrays, numbers, strings vs. gibi nesneler üzerinde işlem yapmak için SharePoint Framework lodash kütüphanesini içermektedir.

<https://lodash.com> & <https://www.npmjs.com/package/@microsoft/sp-lodash-subset>

```
import { escape, findIndex } from '@microsoft/sp-lodash-subset';

const index : number = findIndex(this._spItems,
  (item: ISPIItem) => item.Title === 'Mock Title 3');

public render(): void {
  this.domElement.innerHTML = `
    <div class="${ styles.helloWorld }">
      <div class="${ styles.container }">
        <div class="${ styles.row }"><div class="${ styles.column }">
          <p class="${ styles.description }">
            ${escape(this.properties.description)}
          </p>
        </div></div>
      </div>
    </div>`;
```

Page Display Modes

Classic Pages

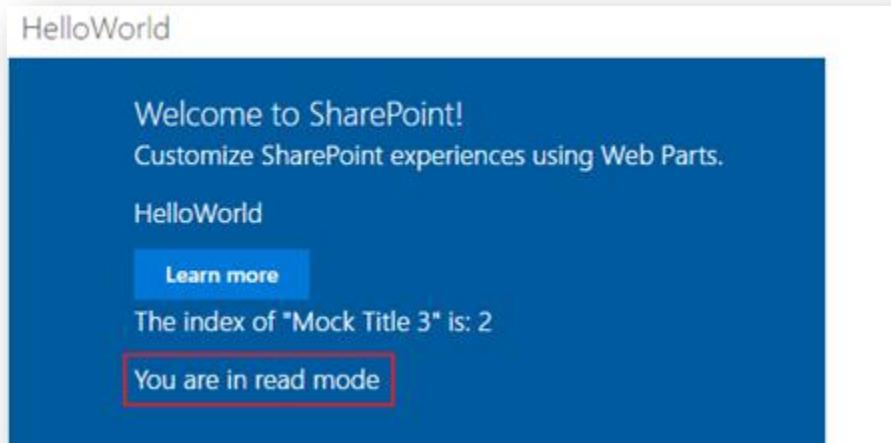
- ✓ Page ve WebPart farklı modlarda görüntülenebilir. (Edit & Display gibi)

Modern Pages

- ✓ Page ve WebPart hep aynı modda görüntülenir. İhtiyacınıza göre modülünüzü aşağıdaki gibi konfigüre edebilirsiniz.

```
import { DisplayMode } from '@microsoft/sp-core-library';

const pageMode : string = (this.displayMode === DisplayMode.Edit)
  ? 'You are in edit mode'
  : 'You are in read mode';
```



Classic Page – Page edit modunda değil

HelloWorld

Welcome to SharePoint!
Customize SharePoint experiences using Web Parts.

HelloWorld

[Learn more](#)

The index of "Mock Title 3" is: 2

You are in read mode

Classic Page – Page edit modunda fakat WebPart edit modunda değil

HelloWorld

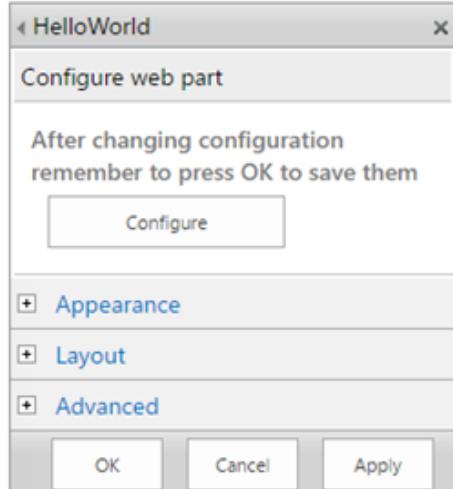
Welcome to SharePoint!
Customize SharePoint experiences
using Web Parts.

HelloWorld

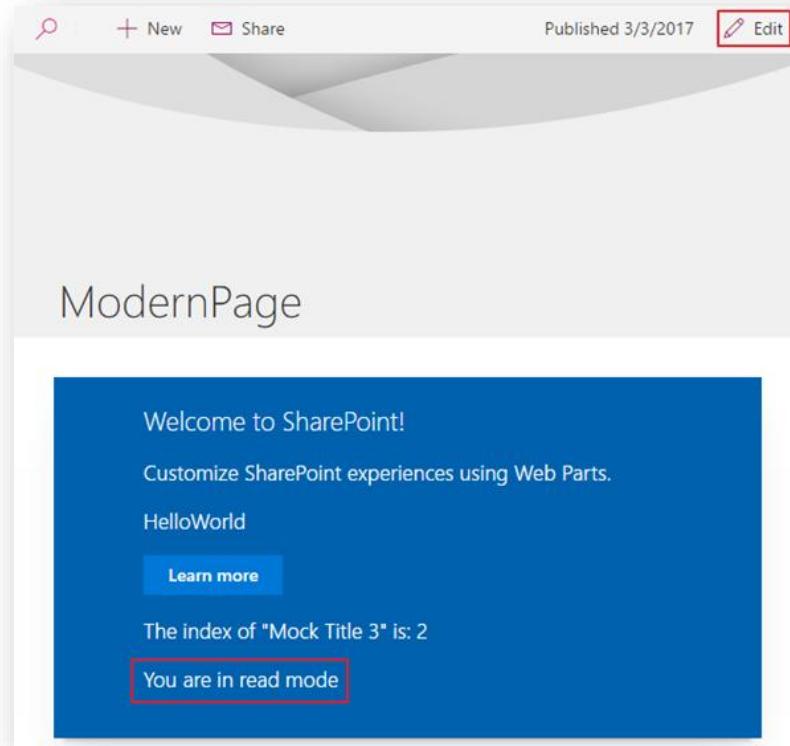
[Learn more](#)

The index of "Mock Title 3" is: 2

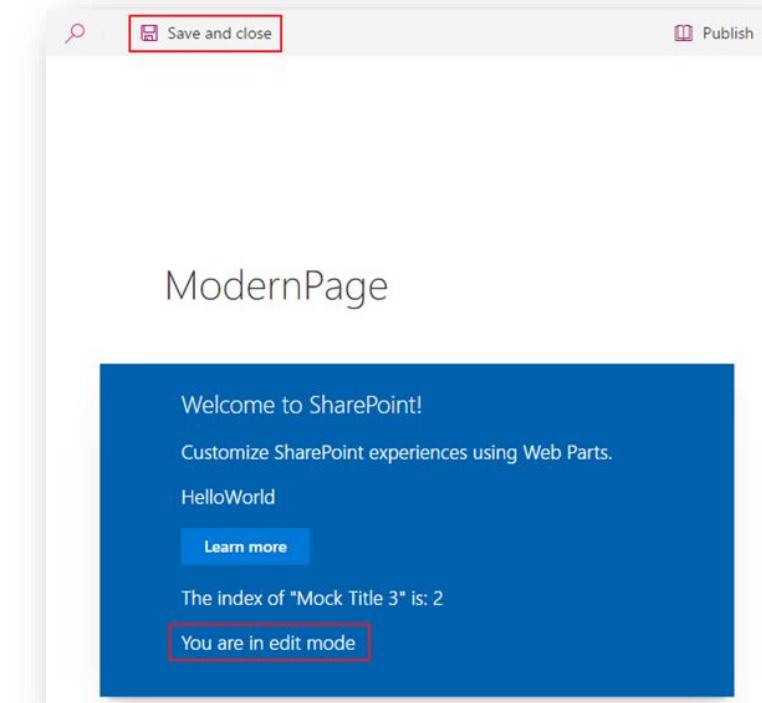
You are in edit mode



Classic Page – Page ve WebPart edit modunda



Modern Page – Read modu



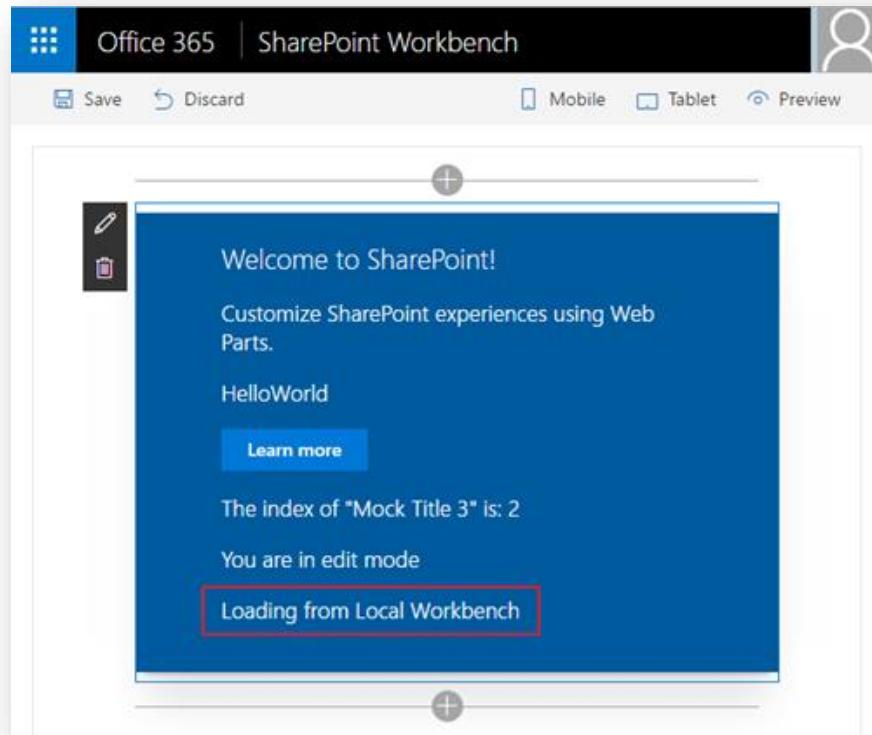
Modern Page – Edit modu

Page Context

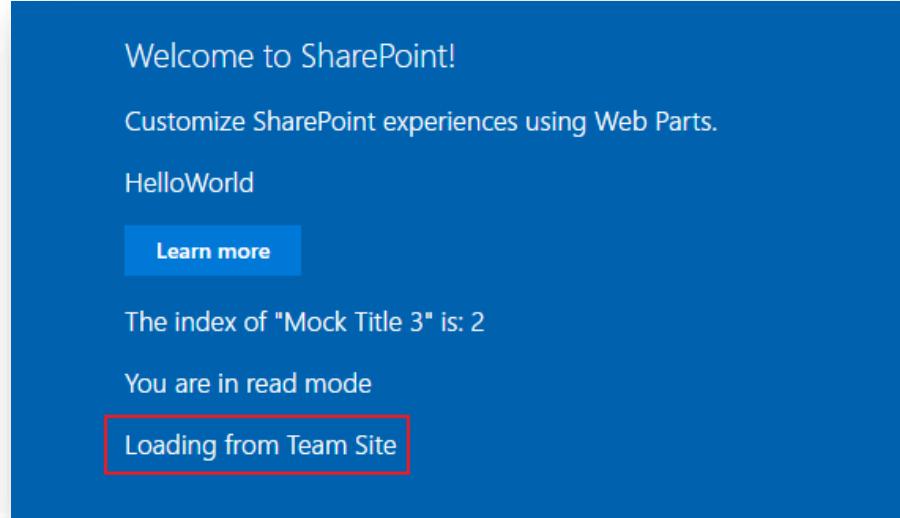
SharePoint Workbench ile çalıştığınızda SharePoint page context ' sahip değilsiniz fakat lokalinizde bir çok özelliği kullanabilirsiniz. SharePoint Workbench üzerinde mock datalar ile simule edip WebPart UX anlamında testlerinizi yapabilirsiniz. Ancak SharePoint Workbench ile bağlandığınızda aşağıdaki özelliklere erişebilirsiniz;

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

```
this.context.pageContext.web.title
```



Local Workbench



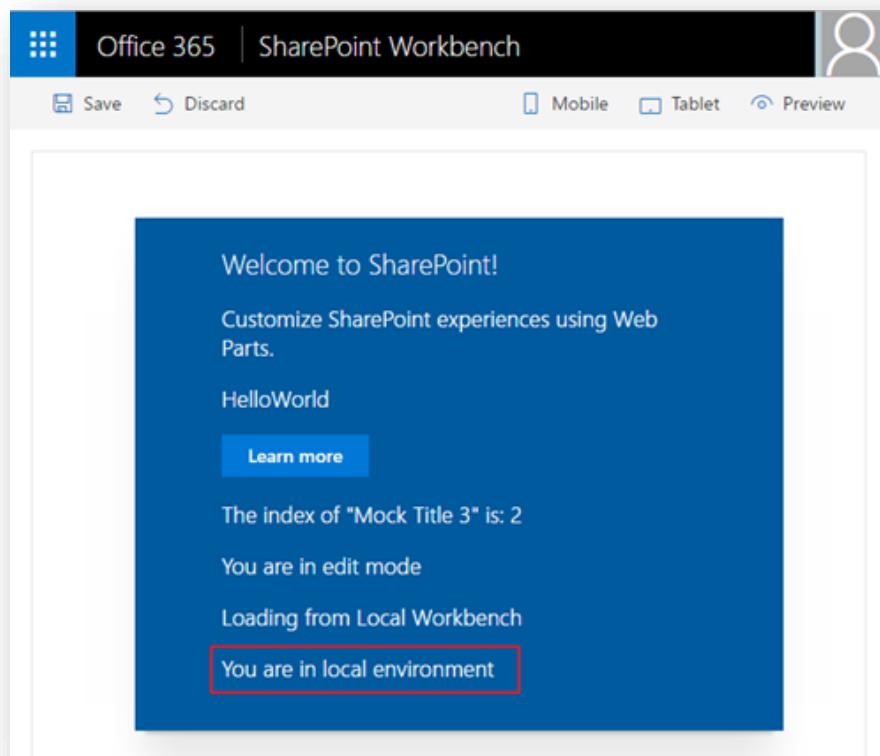
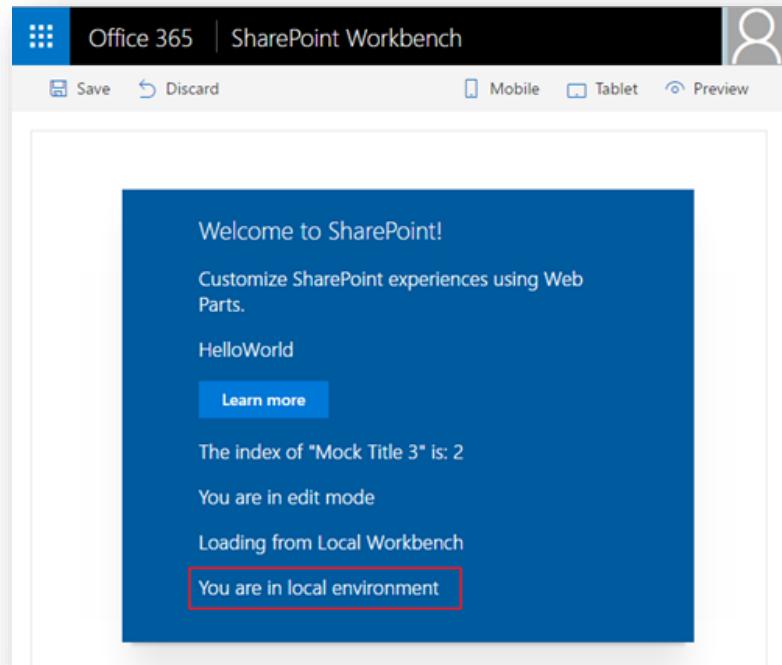
SharePoint Workbench

Environment Type

Environment Type ile modülünüzü geliştirirken SharePoint ya da lokal ayrimınızı yapıp konfigürasyonlarınızı sağlayabilirsiniz.

```
import { Environment, EnvironmentType } from '@microsoft/sp-core-library';

const envType : string = (Environment.type === EnvironmentType.Local)
  ? 'You are in local environment'
  : 'You are in sharepoint environment';
```



Logging

JavaScript içerisindeki Alert ya da Breakpoint ‘ı kullanmak yerine log yazdırıp görüntülemek daha kullanışlı olmaktadır. SharePoint Framework içerisinde built-in logging mekanizmasına sahiptir.

Not: Log sınıfı (class) 4 farklı statik method içermektedir:

- ✓ info: log information
- ✓ warn: log warnings
- ✓ error: log errors
- ✓ verbose: log everythings

SharePoint Framework içerisinde kullanılan tüm log bilgileri JavaScript konsoluna yazdırılmaktadır. Dolayısıyla tarayıcınız üzerinden görüntüleme yapabilirsiniz.

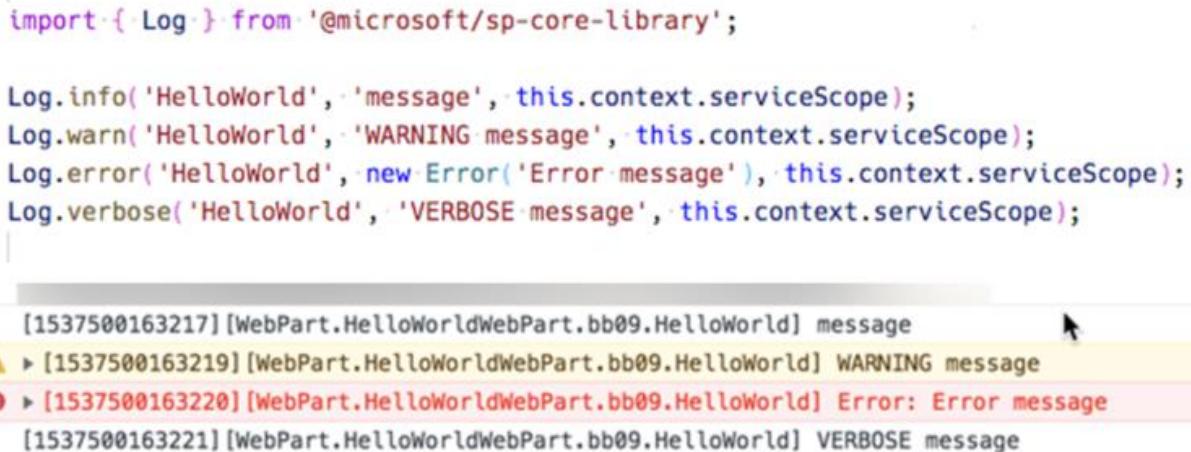
Error dışında diğer 3 method aynı argümanı almaktadır:

- ✓ **source:** Method ya da class name gibi log bilgisini alır (maks. 20 karakter).
- ✓ **message:** Kaydedilecek mesaj (maks. 100 karakter).
- ✓ **scope:** Optional bir parametredir, servis kapsamı.

Error methodunda ise message yerine error object almaktadır.

```
import { Log } from '@microsoft/sp-core-library';

Log.info('HelloWorld', 'message', this.context.serviceScope);
Log.warn('HelloWorld', 'WARNING message', this.context.serviceScope);
Log.error('HelloWorld', new Error('Error message'), this.context.serviceScope);
Log.verbose('HelloWorld', 'VERBOSE message', this.context.serviceScope);
```



The screenshot shows a browser's developer tools console window. It displays four log entries from a script named 'HelloWorld'. The entries are color-coded: a standard message in blue, a warning in orange, an error in red, and a verbose message in green. Each entry includes a timestamp, the source component ('WebPart.HelloWorldWebPart.bb09.HelloWorld'), and the log level followed by the message text.

Timestamp	Source	Level	Message
[1537500163217]	[WebPart.HelloWorldWebPart.bb09.HelloWorld]	message	message
[1537500163219]	[WebPart.HelloWorldWebPart.bb09.HelloWorld]	WARNING	WARNING message
[1537500163220]	[WebPart.HelloWorldWebPart.bb09.HelloWorld]	Error	Error: Error message
[1537500163221]	[WebPart.HelloWorldWebPart.bb09.HelloWorld]	VERBOSE	VERBOSE message

SPComponentLoader

Script ‘lerinizi yüklemek için “SPComponentLoader” class kullanabilirsiniz.

```
import { SPComponentLoader } from '@microsoft/sp-loader';

export default class HelloWorldWebPart
  extends BaseClientSideWebPart<IHelloWorldWebPartProps> {

  public render(): void {
    if (!this.renderedOnce) {
      this.domElement.innerHTML = `<div class="${styles.button}"></div>`;

      SPComponentLoader.loadScript('https://.../jquery.min.js', jQuery)
        .then(($: any): void => {
          this.jQuery = $;
          SPComponentLoader.loadCss('https://.../jqueryui.css');
        });
    }
  }
}
```

SharePoint Framework WebPart SharePoint Entegrasyonu 2

Söyle bir senaryo düşünelim, bulunduğu web içerisinde ve parametre olarak aldığı SharePoint listesine bağlanıp, Title fieldini çekip, WebPart içerisinde görüntüleyelim.

Model Tanımlama

Projemiz içerisinde “models” diye bir klasör oluşturalım. İhtiyacınıza göre siz farklı isimlendirmeler yapabilirsiniz (Bazı kısımları çok karmaşıklıştırmamak adına değiirmiyorum olacağım). Model tanımlamak katmanlar arasında size gelecek data hakkında bilgi vermesine ve daha statik olarak ilerleneceği için OOP açısından sağlıklı olacaktır.

Şimdi adım adım işlemlerimize bir göz atalım:

- ✓ “Item.ts” isminde dosya oluşturalım.

```
export default class Item {
  Title: string;
  Id: number;
```

```
constructor(options: Item) {  
    this.Title = options.Title;  
    this.Id = options.Id;  
}  
}
```

- ✓ “ILIService.ts” isminde dosya oluşturalım.

```
import Item from "./Item";  
import { IWebPartContext } from "@microsoft/sp-webpart-base";  
  
export interface IListService {  
    getItems(): Promise<Array<Item>>;  
    context: IWebPartContext;  
    listName: string;  
}
```

“models” ile ilgili işlemlerimiz bu kadar. İsterseniz [GitHub](#) üzerinden görüntüleyebilirsiniz.

Servis Tanımlama

Verilerimizi component ‘e iletken bir data provider ihtiyacımız bulunmaktadır. Bunun için “services” isimli bir klasör oluşturunuz.

- ✓ “HelloWorldService.ts” isimli dosyayı oluşturup, aşağıdaki gibi güncelleyiniz.
TypeScript ile pattern örneği olarak, singleton pattern kullandım.

```
import Item from '../models/Item';  
import { IListService } from '../models/IListService';  
import {  
    SPHttpClient,  
    SPHttpClientResponse  
} from '@microsoft/sp-http';  
import { IWebPartContext } from '../../../../../node_modules/@microsoft/sp-webpart-base'  
;
```



```

        }
        else {
            reject("Lütfen 'listName' parametresini giriniz.");
        }
    });
}

export default HelloWorldService.getInstance();

```

- ✓ “HelloWorldServiceMock.ts” isimli dosyayı oluşturup, aşağıdaki gibi güncelleyiniz.

```

import Item from '../models/Item';
import { IListService } from '../models/IListService';
import { IWebPartContext } from '../../../../../node_modules/@microsoft/sp-webpart-base';
import {isNull,isEmpty} from 'lodash';

export class HelloWorldServiceMock implements IListService {
    context: IWebPartContext;
    listName: string;
    private static instance: HelloWorldServiceMock;

    private constructor() {}

    static getInstance() {
        if (!HelloWorldServiceMock.instance) {
            HelloWorldServiceMock.instance = new HelloWorldServiceMock();
        }
        return HelloWorldServiceMock.instance;
    }
}

```

```
getItems(): Promise<Array<Item>> {
    return new Promise((resolve, reject) => {
        if (isNull(this.listName) == false && isEmpty(this.listName) == false) {
            const fakeData: Array<Item> = [
                {
                    Id: 0,
                    Title: "Title 1"
                },
                {
                    Id: 1,
                    Title: "Title 2"
                },
                {
                    Id: 2,
                    Title: "Title 3"
                },
                {
                    Id: 3,
                    Title: "Title 4"
                },
                {
                    Id: 4,
                    Title: "Title 5"
                },
            ];
            resolve(fakeData);
        } else {
            reject("Lütfen 'listName' parametresini giriniz.");
        }
    });
}
```

```
    }

}

export default HelloWorldServiceMock.getInstance();
```

“services” ile ilgili işlemlerimiz bu kadar, isterseniz [GitHub](#) üzerinden görüntüleyebilirsiniz.

SharePoint Framework WebPart Güncelleme

Model ve servis tarafı ile ilgili kodumuzu yazdık. Şimdi bunları daha önceden öğrenmiş olduğumuz “EnvironmentType” göre servisleri yüklememiz ve “listName” isimli property’i güncellememiz gerekmektedir. İlk olarak “HelloWorldWebPart.ts” isimli dosyayı açıp aşağıdaki gibi güncelleyiniz. İsterseniz [GitHub](#) üzerinden görüntüleyebilirsiniz.

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { Version, Environment, EnvironmentType } from '@microsoft/sp-core-library';
import {
  BaseClientSideWebPart,
  IPropertyPaneConfiguration,
  PropertyPaneTextField
} from '@microsoft/sp-webpart-base';

import * as strings from 'HelloWorldWebPartStrings';
import HelloWorld from './components/HelloWorld';
import { IHHelloWorldProps } from './components/IHHelloWorldProps';
import { IListService } from './models/IListService';
import HelloWorldService from './services/HelloWorldService';
import HelloWorldServiceMock from './services/HelloWorldServiceMock';

export interface IHHelloWorldWebPartProps {
  listName: string;
}
```

```
export default class HelloWorldWebPart extends BaseClientSideWebPart<IHelloWorldWebPartProps> {

    private _listService: IListService;

    protected OnInit(): Promise<void> {
        if (Environment.type === EnvironmentType.Local) {
            this._listService = HelloWorldServiceMock;
        } else {
            this._listService = HelloWorldService;
        }

        this._listService.context = this.context;
        this._listService.listName = this.properties.listName;

        return super.OnInit();
    }

    public setReactComponent() {
        const element: React.ReactElement<IHelloWorldProps> = React.createElement(
            HelloWorld,
            {
                listService: this._listService,
            }
        );
        ReactDOM.render(element, this.domElement);
    }

    public render(): void {
        this.setReactComponent();
    }

    protected onDispose(): void {
        ReactDOM.unmountComponentAtNode(this.domElement);
    }
}
```

```
}

protected get dataVersion(): Version {
    return Version.parse('1.0');
}

protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {
    return {
        pages: [
            {
                header: {
                    description: strings.PropertyPaneHelloWorld
                },
                groups: [
                    {
                        groupName: strings.BasicGroupName,
                        groupFields: [
                            PropertyPaneTextField('listName', {
                                label: strings.ListNameFieldLabel
                            })
                        ]
                    }
                ]
            };
        }
    }
}
```

Localization Güncelleme

Yukarıdaki WebPart ‘ımızı güncelledikten sonra “loc” klasörü içerisinde yer alan dosyaları güncellemeniz gerekmektedir.

“loc” içerisinde yer alan “mystrings.d.ts” isimli dosyayı açınız ve aşağıdaki gibi güncelleyiniz.

```
declare interface IHelloWorldWebPartStrings {  
    PropertyPaneHelloWorld: string;  
    BasicGroupName: string;  
    ListNameFieldLabel: string;  
}  
  
declare module 'HelloWorldWebPartStrings' {  
    const strings: IHelloWorldWebPartStrings;  
    export = strings;  
}
```

“loc” içerisinde yer alan “en-us.js” isimli dosyayı açınız ve aşağıdaki gibi güncelleyiniz.

```
define([], function() {  
    return {  
        "PropertyPaneHelloWorld": "Hello World Pane",  
        "BasicGroupName": "Group Name",  
        "ListNameFieldLabel": "List Name Field"  
    }  
});
```

React Component Güncelleme

İlk olarak “components” klasörü içerisinde yer alan “IHelloWorldProps.ts” dosyasını açıp aşağıdaki gibi güncelleyiniz. Burada “listService” props tanımlıyoruz ve interface olarak belirtiyoruz, environment göre mock ya da gerçek datanın servisi geliyor olacaktır.

```
import { IListService } from "../models/IListService";  
  
export interface IHelloWorldProps {  
    listService:IListService;
```

```
}
```

“components” klasörü içerisinde “IHelloWorldState.ts” isimli dosyayı oluşturunuz ve aşağıdaki gibi güncelleyiniz.

```
import Item from "../models/Item";

export interface IHelloWorldState {
  items: Array<Item>;
  loading:boolean;
  error?:Error;
}
```

Son olarak React component’ı güncellememiz gerekmektedir. “HelloWorld.tsx” dosyasını açınız ve aşağıdaki gibi güncelleyiniz.

```
import * as React from 'react';
import styles from './HelloWorld.module.scss';
import { IHelloWorldProps } from './IHelloWorldProps';
import { escape } from '@microsoft/sp-lodash-subset';
import { IHelloWorldState } from './IHelloWorldState';
import Item from '../models/Item';

export default class HelloWorld extends React.Component<IHelloWorldProps, IHelloWorldState> {

  constructor(props: IHelloWorldProps) {
    super(props);
    this.state = {
      items: new Array<Item>(),
      loading: true
    };
  }

  public getListItems() {
```

```
        this.props.listService.getItems().then((data: Array<Item>) => {
            this.setState({ items: data, loading: false });
        }).catch((err) => {
            this.setState({ error: err, loading: false });
            console.error('HelloWorld', err);
        });
    }

    public componentDidMount(): void {
        this.getListItems();
    }

    public render(): React.ReactElement<IHelloWorldProps> {

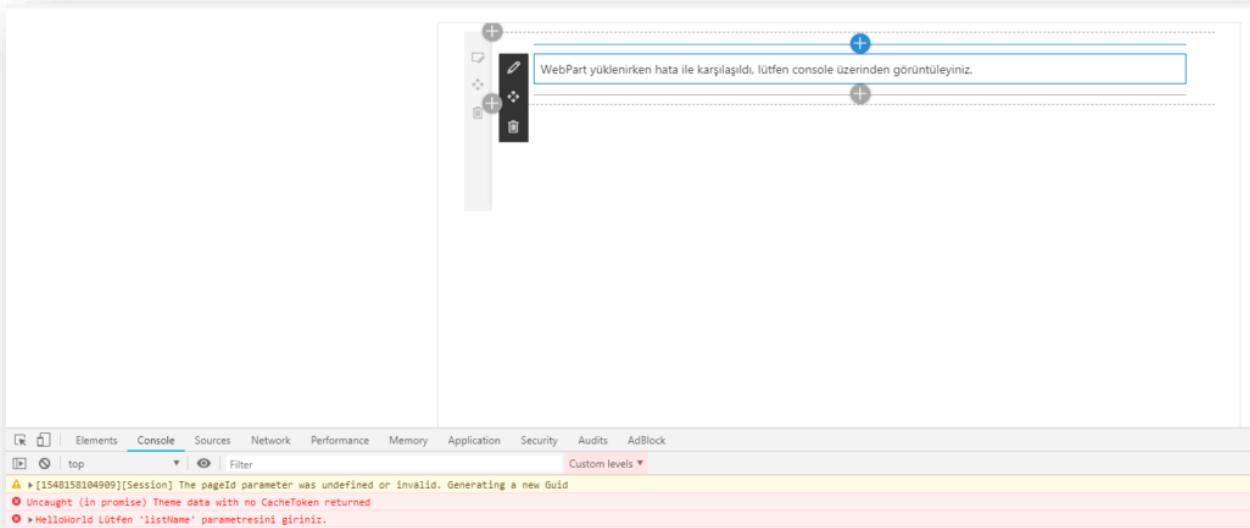
        var getItemRender = this.state.items.map((listItem, i) => {
            return <li key={i}>
                {listItem.Title}
            </li>;
        });

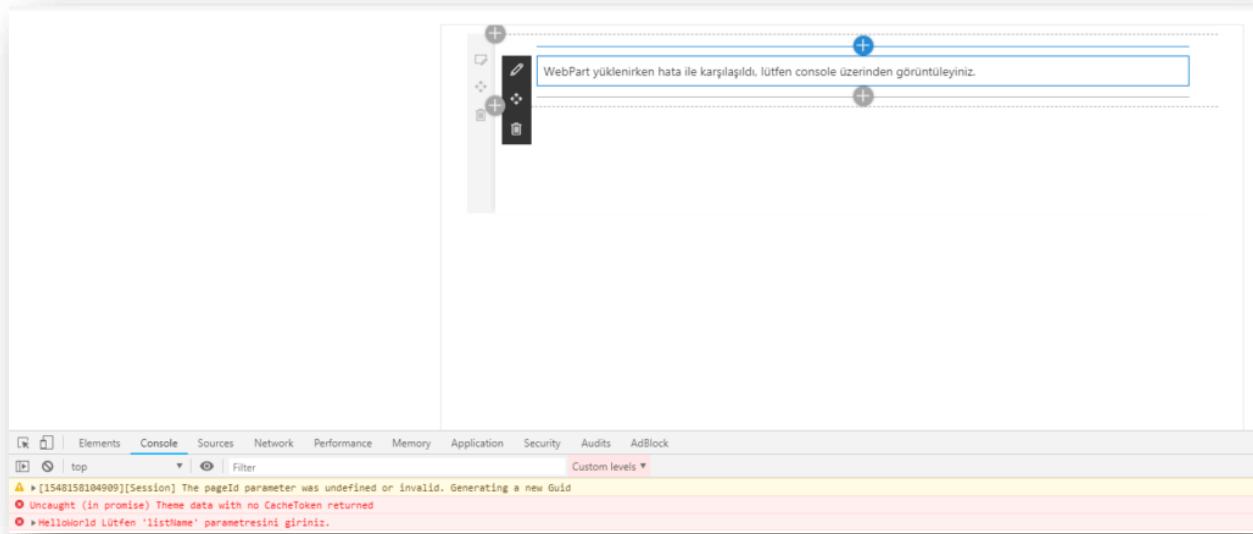
        var getRender = () => {
            if (this.state.loading == true) {
                return <div>Yükleniyor...</div>
            }
            else if (this.state.error) {
                return <div>WebPart yüklenirken hata ile karşılaşıldı, lütfen console üzerinden görüntüleyiniz.</div>
            }
            else {
                return <div>
                    <h1>{this.props.listService.listName}</h1>
                    <ul>
                        {getItemRender}
                    </ul>
                </div>
            }
        }
    }
}
```

```
</ul>
</div>
}

}

return (
  <div className={styles.helloWorld}>
    {getRender()}
  </div>
);
}
```

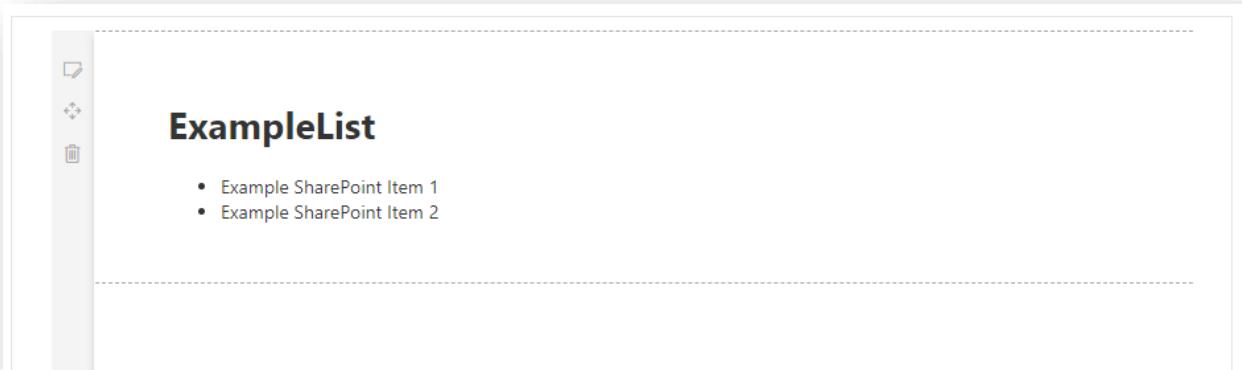




“listName” parametresi olmaması ya da WebPart içerisinde hata olması durumunda verilecek hata örneği;



Local Workbench ekran görüntüsü



SharePoint Workbench ekran görüntüsü

React Component İçerisine SharePoint Framework WebPart Context Aktarılmalı Mıdır?

WebPart context 'i React component içerasine aktarıp kullanmak ilk bakışta uygun görünebilir fakat uzun vadede sorun yaşanmasına sebep olmaktadır. Bu konu ile ilgili önerilen ve uzun vadede neler yapılması gerekenler neler, bir göz atalım.

Kolaylık Ama Gerçekten Sağlıklı Bir Yöntem Mi?

SharePoint Framework kullanarak WebPart ya da Extension oluşturuyoruz. React component içerisinde de verilere ihtiyaç duyuyoruz. Verileri alma noktasında ise, SharePoint Framework HTTP clients kullanarak ilgili işlemimizi gerçekleştiriyoruz. Tam bu noktada kolaya kaçarak React component içerasine WebPart context iletiyoruz. (Verileri çekerken current site url ihtiyaç duyması vs. gibi durumlar için)

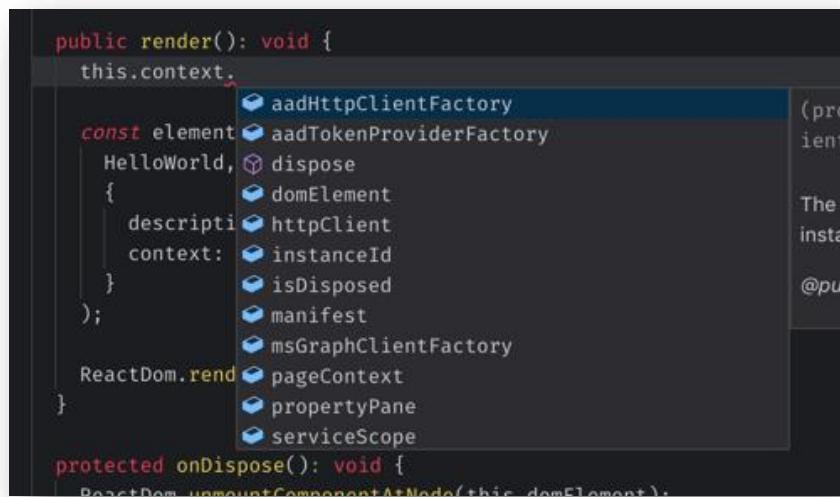
```
17
18  export default class HelloWorldWebPart extends BaseClientSideWebPart<IHelloWorldWebPartProps>
19
20  public render(): void {
21      const element: React.ReactElement<IHelloWorldProps > = React.createElement(
22          HelloWorld,
23          {
24              description: this.properties.description,
25              context: this.context
26          }
27      );
28
29      ReactDOM.render(element, this.domElement);
30  }
31
32  protected onDispose(): void {
33      ReactDOM.unmountComponentAtNode(this.domElement);
```

Aslına bakarsanız, kısa vadede böyle yapmak kolay ama uzun vadede kötü bir fikir.

Peki Neden Kötü Fikir?

Kötü fikir olmasının en büyük sebebi, building unit veya integration test yazılması durumunda yaşanan sorunlardan kaynaklanmaktadır.

Aşağıdaki resim WebPart context class nasıl görüneceğini göstermektedir.



Intellisense box içerisinde gördüğünüz gibi bir çok properties mevcut ve çoğu kendisine ait complex object. Dolayısıyla da unit test yazarken mock object oluşturmak zorunda kalıyoruz. Bu durum ise çok zaman harcanmasına sebep oluyor.

Bu durumdan dolayı React component içerisinde WebPart context göndermek iyi fikir olmuyor, ne yazık ki context göndermek zorunda değiliz.

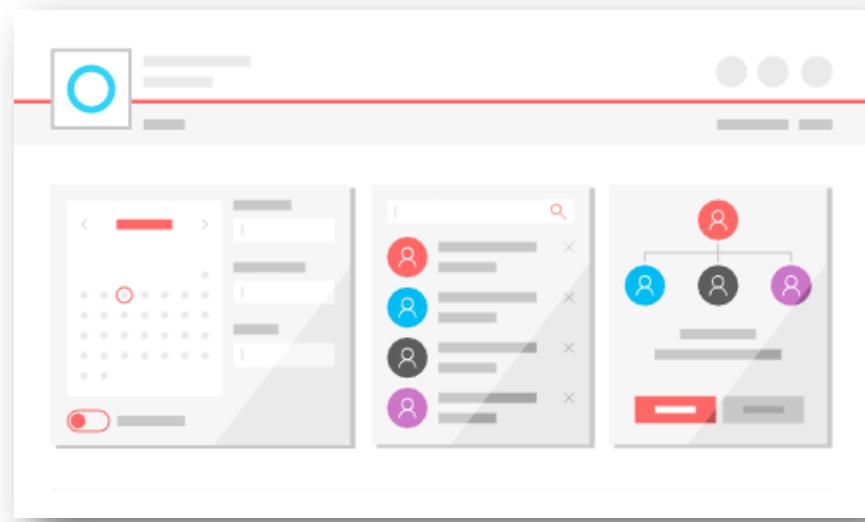
Peki Ne Yapmalıyız?

SharePoint Framework Http client ile verileri çekerken context içerisinde yer alan “var olan sitenin url (current site url)” ihtiyaç duymaktayız. Bunun için tüm context’ı göndermek yerine sadece ihtiyacınız olanı göndermeniz yeterli olacaktır. İlk başta zaman kaybı yaşatabilir ama hem clean code hem de unit test sürecinde çok rahat ediyor olacaksınız.

SharePoint Framework WebPart Office Fabric UI React Kullanımı

Office UI Fabric, Office 365 ve SharePoint (modern team sites, modern pages ve modern lists) tarafından kullanılan front-end framework. Aynı zamanda kendi WebPart’larımız içerisinde kullanmamıza olanak sağlamaktadır.

Office UI Fabric kullanılırken Global CSS styles kullanımına dikkat etmeniz gerekmektedir. Global CSS styles yerine Sass declaration file üzerinde Office Core mixins ve variables kullanmanız gerekmektedir. Global CSS style ile ilgili React CSS in JS linke tıklayarak detaylı bilgi edinebilirsiniz.



Hedef

Key design principles olarak hedefler:

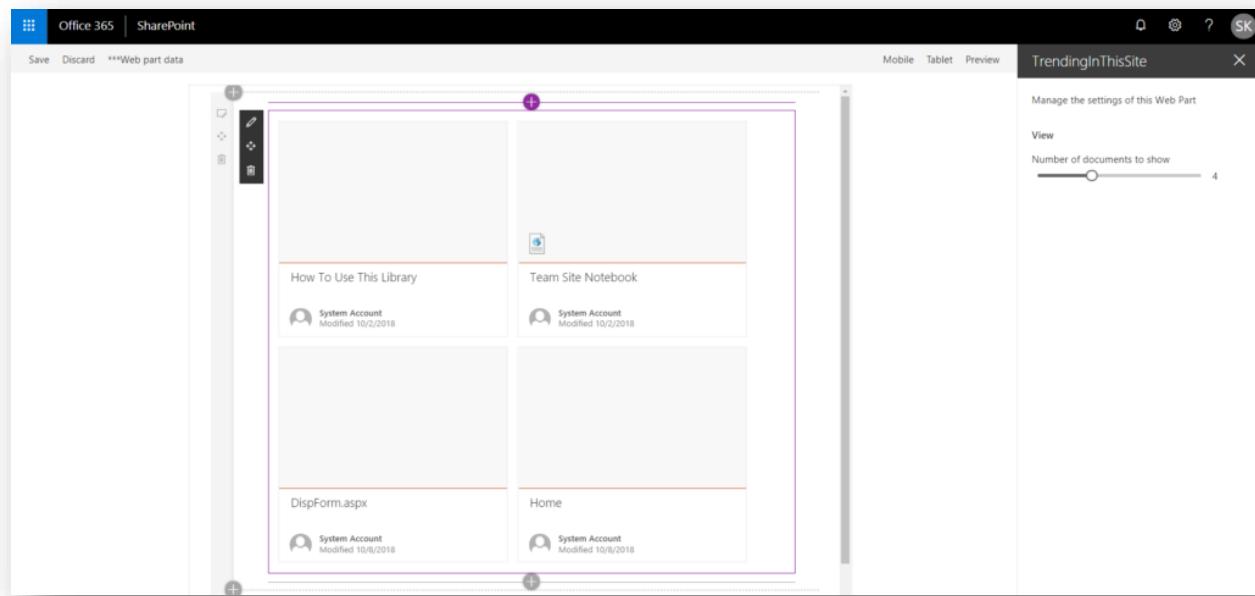
- ✓ Kullanıcıların Fabric Core ve Fabric React sorunsuz bir şekilde kullanılabilmesi
- ✓ Styles, designs ve components istenildiği gibi özelleştirilebilmesi

Office UI Fabric için developer 'lar tarafından kullanıma sunulan iki versiyonu vardır:

- ✓ [Office UI Fabric Core](#), core içerisinde style, typography, responsive grid, animations, icon gibi temel yapı taşlarını içermektedir.
- ✓ [Office UI Fabric React](#), React bazlı kullanılan projeler için React components içermektedir.

Demo

Yukarıdaki bilgilerimizi pekiştirmek için [GitHub](#) üzerinden erişebileceğiniz "Trending In This Site" projesine bir göz atalım.



WebPart içerisindeki bazı yerleri atlayıp (Önceki başlıklarda dejindiğimden dolayı), sadece Office UI Fabric React kullanımını ile ilgili kısımlarını anlatıyor olacağım.

Office UI Fabric React component 'lerini kullanabilmek için React component içerisinde import etmeniz gerekmektedir.

```
import {
  DocumentCard,
  DocumentCardPreview,
  DocumentCardTitle,
  DocumentCardActivity,
  Spinner
} from 'office-ui-fabric-react';
```

import işlemini gerçekleştirdikten sonra import ettiğiniz component 'i render() methodu içerisinde kullanabilirsiniz.

```
<DocumentCard onClickHref={doc.url} key={doc.id}>
  <DocumentCardPreview
    previewImages={[
      {
        previewImageSrc: doc.previewImageUrl,
        iconSrc: iconName,
        width: 318,
        height: 196,
        accentColor: '#ce4b1f'
      }
    ]}
  />
  <DocumentCardTitle title={doc.title} />
  <DocumentCardActivity
    activity={'Modified ' + doc.lastModifiedTime}
    people={
      [
        { name: doc.lastModifiedByName, profileImageSrc: doc.lastModifiedByPhotoUrl }
      ]
    }
  />
```

```
</DocumentCard>
```

```
<Spinner label={'Loading...'} />
```

Görünümü özelleştirmek için oluşturduğunuz WebPart içerisinde yer alan Sass declaration file (*.module.scss) kendinize göre özelleştirebilirsiniz.

```
.trendingInThisSite {  
    .container {  
        max-width: 700px;  
        margin: 0px auto;  
        box-shadow: 0 2px 4px 0 rgba(0, 0, 0, 0.2), 0 25px 50px 0 rgba(0, 0, 0, 0.1);  
    }  
  
    .row {  
        padding: 20px;  
    }  
  
    .listItem {  
        max-width: 715px;  
        margin: 5px auto 5px auto;  
        box-shadow: 0 0 4px 0 rgba(0, 0, 0, 0.2), 0 25px 50px 0 rgba(0, 0, 0, 0.1);  
    }  
  
    .button {  
        text-decoration: none;  
    }  
  
    [class*='ms-DocumentCard '] {  
        float: left;  
        margin: 0.5em;  
    }  
}
```

```
}

[class='ms-Spinner'] {

width: 7em;

margin: 0 auto;

}

}
```

SharePoint Framework WebPart PnP Kütüphanesi

SharePoint Framework WebPart geliştirirken [@pnp/sp](#) kullanabilirsiniz. PnP library, SharePoint Rest API sorgularınızda batching ve caching gibi özellikleri destekleyerek spHttpClient ile request göndermekten kurtarmaktadır.

Örnek olması açısından demo olarak hazırladığım projeye [GitHub](#) üzerinden göz atabilirsiniz.

@pnp/sp Kurulumu

Projenizi oluşturduktan sonra @pnp/sp paketini kurmanız gerekmektedir. Proje içerisine eklemek için:

```
npm install @pnp/logging @pnp/common @pnp/odata @pnp/sp --save
```

@pnp/sp import

Projeniz içeresine @pnp/sp kurulumu yaptıktan sonra kullandığınız kısımlarda import etmeniz gerekmektedir.

```
import { sp } from '@pnp/sp';
```

import işlemini başarıyla gerçekleştirdikten sonra kullanmaya başlayabilirsiniz.

Örneğin; belirttiğiniz listedeki tümdataları almak için:

```
sp.web.lists.getById(listId)
  .items
  .select("Id,Title")
  .getAll()
  .then((data) => {
    resolve(data.map((item) => {
      return new Item({
        Id: item.Id,
        Title: item.Title,
        ObjectStateOption: ObjectStateOptions.Pristine
      })
    }))
  })
  .catch((err) => {
    reject(err);
  });
});
```

Intellisense sayesinde SharePoint Rest API ‘ye hakim olmasanız bile işlemlerinizi gerçekleştirebilirsiniz. Yukarıdaki kod örneğinde “sp.web.lists” ile current site içerisindeki listelere erişik ve “getById” methoduna ilgili “listId” parametre olarak verdik. “items” içerisinde aşağıda gördüğünüz gibi “getById” ile istediğiniz item ‘a erişip işlem yapabilirsiniz ya da “select”, “filter”, “top”, “orderBy” gibi parametreler ekleyip istediğinizdatalara erişebiliriz.

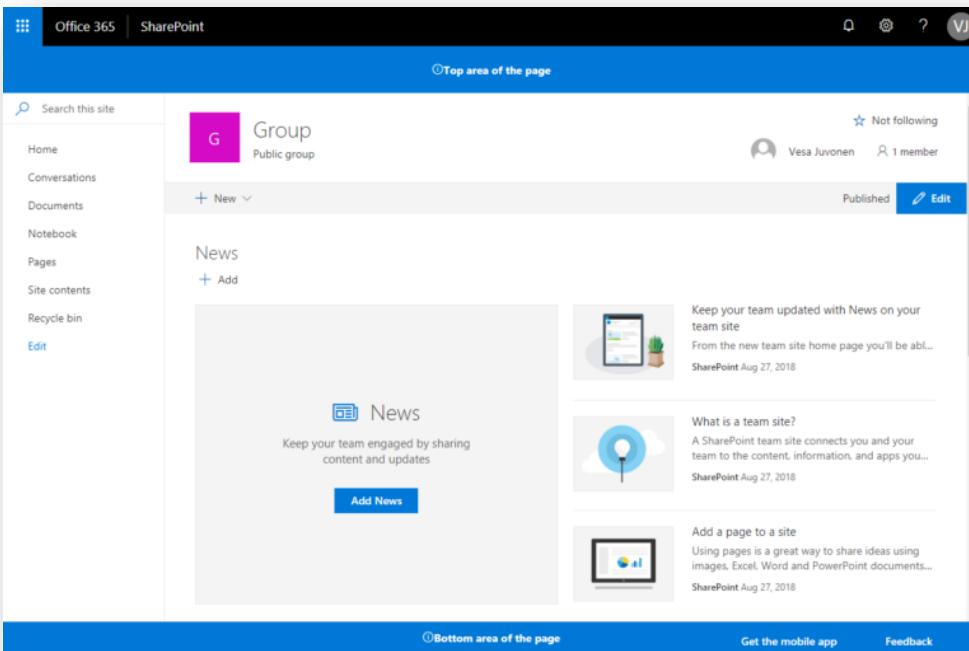
```
sp.web.lists.getById(listId)
  .items
    You, a few seconds ago • Uncommitted changes
      ⚡ getById (method) Items.getById(id: number): Item ⓘ
      ⚡ getItemByIdStringId
      ⚡ getPaged
      ⚡ inBatch
      ⚡ orderBy
      ⚡ query
      ⚡ select
      ⚡ skip
      ⚡ toUrl
      ⚡ toUrlAndQuery
      ⚡ top
      ⚡ usingCaching
```

SharePoint Framework Extensions

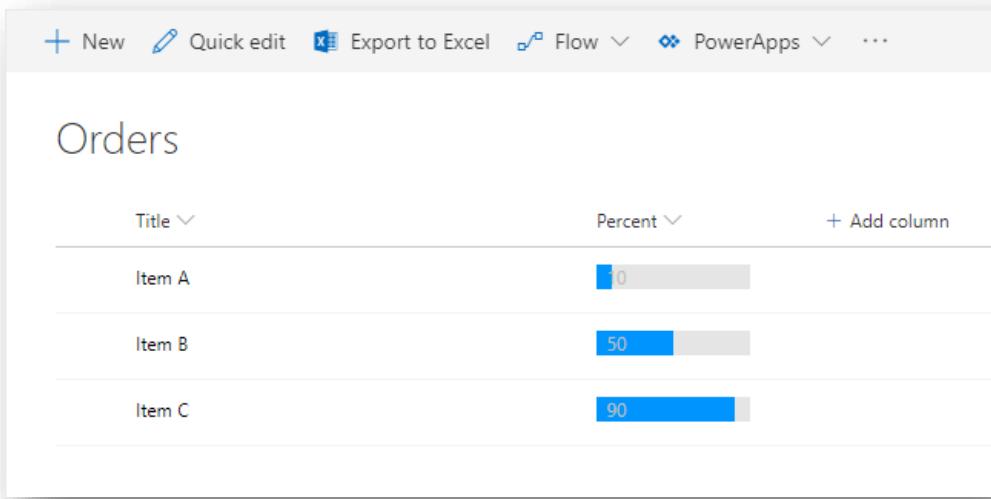
SharePoint user experience anlamında genişletmek için extension kullanabilirsiniz. SharePoint Framework Extensions, notification alanı, toolbars ve list data views gibi bir çok alanı kendi ihtiyacınıza göre özelleştirebilirsiniz.

SharePoint Framework Extensions 3 farklı tipi mevcuttur:

- ✓ **Application Customizers:** SharePoint pages erişmenize ve özelleştirmenize olanak sağlamaktadır. (UserCustomAction benzer bir yapıdır.) Aşağıdaki resimde yer alan sayfa içerisindeki header ve footer alanlarını özelleştirmek gibi.



- ✓ **Field Customizers:** Liste içerisinde yer alan field 'ları özelleştirmenize olanak sağlar. Örneğin; liste içerisindeki yüzde değerinin grafik ile gösterilmesi.



- ✓ **Command Sets:** SharePoint içerisinde action eklemek için kullanabileceğiniz extension tipidir. Örneğin; lisview içine yeni bir command set ekleyip tıklanlığında ihtiyacınıza göre geliştirme yapabilmeniz gibi.

Orders

Title	Percent
Item A	10
Item B	50
Item C	90

Orders

Title	Percent
Item A	10
Item B	50
Item C	90

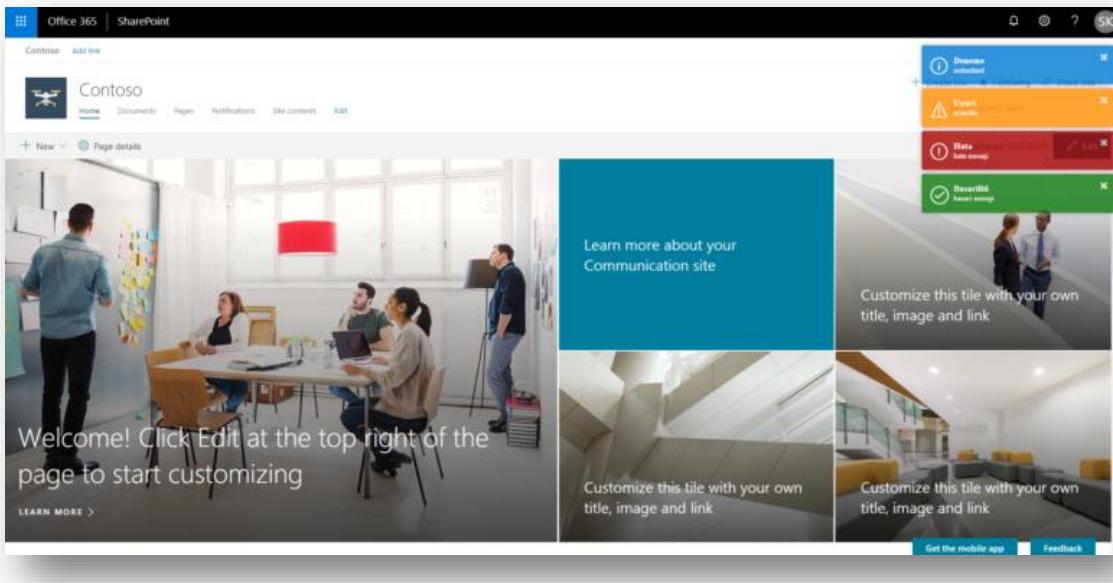
This command is always visible.

OK

SharePoint Framework Extensions Application Customizer

Application Customizer ile SharePoint page üzerinde müdahale edebileceğimizi belirtmiştim. (Örneğin; sayfa üzerinde standart bir header ve footer alanı oluşturmak gibi)

Demo source code için [GitHub](#) linkine göz atabilirsiniz.



Proje Oluşturma

İlk olarak çalıştığınız lokasyon üzerinde aşağıdaki kodu çalıştırarak yeni bir klasör oluşturunuz. (md bulunduğu konum üzerinde klasör oluşturmanızı sağlar.)

```
md app-extension
```

Oluşturduğunuz proje dizinine gitmek için aşağıdaki kodu çalıştırınız. (cd komutu belirttiğiniz dizine gitmenizi sağlar.)

```
cd app-extension
```

Yeoman SharePoint Generator kullanarak yeni solution oluşturalım.

```
yo @microsoft/sharepoint
```

Sizden solution oluşturmak için aşağıdaki bilgileri isteyecektir:

1. “What is your solution name?” default olarak “app-extension” yazmaktadır, “enter” tuşuna basıp kullanabilir ya da farklı bir solution name belirtebilirsiniz.
2. “Which baseline packages do you want to target for your component(s)? (Use arrow keys)” son versiyon üzerinden devam edebilirsiniz.

3. "Where do you want to place the files? (Use arrow keys)" oluşturulacak dosyaların hangi kısma atılmasını istediğiniz belirtiniz. "Use the current folder" deyip, var olan klasör üzerine oluşturmayı tercih ediniz.
4. "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? (y/N)" her siteye uygulamanızı eklemek yerine, tüm sitelerde otomatik olarak dağıtılmış bir şekilde kullanmak istiyorsanız, "y" deyip, "enter" tuşuna basınız. Default olarak "No" yazmaktadır, herhangi bir şey yazmadan "enter" dediğinizde "No" olarak değeri atayacaktır.
5. "Which type of client-side component to create? (Use arrow keys)" client -side component olarak seçiminizi "Extension" olarak yapınız.
6. "Which type of client-side extention to create?" "Application Customizer" olarak seçiniz.

Sonraki parametreler Extension için gerekli olan bilgilerdir:

1. "What is your Application Customizer name?" Extension ismini yazınız.
2. "What is your Application Customizer description?" Extension için açıklama giriniz.

Tüm parametrelerinizi girdikten sonra Yeoman gerekli olan dependencies yükler ve gerekli alt yapıyı hazırlar. Bu işlem 1-2 dakika sürebilir.

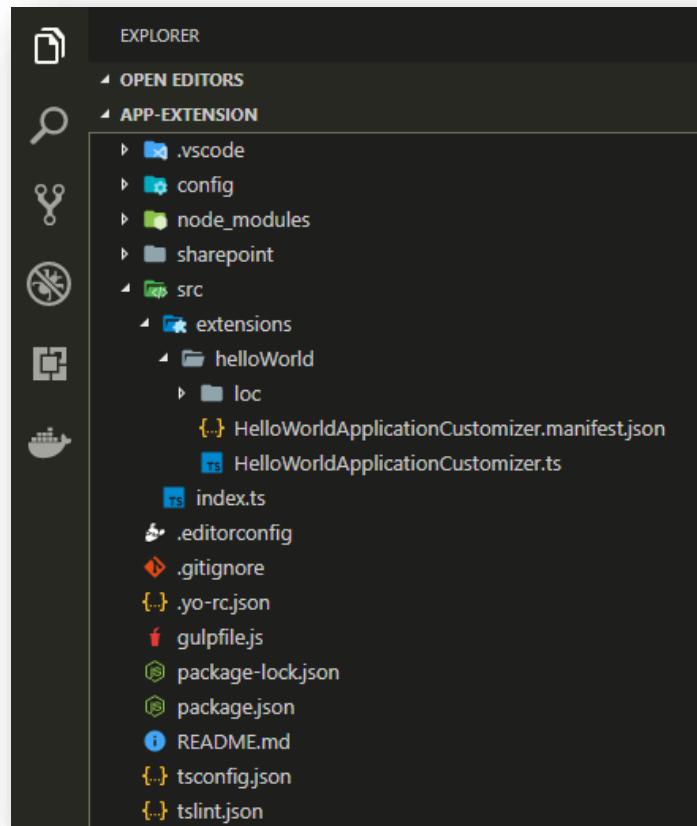
Extension name 40 karakterden uzun olması durumunda "gulp serve –nobrowser" komutunu çalıştırığınızda hata alabilirsiniz. Bu durumda Application Customizer manifest JSON file üzerinden düzenlemeniz gerekmektedir.

```

_+#####
#####
###/ (#|(@) |----- Congratulations!
## ##| \ |----- Solution app-extension is created.
##/ /##| (@) |----- Run gulp serve to play with it!
##### #| /
##| /##| (@)
#####
**=+#####
▶ D\...\app-extension |

```

Visual studio code ile projeyi açtığınızda aşağıdaki gibi solution structure oluşacaktır.



Application Customizer Örnek Kod

“HelloWorldApplicationCustomizer.ts” dosyasını açınız. Projeyi açtığinizda “sp-application-base” paketinin import edildiğini göreceksiniz. “BaseApplicationCustomizer” sınıfı üzerinden oluşturduğumuz extension extend olmaktadır.

```
1 import { override } from '@microsoft/decorators';
2 import { Log } from '@microsoft/sp-core-library';
3 import {
4   BaseApplicationCustomizer
5 } from '@microsoft/sp-application-base';
6 import { Dialog } from '@microsoft/sp-dialog';
7
```

onInit() methodu içerisinde gerekli işlemlerinizi yapabilirsiniz. “this.context” ve “this.properties” WebPart’ı olduğu gibi geçerlidir.

```
@override
public OnInit(): Promise<void> {
    Log.info(LOG_SOURCE, `Initialized ${strings.Title}`);

    let message: string = this.properties.testMessage;
    if (!message) {
        message = '(No properties were provided.)';
    }

    Dialog.alert(`Hello from ${strings.Title}:\n\n${message}`);
}

return Promise.resolve();
}
```

Application Customizer Debug

SharePoint Framework Extentions Local Workbench üzerinde test edemezsiniz. Debug işlemleri için SharePoint Workbench tercih etmeniz gerekmektedir. Şimdi bunun için neler yapmanız gerektiğine bir göz atalım.

- ✓ “config” klasörü içerisinde yer alan “serve.json” dosyasını açınız. (serve.json içerisinde Guid olduğunu göreceksiniz. Buradaki Guid, extension klasöründe yer alan “*.manifest.json” yer alan Guid ile eşleşmesi gerekmektedir.)
- ✓ “pageURL” alanını test edeceğiniz SharePoint sayfası ile güncelleyiniz.

<https://sppnp.sharepoint.com/sites/yoursite/SitePages/Home.aspx>

```
serve.json x
1  {
2    "$schema": "https://developer.microsoft.com/json-schemas/core-build/serve.schema.json",
3    "port": 4321,
4    "https": true,
5    "serveConfigurations": {
6      "default": {
7        "pageUrl": "https://sppnp.sharepoint.com/sites/Group/SitePages/Home.aspx",
8        "customActions": {
9          "05966ad1-55c7-47f6-9ff5-4fee8bff838a": {
10            "location": "ClientSideExtension.ApplicationCustomizer",
11            "properties": {
12              "testMessage": "Test message"
13            }
14          }
15        }
16      },
17      "helloWorld": {
18        "pageUrl": "https://sppnp.sharepoint.com/sites/Group/SitePages/Home.aspx",
19        "customActions": {
20          "05966ad1-55c7-47f6-9ff5-4fee8bff838a": {
21            "location": "ClientSideExtension.ApplicationCustomizer",
22            "properties": {
23              "testMessage": "Test message"
24            }
25          }
26        }
27      }
28    }
29  }
30 }
```

Daha önceden developer certificate yüklememişseniz, aşağıdaki komutu çalıştırınız.

```
gulp trust-dev-cert
```

Test etmek için aşağıdaki komutu çalıştırınız.

```
gulp serve
```

“serve.json” içerişine tanımlamış olduğunuz sayfa açılıyor olacaktır. “Load debug scripts” butonuna tıklayınız.

Allow debug scripts?

WARNING: This page contains unsafe scripts that, if loaded, could potentially harm your computer. Do not proceed unless you trust the developer and understand the risks.

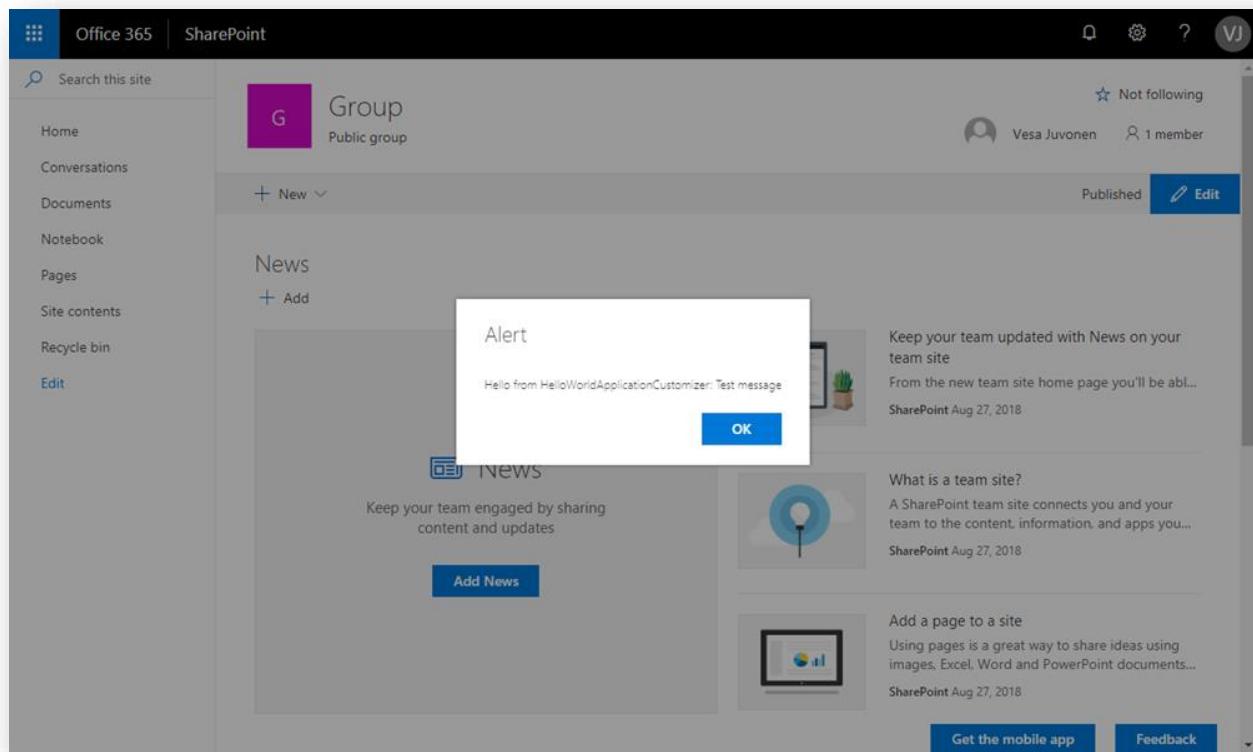
If you are unsure, click Don't load debug scripts.

[Load debug scripts](#)

[Don't load debug scripts](#)

If you don't
know what to
do, click here.

Karşınıza dialog mesajı çıkıyor olacaktır.



SharePoint Framework Extensions Field Customizer

Field Customizer, liste içerisindeki field 'ları customize etmenize olanak sağlamaktadır. Örnek olması açısından seçilen location 'ın hava durumunu getiren field customizer extension yazdım, [GitHub](#) üzerinden ulaşabilirsiniz.

The screenshot shows a SharePoint list titled "Weather". The list contains three items:

- Ankara Hava Durumu: Ankara -5°C
- İstanbul Hava Durumu: İstanbul 13°C
- İzmir Hava Durumu: İzmir 8°C

Proje Oluşturma

İlk olarak çalıştığınız lokasyon üzerinde aşağıdaki kodu çalıştırarak yeni klasör oluşturunuz. (md bulunduğu konum üzerinde klasör oluşturmanızı sağlar.)

```
md app-extension
```

Oluşturduğunuz proje dizinine gitmek için aşağıdaki kodu çalıştırınız. (cd komutu belirttiğiniz dizine gitmenizi sağlar.)

```
cd app-extension
```

Yeoman SharePoint Generator kullanarak yeni solution oluşturalım.

```
yo @microsoft/sharepoint
```

Sizden solution oluşturmak için aşağıdaki bilgileri isteyecektir:

- ✓ “What is your solution name?” default olarak “Weather” yazmaktadır, “enter” tuşuna basıp kullanabilir ya da farklı bir solution name belirtebilirsiniz.
- ✓ “Which baseline packages do you want to target for your component(s)? (Use arrow keys)” son versiyon üzerinden devam ediniz.
- ✓ “Where do you want to place the files? (Use arrow keys)” oluşturulacak dosyaların hangi kısma atılmasını belirtiniz. “Use the current folder” deyip, var olan klasör üzerine oluşturmayı tercih ediniz.
- ✓ “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? (y/N)” her siteye uygulamanızı eklemek yerine, tüm sitelerde otomatik olarak dağıtılmış bir şekilde kullanmak istiyorsanız, “y” deyip, “enter” tuşuna basınız. Default olarak “No” yazmaktadır, herhangi bir şey yazmadan “enter” dediğinizde “No” olarak değeri atayacaktır.
- ✓ “Which type of client-side component to create? (Use arrow keys)” client -side component olarak seçiminizi “Extension” olarak yapınız.
- ✓ “Which type of client-side extention to create?” “Field Customizer” olarak seçiniz.

Sonraki parametreler Extension için gerekli olan bilgilerdir:

- ✓ “What is your Field Customizer name?” Extension ismini yazınız.
- ✓ “What is your Field Customizer description?” Extension için açıklama giriniz.
- ✓ “Which framework would you like to use?” React seçiniz.

Tüm parametrelerinizi girdikten sonra Yeoman gerekli olan dependencies yükler ve gerekli alt yapıyı hazırlıyor olacaktır, 1-2 dk. sürebilir.

```
Let's create a new SharePoint solution.
? What is your solution name? Weather
? 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
Found npm version 6.4.1
? 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
? Which type of client-side component to create? Extension
? Which type of client-side extension to create? Field Customizer
Add new Field Customizer to solution weather.
? What is your Field Customizer name? WeatherFieldCustomizer
? What is your Field Customizer description? Weather SharePoint Framework field customizer
? Which framework would you like to use? React
```

```
_=+#####!
#####
##|  (@) | Congratulations!
##|  \ | Solution app-extension is created.
##| /##| (@) | Run gulp serve to play with it!
##| ##| / |
##| /##| (@)
#####
**=+####!
```

► D\..\..\app-extension |

Field Customizer Debug

Kurulum işlemlerimizi yaptık, şimdi listeye bağlayalım. Hava durumu örneği yaptığım için [GitHub](#) üzerindeki pnp powershell scriptini çalıştırabilir ya da “Weather” isminde “Title, Location” fieldlarını oluşturup kendiniz manuel olarak da oluşturabilirsiniz.

“config” folder içerisinde yer alan “serve.json” dosyasını açınız. “InternalFieldName” alanını customize ettiğiniz field ismi ile güncelleyiniz (“Location” gibi). Sonrasında “pageURL” alanına oluşturduğunuz listenin url’i ile güncelleyiniz.

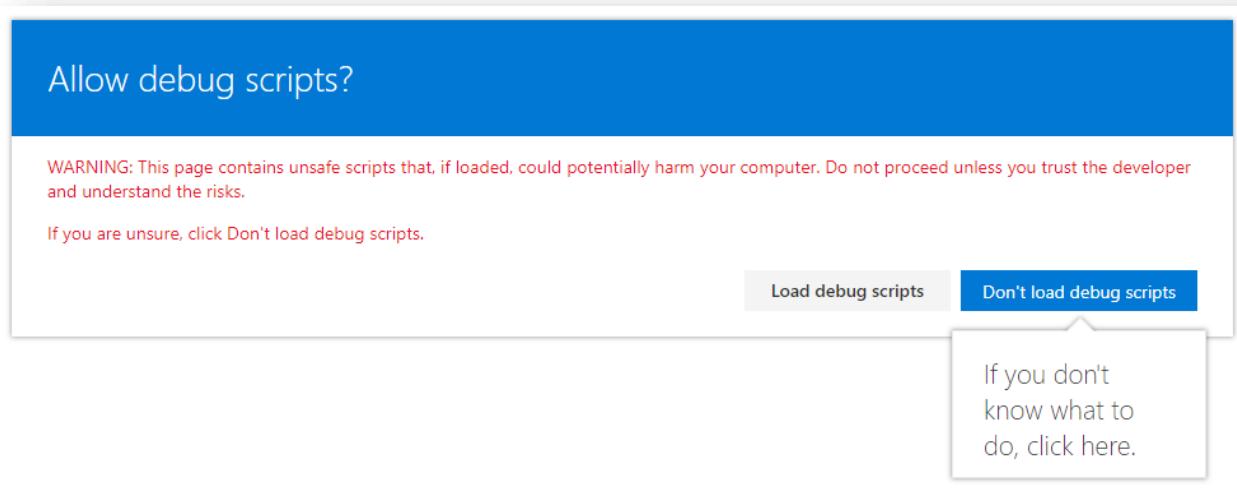
```
{
  "$schema": "https://developer.microsoft.com/json-schemas/core-build/serve.schema.json",
  "port": 4321,
  "https": true,
  "serveConfigurations": {
    "default": {
      "pageUrl": "https://serdarketenci.sharepoint.com/sites/pnp_portal/Lists/Weather/AllItems.aspx",
      "fieldCustomizers": {
        "Location": {
```

```
        "id": "c11b30cc-775f-4760-9273-3f09fa714aaa",
        "properties": {
            "units": "metric"
        }
    },
},
"weatherFieldCustomizer": {
    "pageUrl": "https://serdarketenci.sharepoint.com/sites/pnp_portal/Lists/Weather/AllItems.aspx",
    "fieldCustomizers": {
        "Location": {
            "id": "c11b30cc-775f-4760-9273-3f09fa714aaa",
            "properties": {
                "units": "metric"
            }
        }
    }
}
}
```

Sonrasında aşağıdaki komutu çalıştırınız.

```
gulp serve
```

Karşınıza “pageURL” eklediğiniz link açılıyor olacaktır. “Load debug scripts” butonuna tıklayınız.



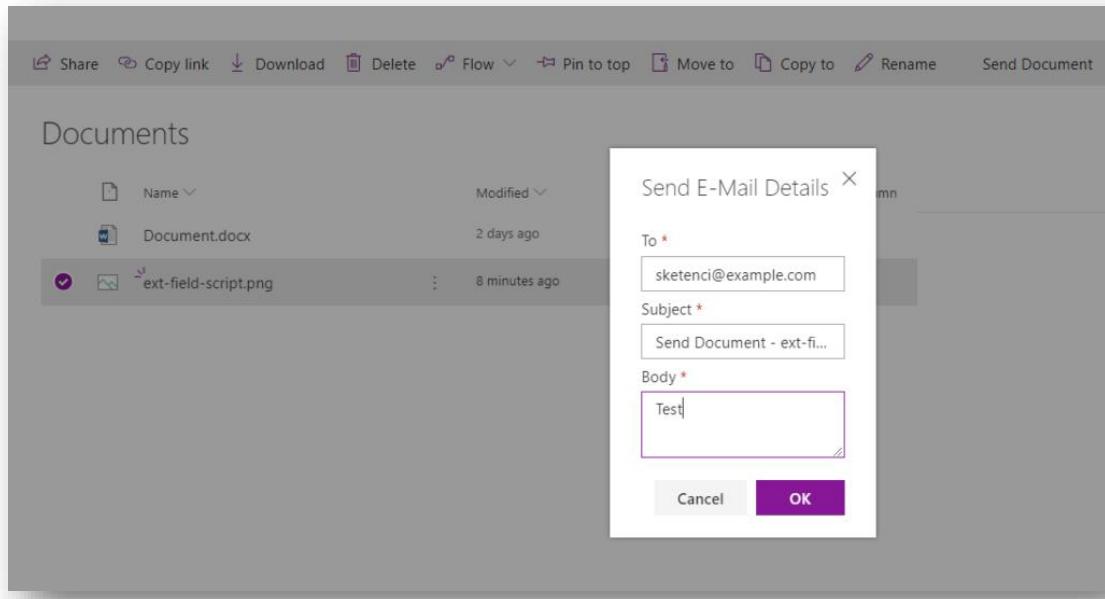
Aşağıdaki gibi ekran görüntüsü oluşacaktır.

Title	Location	
Ankara Hava Durumu	Ankara -5°C	[Edit]
İstanbul Hava Durumu	İstanbul 13°C	[Edit]
İzmir Hava Durumu	Izmir 8°C	[Edit]

SharePoint Framework Extensions Command Set

Command Set, listview üzerinde custom action oluşturmanıza olanak sağlamaktadır. Microsoft sp-dev-fx-extensions reposu içerisinde bir örnek geliştirdim, [GitHub](#) üzerinden göz atabilirsiniz. (Doküman kütüphanesi içerisinde seçmiş olduğunuz

bir dokümanı @pnp/sp ile base64 halini elde ettikten sonra Microsoft Graph ile e-mail göndermektedir.)



Proje Oluşturma

İlk olarak çalışığınız lokasyon üzerinde aşağıdaki kodu çalıştırarak yeni bir klasör oluşturunuz. (md bulunduğu konum üzerinde yeni bir klasör oluşturmanızı sağlar.)

```
md app-extension
```

Oluşturduğunuz proje dizinine gitmek için aşağıdaki kodu çalıştırınız. (cd komutu belirttiğiniz dizine gitmenizi sağlar.)

```
cd app-extension
```

Yeoman SharePoint Generator kullanarak yeni solution oluşturalım.

```
yo @microsoft/sharepoint
```

Solution oluşturmak için aşağıdaki bilgiler istenecektir:

- ✓ “What is your solution name?” default olarak “Send Document” yazmaktadır, “enter” tuşuna basıp kullanabilir ya da farklı bir solution name belirtebilirisiniz.

- ✓ “Which baseline packages do you want to target for your component(s)? (Use arrow keys)” son versiyon üzerinden devam edebilirsiniz.
- ✓ “Where do you want to place the files? (Use arrow keys)” oluşturulacak dosyaların hangi kısma atılmasını belirtiniz. “Use the current folder” deyip, var olan klasör üzerine oluşturmayı tercih ediniz.
- ✓ “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? (y/N)” her siteye uygulamanızı eklemek yerine, tüm sitelerde otomatik olarak dağıtılmış bir şekilde kullanmak istiyorsanız, “y” deyip, “enter” tuşuna basınız. Default olarak “No” yazmaktadır, herhangi birsey yazmadan “enter” dediğinizde “No” olarak değeri atayacaktır.
- ✓ “Which type of client-side component to create? (Use arrow keys)” client -side component olarak seçiminizi “Extension” olarak yapınız.
- ✓ “Which type of client-side extention to create?” “ListView Command Set” olarak seçiniz.

Sonraki parametreler Extension için gerekli olan bilgilerdir:

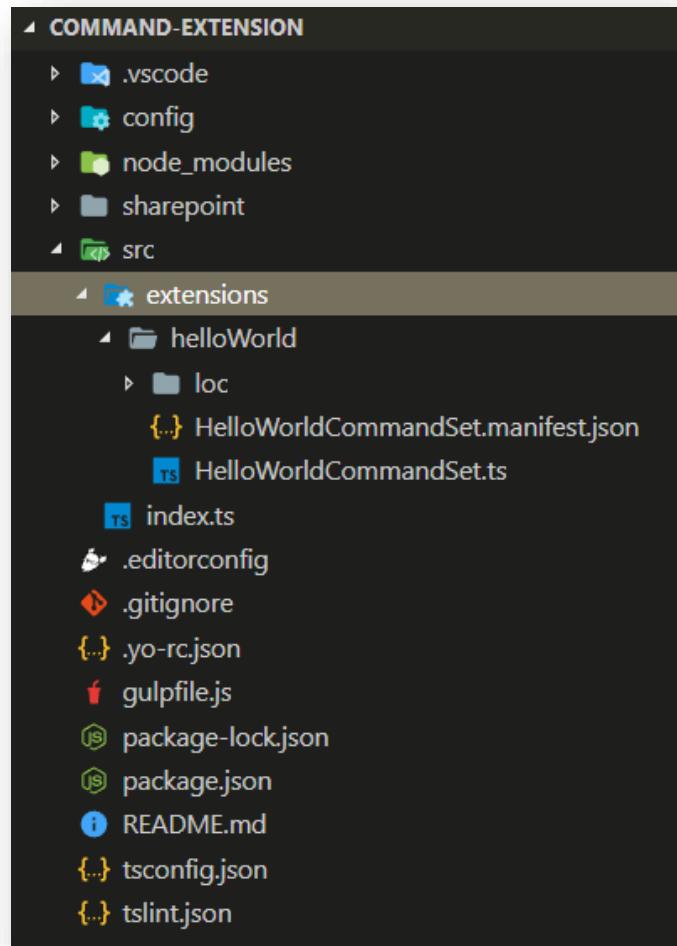
- ✓ “What is your Command Set name?” Extension ismini yazınız.
- ✓ “What is your Command Set description?” Extension için açıklama giriniz.

Tüm parametrelerinizi girdikten sonra Yeoman gerekli olan dependencies yükler ve gerekli alt yapıyı hazırlar. Bu işlem 1-2 dakika sürebilir.

```
D:\...\command-extension> yo @microsoft/sharepoint
[...]
Welcome to the
  SharePoint Client-side
    Solution Generator
[...]

Let's create a new SharePoint solution.
? What is your solution name? command-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
Found npm version 5.8.0
? Do you want to allow the tenant admin the choice of being able to deploy the solution to all sites immediately without
  loyment or adding apps in sites? 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 command-extension.
? What is your Command Set name? HelloWorld
? What is your Command Set description? (HelloWorld description) |
```

Projenizi başarıyla oluşturduktan sonra aşağıda gibi folder structure oluşacaktır.



Extension içerisinde yer alan “.manifest.json” ile parametrelerinize erişebilirsiniz ve ihtiyacınıza göre güncelleme yapabilirsiniz.

ListView Command Set Debug

“config” folder, “serve.json” içerisinde yer alan pageUrl ‘i güncellemeniz gerekmektedir. “gulp serve” demeniz durumunda browser üzerinde ilgili url ‘i açıyor olacaktır.

```
{  
  "$schema": "https://developer.microsoft.com/json-schemas/core-build/serve.schema.json",  
  "port": 4321,  
  "https": true,
```

```

"serveConfigurations": {

    "default": {

        "pageUrl": "https://sppnp.sharepoint.com/sites/Group/Lists/Orders/AllItems.aspx",
        "customActions": {

            "bf232d1d-279c-465e-a6e4-359cb4957377": {

                "location": "ClientSideExtension.ListViewCommandSet.CommandBar",
                "properties": {

                    "sampleTextOne": "One item is selected in the list",
                    "sampleTextTwo": "This command is always visible."
                }
            }
        }
    },
    "helloWorld": {

        "pageUrl": "https://sppnp.sharepoint.com/sites/Group/Lists/Orders/AllItems.aspx",
        "customActions": {

            "bf232d1d-279c-465e-a6e4-359cb4957377": {

                "location": "ClientSideExtension.ListViewCommandSet.CommandBar",
                "properties": {

                    "sampleTextOne": "One item is selected in the list",
                    "sampleTextTwo": "This command is always visible."
                }
            }
        }
    }
}

```

[serve.config ile İlgili Detaylar](#)

- ✓ **GUID:** extension ID 'sidir.
- ✓ **Location:** Extension görüntüleneceği yeri belirtmektedir.

- **ClientSideExtension.ListViewCommandSet.ContextMenu**: item üzerinde açılan context menüdür.
 - **ClientSideExtension.ListViewCommandSet.CommandBar**: Liste ve kütüphane içerisinde görüntülecek üst command set menüdür.
 - **ClientSideExtension.ListViewCommandSet**: Hem context menü hem de command bar üzerinde görüntülenmektedir.
- ✓ **Properties**: Property tanımlayabileceğiniz alandır.

SharePoint Framework Deployment Office 365 CDN

SharePoint Framework (SPFx) ile geliştirmiş olduğunuz application 'ı Office CDN üzerinde kullanabilirsiniz. Varsayılan olarak pasif olmasından dolayı PowerShell script yazarak aktifleştirmeniz gerekmektedir.

Office 365 CDN Aktifleştirme

SharePoint Online Management Shell 'i [link](#) tıklayarak bilgisayarınıza yükleyiniz.

SharePoint Online Management Shell sadece Window üzerinde çalışmaktadır. Farklı bir işletim sisteminiz mevcut ise [Office 365 CLI](#) kullanarak bu adımları gerçekleştirebilirsiniz.

Aşağıdaki komutu yazarak, SharePoint tenant ortamınıza bağlanabilirsiniz.

```
Connect-SPOService -Url https://contoso-admin.sharepoint.com
```

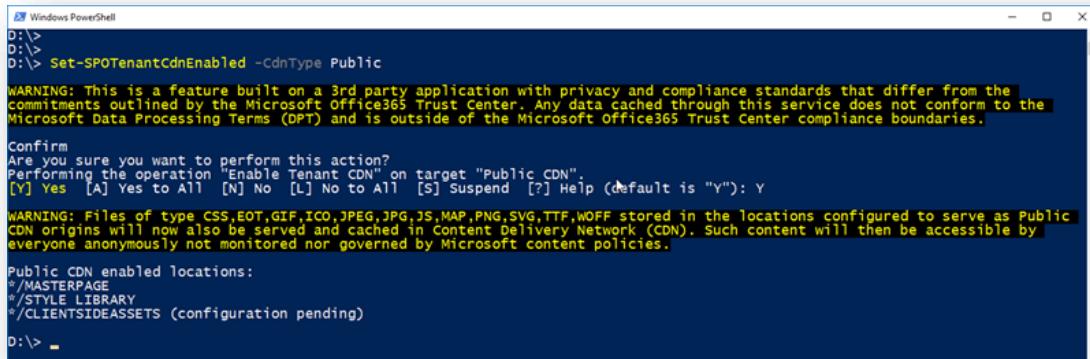
SharePoint tenant hesabınızda CDN 'in aktif edilip, edilmediğini öğrenmek için aşağıdaki komutu kullanabilirsiniz.

```
Get-SPTenantCdnEnabled -CdnType Public  
Get-SPTenantCdnOrigins -CdnType Public  
Get-SPTenantCdnPolicies -CdnType Public
```

SharePoint tenant hesabınızı aktifleştirmek için aşağıdaki komutu çalıştırınız.

```
Set-SPTenantCdnEnabled -CdnType Public
```

Sizden ayarlar için onay isteyecektir. “Y” tuşuna basıp, enter tuşuna basınız.



```
D:\>
D:\>
D:\> Set-SPOTenantCdnEnabled -CdnType Public
WARNING: This is a feature built on a 3rd party application with privacy and compliance standards that differ from the commitments outlined by the Microsoft Office365 Trust Center. Any data cached through this service does not conform to the Microsoft Data Processing Terms (DPT) and is outside of the Microsoft Office365 Trust Center compliance boundaries.
Confirm
Are you sure you want to perform this action?
Performing the operation "Enable Tenant CDN" on target "Public CDN".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
WARNING: Files of type CSS,EOT,GIF,ICO,JPEG,JPG,JS,MAP,PNG,SVG,TTF,WOFF stored in the locations configured to serve as Public CDN origins will now also be served and cached in Content Delivery Network (CDN). Such content will then be accessible by everyone anonymously not monitored nor governed by Microsoft content policies.
Public CDN enabled locations:
  */MASTERPAGE
  */STYLE_LIBRARY
  */CLIENTSIDEASSETS (configuration pending)
D:\>
```

Aktifleştirilmesi biraz zaman alacaktır, Office 365 CDN için gerekli işlemlerimiz bu kadar.

Solution Settings

“config” folder içerisinde yer alan “package-solution.json” dosyasını açınız.

```
{
  "$schema": "https://developer.microsoft.com/json-schemas/spfx-build/package-solution.schema.json",
  "solution": {
    "name": "helloworld-webpart-client-side-solution",
    "id": "3c1af394-bbf0-473c-bb7d-0798f0587cb7",
    "version": "1.0.0.0",
    "includeClientSideAssets": true
  },
  "paths": {
    "zippedPackage": "solution/helloworld-webpart.sppkg"
  }
}
```

“includeClientSideAssets” varsayılan “true” olarak gelmektedir. Bu property ile statik assets “.sppkg” paketi içerisine otomatik dahil edilmektedir.

SharePoint Framework Deploy Süreci

Aşağıdaki komutu çalıştırınız. “gulp bundle –ship” solution ‘ı bundle etmenizi sağlayacaktır.

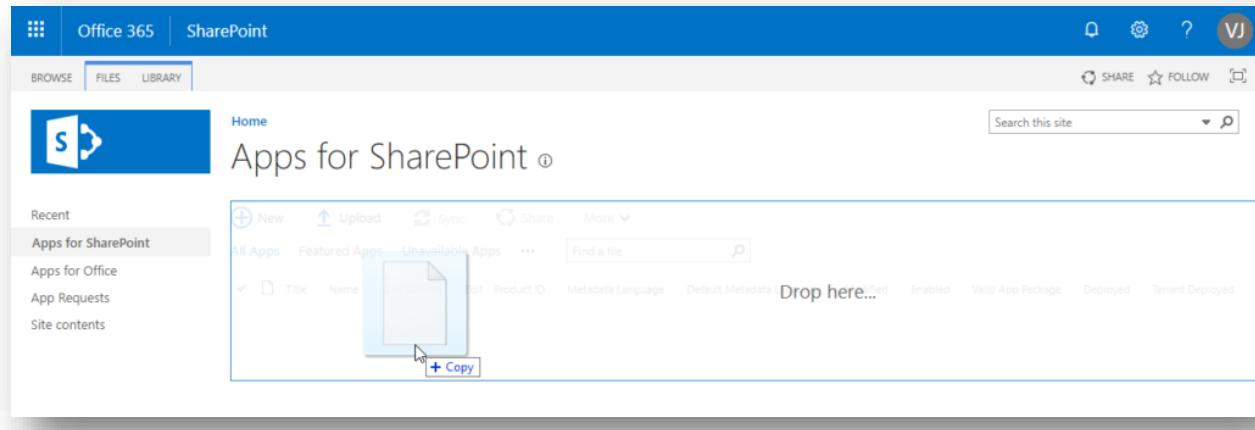
```
gulp bundle --ship
```

Solution ‘ı paket haline getirmek için aşağıdaki komutu çalıştırınız. Komut işlemini tamamladıktan sonra config için belirtmiş olduğunuz path içerisinde oluşturmaktadır (solution klasörü).

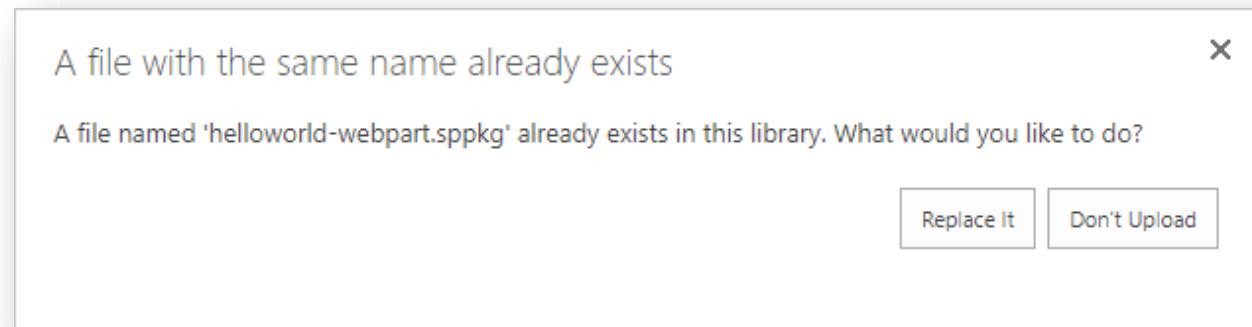
```
gulp package-solution --ship
```

Paket içerisindeğini görüntülemek isterseniz “solution/debug” klasörüne giderek, görüntüleyebilirsiniz.

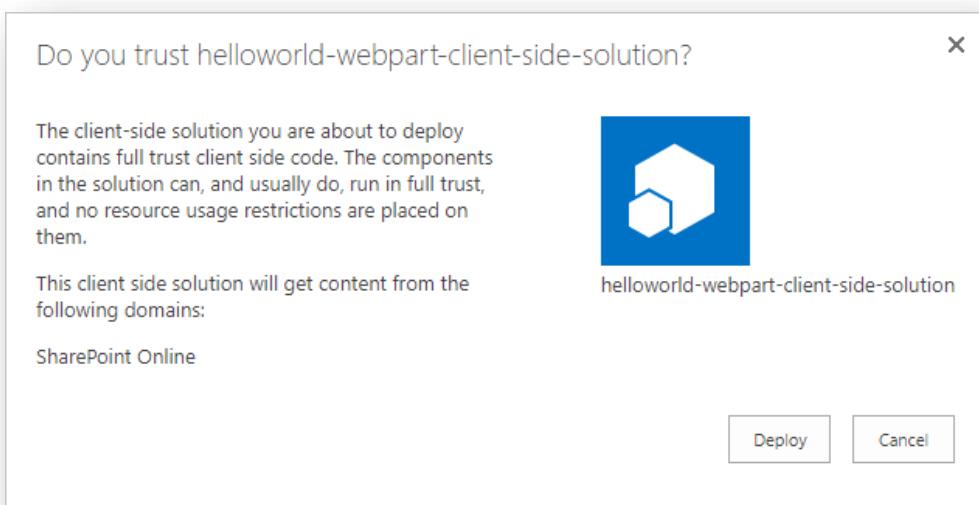
Paketleme işlemimiz bu kadar. Şimdi ilgili “.sppkg” dosyasını SharePoint App Katalog içerisinde yükleyiniz.



Eğer daha önceden aynı isimde bir app mevcut ise size değiştirip, değiştirmeyeceğini soracaktır. “Replace it” butonuna tıklayarak uygulamayı upload edebilirsiniz.



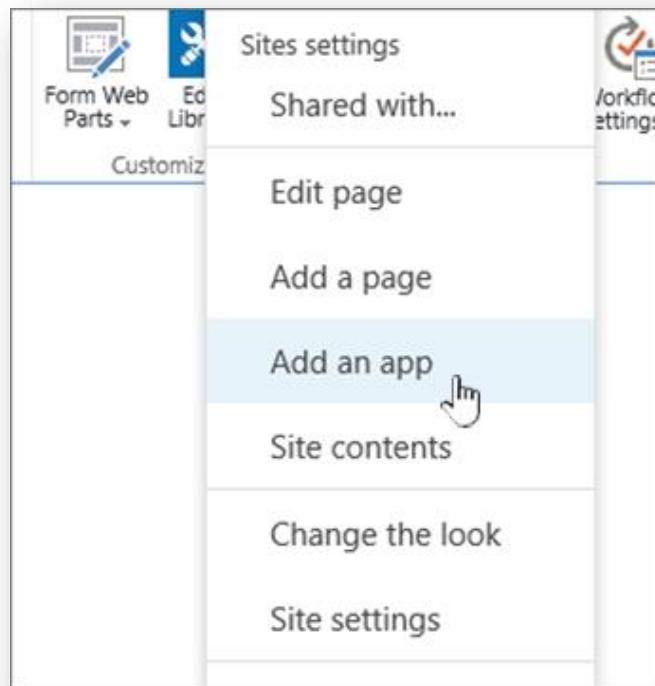
Başarı bir şekilde upload ettikten sonra “Deploy” butonuna tıklayınız. (Domain olarak SharePoint Online yazdığınıza dikkat ediniz.)



Tüm deploy işlemlerimiz bu kadar.

SharePoint Framework Uygulama Yükleme

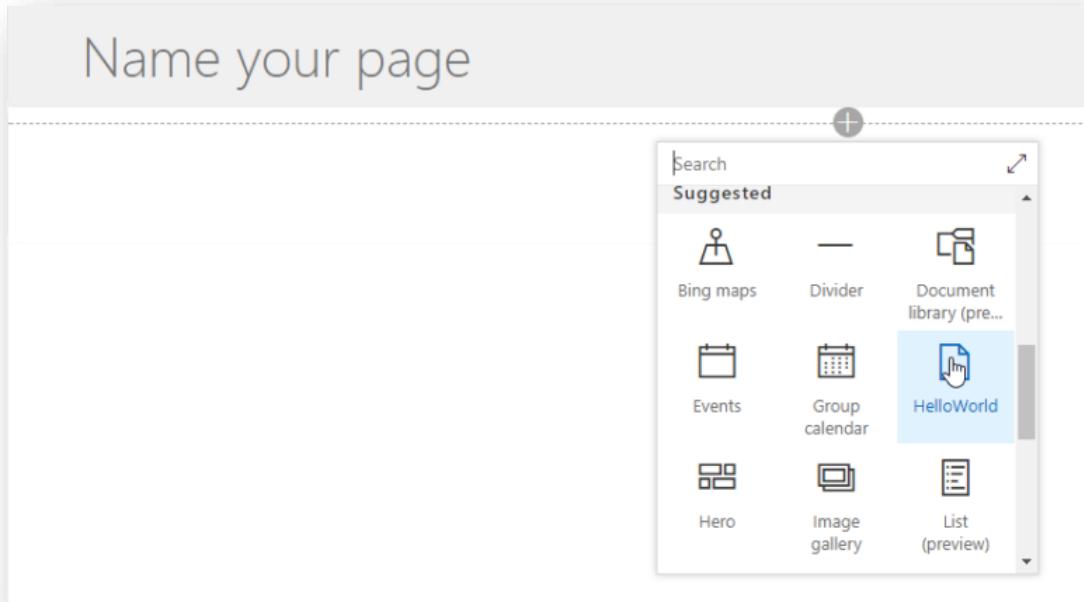
- ✓ Developer site ya da uygulamanızı kullanmak istediğiniz site collection ‘a gidiniz.
- ✓ Sağ üst nav bar üzerinde “Add an app” butonuna tıklayınız.



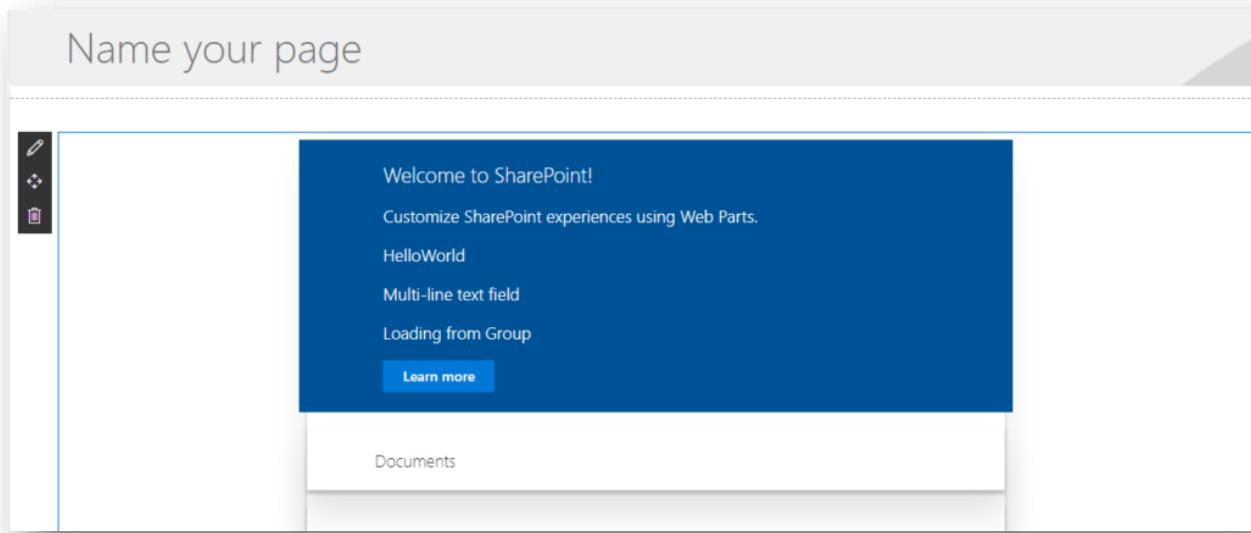
- ✓ Search box üzerinde kendi uygulamanızı aratıp, yükleyiniz.

A screenshot of the SharePoint 'Your Apps' page. The top navigation bar shows 'Office 365'. The main content area has a pink header with a 'G' icon and 'EDIT LINKS'. Below it, the text 'Site contents › Your Apps' is displayed. On the left, a sidebar lists 'Your Apps', 'Apps You Can Add' (with sub-options 'From Your Organization', 'Manage Licenses', 'Your Requests'), and 'SharePoint Store'. In the center, a search bar contains the text 'hello'. Below the search bar, the message '1 app matches your search' is shown, followed by 'Newest' and 'Name' filters. A single search result is listed: 'helloworld-webpart-client-side-solution' with an 'App Details' link. The entire interface is set against a light gray background.

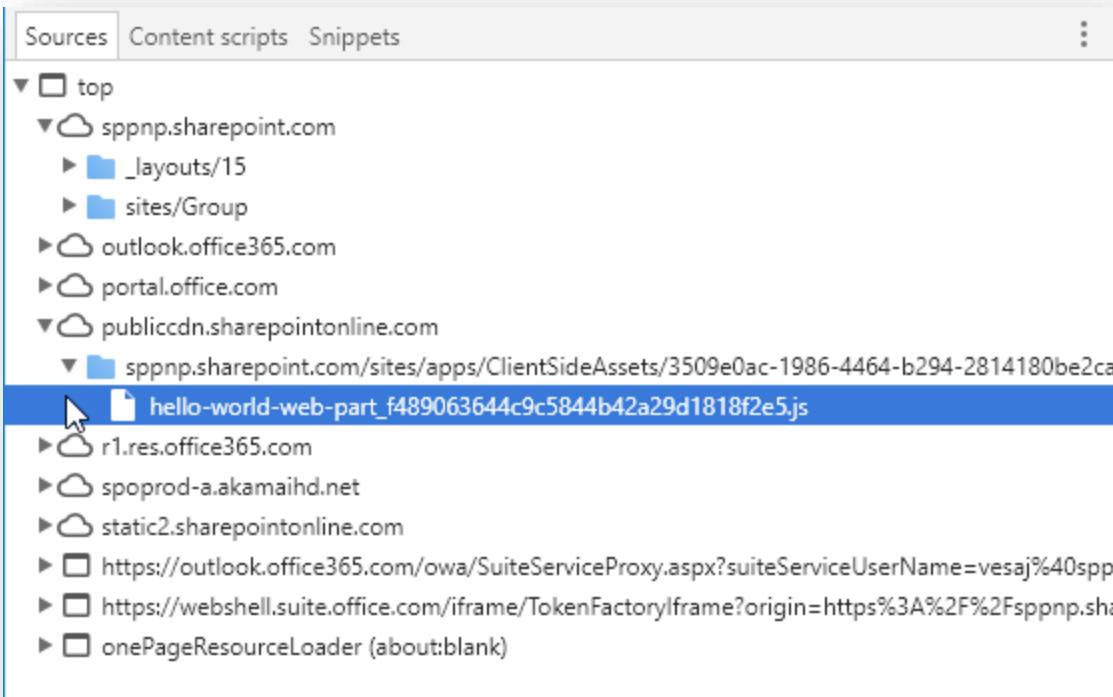
Uygulamanızı yükledikten sonra WebPart mevcut ise görüntülemek istediğiniz sayfayı açınız ve WebPart ekleyiniz.



Local üzerinde node.js çalışmasa bile uygulamanız görüntülenmeye devam edecektir.



F12 tuşuna basıp kontrol ettiğinizde “<http://publiccdn.sharepointonline.com>” adresi üzerinden yüklediğini göreceksiniz.



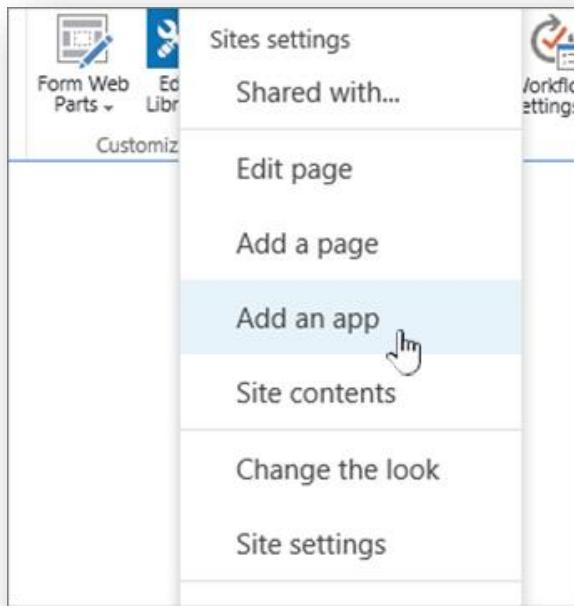
SharePoint Online içerisinde CDN aktif değilse ve “includeClientSideAssets” config içerisinde “true” olarak işaretliyse app catalog içerisinde yer alan “ClientSideAssets” içerisinde yüklenecektir. Bu durumda url: “<https://sppnp.microsoft.com/sites/apps/ClientSideAssets/>.” gibi olacaktır.

SharePoint Framework Deployment SharePoint Library

“.sppkg” paketini kontrol ettiğinizde içerisinde CSS, JavaScript ve diğer assets yer aldığılığını görüyor olacaksınız. Bu dosyalar genellikle CDN üzerinden dağıtım yapılmaktadır. (Azure CDN ya da Office 365 CDN gibi, bu yöntem ile farklı CDN ortamı içerisinde de veri alabilirsiniz.) Fakat SharePoint ‘e deploy yapmak için CDN kullanmak gibi bir zorunluluk yoktur. Şimdi CDN kullanmadan deploy yapabilmek için neler yapmalıyız bir göz atalım.

CDN Yerine Kullanacağınız Doküman Kütüphanesi Oluşturma

- ✓ Dosyaları yüklemek istediğiniz site açınız.
- ✓ Sağ top nav bar içerisinde yer alan “Add an app” butonuna tıklayınız.



- ✓ “Document Library” seçiniz.
- ✓ Permission üzerinde yetkileri düzenleyiniz. (Everyone yetki verebilirsiniz.)

SharePoint Library İçin Solution Package Konfigürasyonu

“config” folder içerisinde yer alan “package-solution.json” dosyasını açınız. “includeClientSideAssets” property “false” olarak güncelleyiniz. Bu durumda asset ‘leri paket içerisine dahil etmeyecektir. Çünkü biz SharePoint library içerisinde çekmesini istiyoruz.

```

EXPLORER
OPEN EDITORS
  package-solution.json config
SPX-DEPLOY-SPLIB
  .vscode
    config
      config.json
      copy-assets.json
      deploy-azure-storage.json
      package-solution.json
      serve.json
      write-manifests.json
  package-solution.json x
  {
    "$schema": "https://developer.microsoft.com/json-schemas/spfx-build/package-solution.schema.json",
    "solution": {
      "name": "spfx-deploy-splib-client-side-solution",
      "id": "e1a202e8-b4bf-4e6a-bddf-d86fc当地3023a",
      "version": "1.0.0.0",
      "includeClientSideAssets": true
    },
    "paths": {
      "zippedPackage": "solution/spfx-deploy-splib.sppkg"
    }
  }
  
```

CDN Path Güncelleme

“config” folder içerisinde yer alan “write-manifest.json” dosyasını açınız. “cdnbasePath” içerisinde oluşturmuş olduğunuz doküman kütüphanesinin url ile güncelleyiniz.

```

OPEN EDITORS 1 UNSAVED
• (1) write-manifests.json config
SPFX-DEPLOY-SPLIB
  .vscode
    config
      config.json
      copy-assets.json
      deploy-azure-storage.json
      package-solution.json
      serve.json
      write-manifests.json

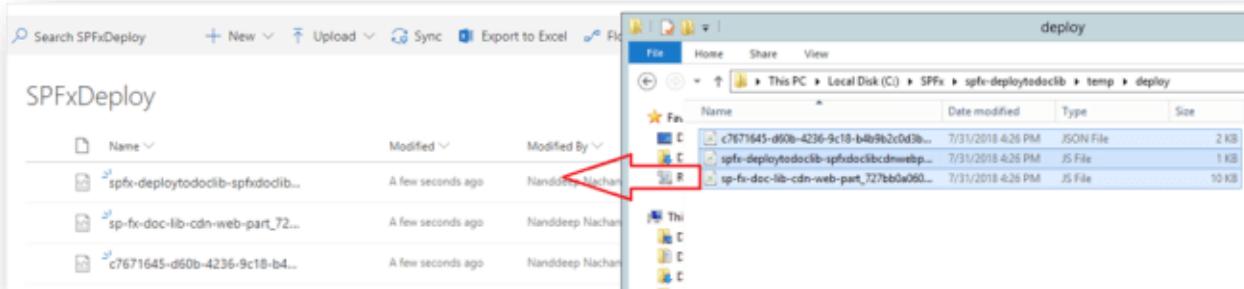
```

SharePoint Framework Deploy Süreci

Aşağıdaki komutu çalıştırınız. “gulp bundle –ship” solution ‘ı bundle etmenizi etmenizi sağlayacaktır.

```
gulp bundle --ship
```

“temp/deploy” folder içerisinde bundle edilmiş dosyaları, oluşturmuş olduğunuz kütüphaneye upload ediniz.

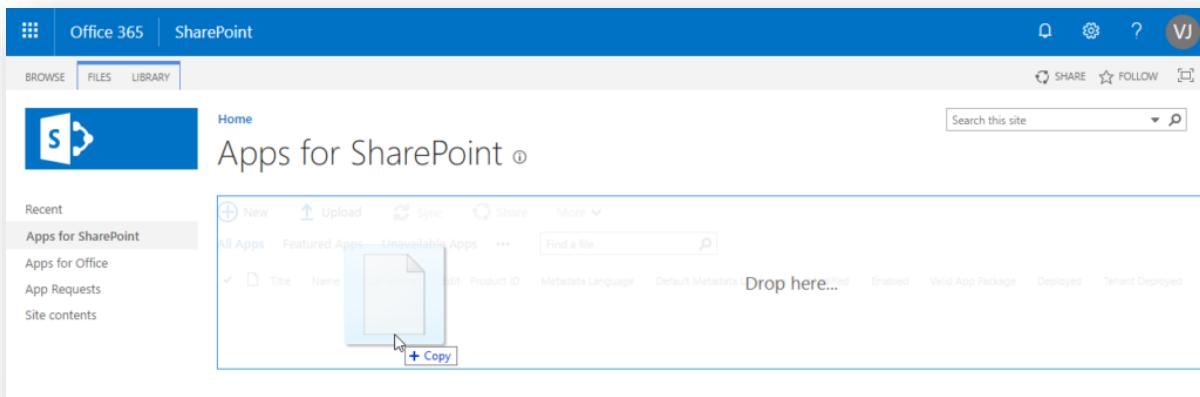


Solution ‘ı paket haline getirmek için aşağıdaki komutu çalıştırınız. Komut işlemini tamamlandıktan sonra config için belirtmiş olduğunuz path içerisinde oluşturmaktadır (solution klasörü).

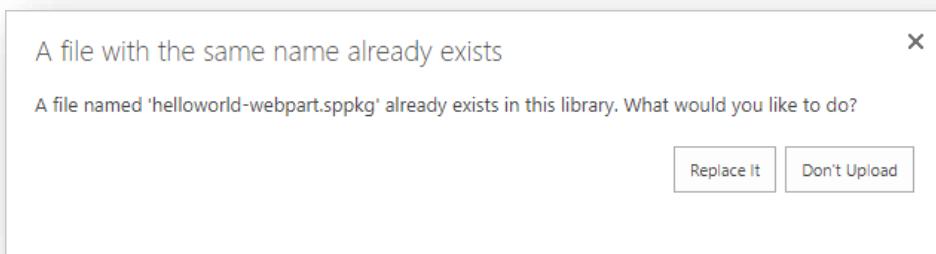
```
gulp package-solution --ship
```

Paket içerisinde görüntülemek isterseniz “solution/debug” klasörüne giderek, görüntüleyebilirsiniz.

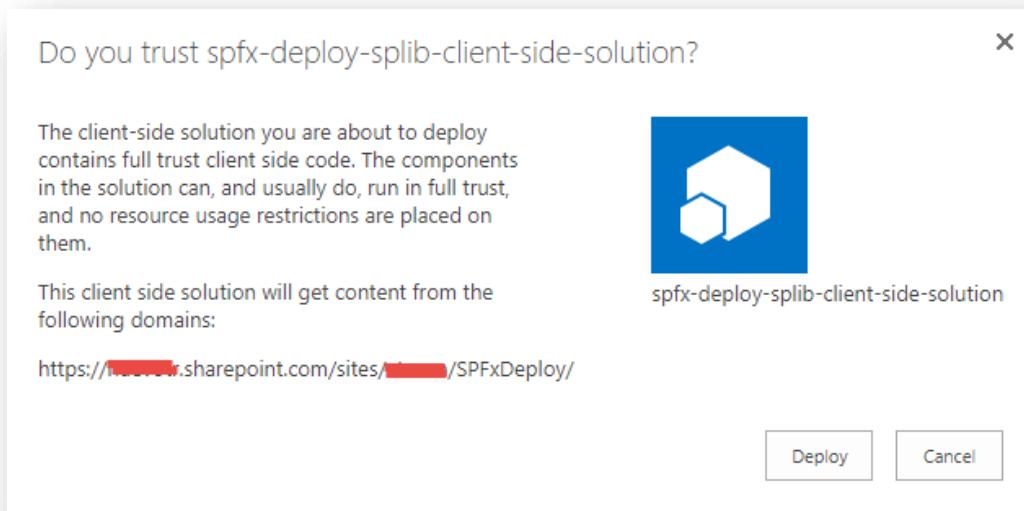
Paketleme işlemimiz bu kadar. Şimdi ilgili “.sppkg” dosyasını SharePoint App Katalog içerisinde yükleyiniz.



Eğer daha önceden aynı isimde bir app mevcut ise size değiştirip değiştirmeyeceğini soracaktır. "Replace it" butonuna tıklayarak uygulamayı upload edebilirsiniz.



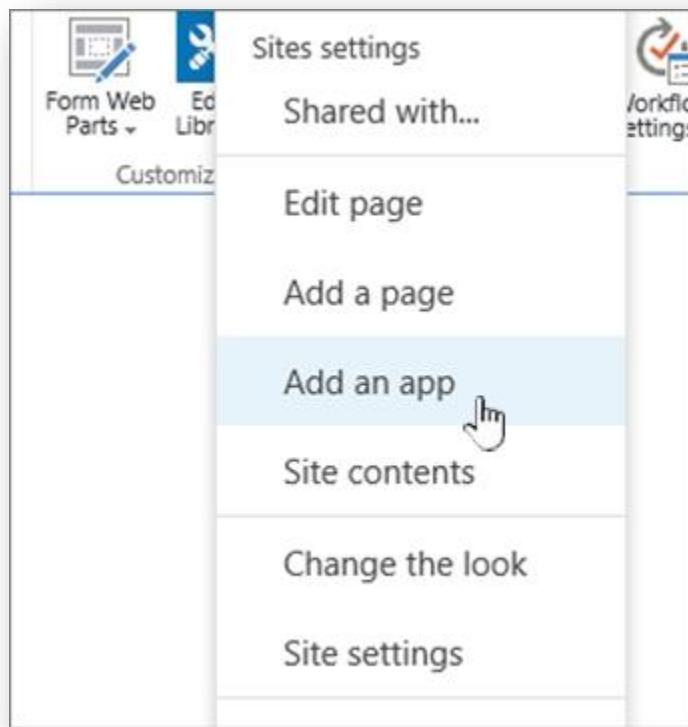
Başarı bir şekilde upload ettikten sonra "Deploy" butonuna tıklayınız. (domain olarak doküman kütüphanesi url olduğuna dikkat ediniz.)



Tüm deploy işlemlerimiz bu kadar.

SharePoint Framework Uygulama Yükleme

- ✓ Developer site ya da uygulamanızı kullanmak istediğiniz site collection 'a gidiniz.
- ✓ Sağ üst nav bar üzerinde "Add an app" butonuna tıklayınız.



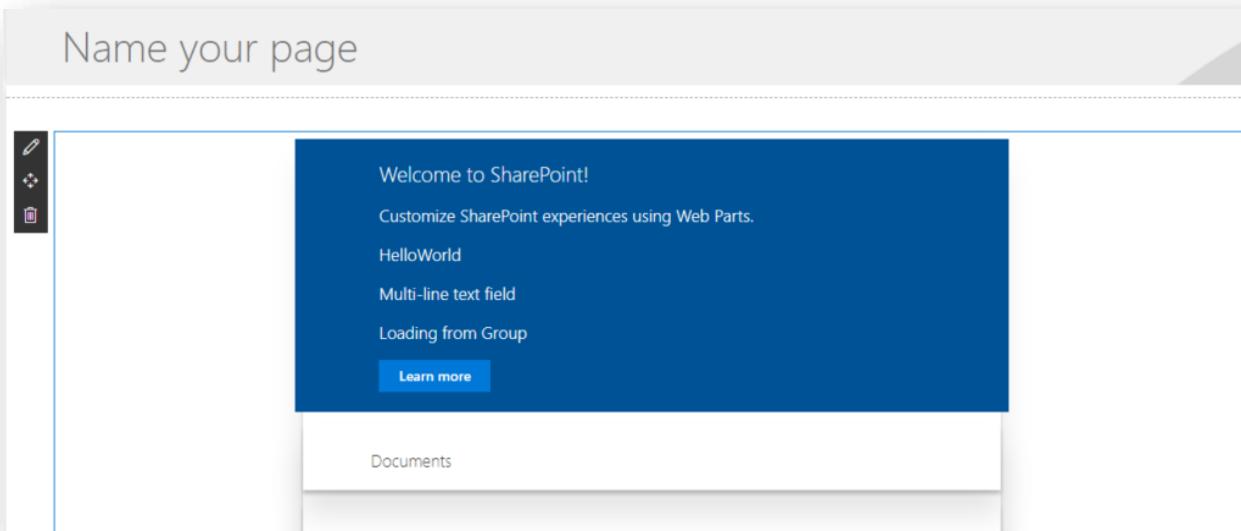
- ✓ Search box üzerinde kendi uygulamanızı aratıp, yükleyiniz.

The screenshot shows the 'Your Apps' section of the Office 365 portal. A search bar at the top contains the text 'hello'. Below it, a message says '1 app matches your search' with options to sort by 'Newest' or 'Name'. A single result is listed: 'helloworld-webpart-client-side-solution' with a blue icon. There is a link to 'App Details'.

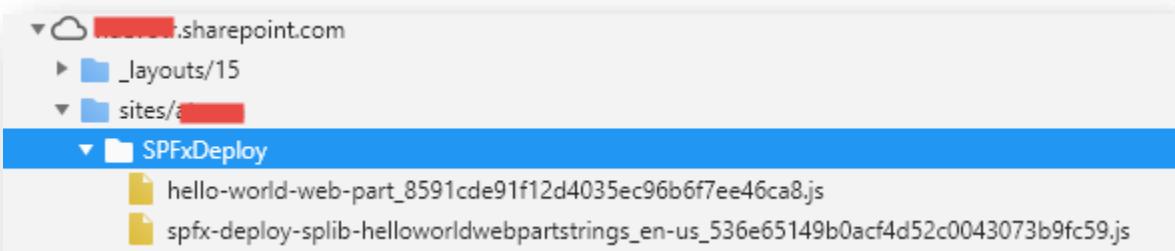
- ✓ Uygulamanızı yükledikten sonra WebPart mevcut ise görüntülemek istediğiniz sayfayı açınız ve WebPart ekleyiniz.

The screenshot shows a SharePoint page editor interface. A floating ribbon on the right side displays a grid of 'Suggested' web part icons. One icon, labeled 'HelloWorld' with a document icon, is highlighted with a light blue background. Other visible icons include 'Bing maps', 'Events', 'Hero', 'Divider', 'Group calendar', 'Image gallery', and 'List (preview)'. The main area of the screen has a placeholder text 'Name your page'.

- ✓ Local üzerinde node.js çalışmasa bile uygulamanız görüntülenmeye devam edecektir.



- ✓ F12 tuşuna basıp kontrol ettiğinizde doküman kütüphanesi üzerinden yüklediğini göreceksiniz.

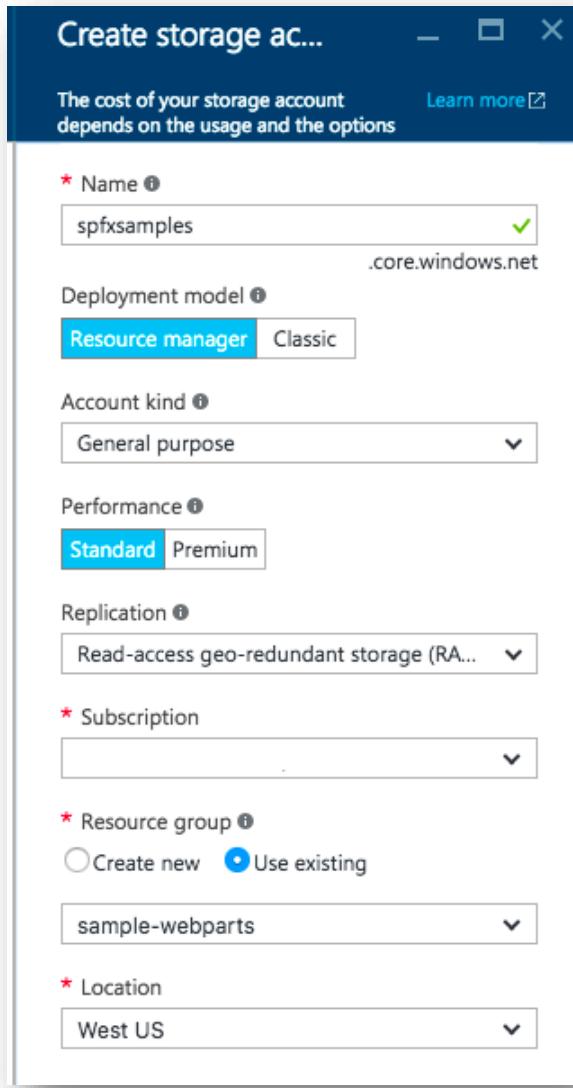


SharePoint Framework Deployment Azure CDN

SharePoint Framework ile Office 365 CDN deploy yapmak zorunda değilsiniz. Azure hesabınız mevcut ise Azure Content Delivery Network (Azure CDN) üzerine deploy yapabilirsiniz. Bunun için Azure Storage hesabınızın olması gerekmektedir. SharePoint Framework otomatik olarak destek sağlar fakat manuel olarak upload yapabilirsiniz.

Azure Storage Account Konfigürasyonu

İlk olarak Azure storage account oluşturmanız gerekmektedir. Detay için [linke](#) tıklayınız.



Hesabınızı oluşturduktan sonra “<http://spfxsamples.blob.core.windows.net>” gibi endpoint oluşacaktır.

SharePoint Framework projeleriniz için benzersiz bir storage account oluşturmanız gerekmektedir.

BLOB Container Oluşturma

Oluşturmuş olduğunuz Azure storage account içerisinde yer alan “+ Container” butonuna tıklarak yeni BLOB Container oluşturunuz.

The screenshot shows the Azure Storage account dashboard for the account 'spfxsamples'. The left pane displays 'Essentials' information: Resource group 'sample-webparts', Status 'Primary: Available, Secondary: Available', Location 'West US, East US', and Subscription details. The right pane shows the 'Blob service' blade for 'spfxsamples'. It includes a search bar, a table for containers (empty), and a message: 'You don't have any containers yet. Click '+ Container' to start creating one.' Red annotations include a circled '1' over the 'Blobs' tile in the services section, and a circled '2' over the 'Container' button in the top right.

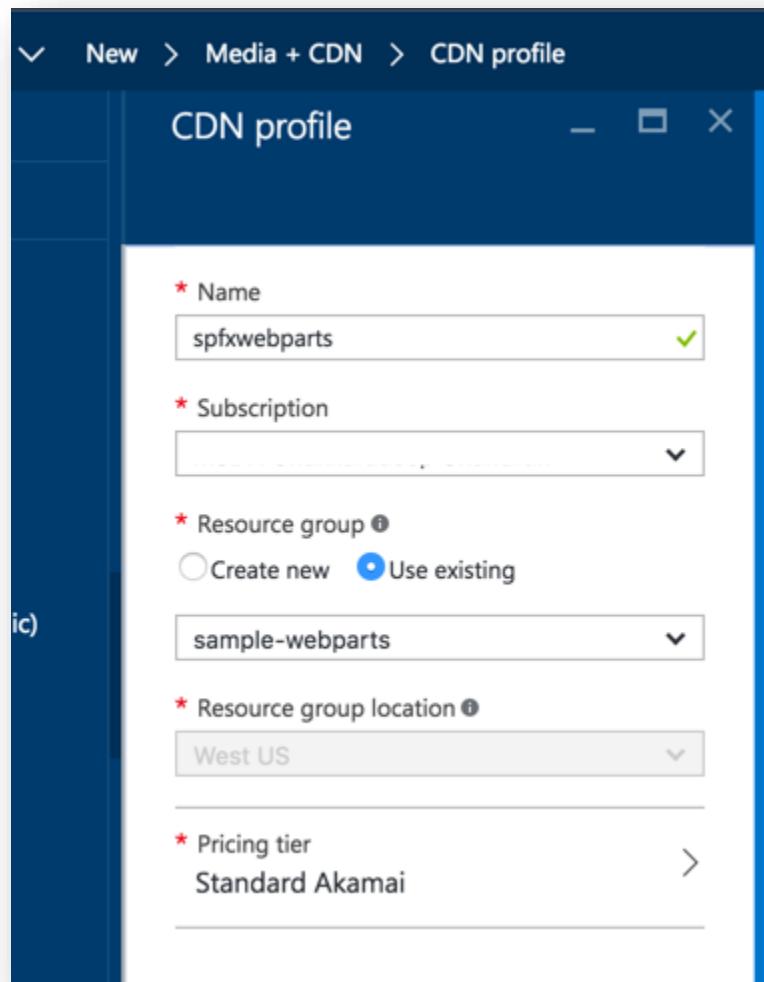
Storage Account Access Key

Storage account dashboard içerisinde yer alan “access key” seçip, sonrasında keylerden herhangi birisini kopyalayınız.

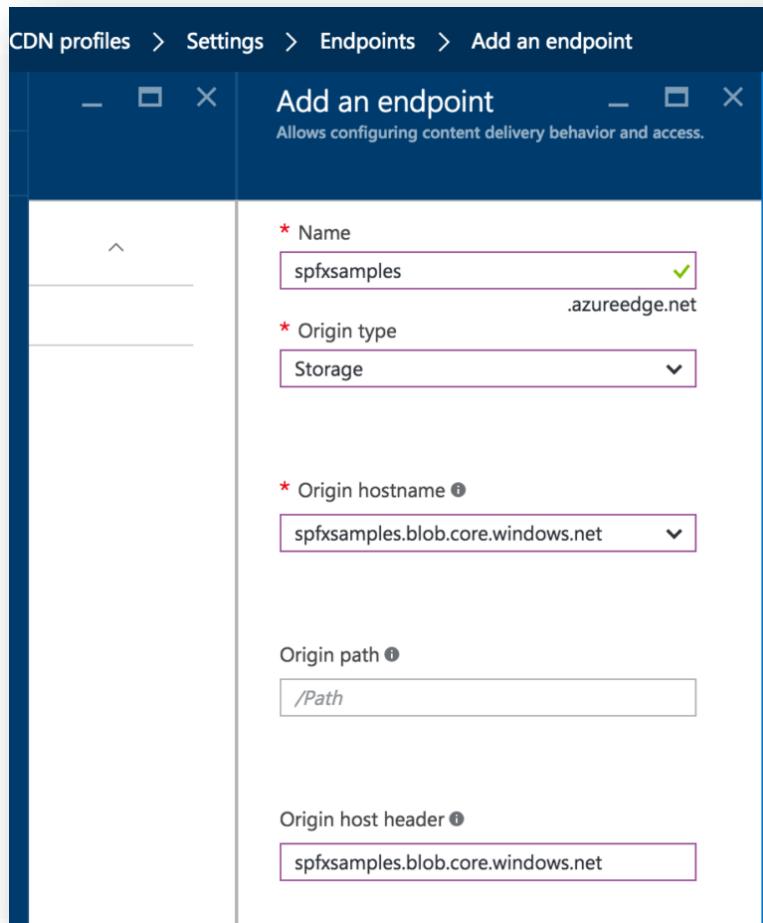
The screenshot shows the 'Access keys' blade for the storage account 'spfxsamples'. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and a Settings section with 'Access keys' highlighted (circled with a red '1'). The main area contains a search bar, a paragraph about access keys, and a table listing two access keys: 'key1' and 'key2'. Each key has a copy icon (circled with a red '2') next to its respective row. The storage account name 'spfxsamples' is also visible at the top.

CDN Profile ve Endpoint

Yeni bir CDN profile oluşturup sonrasında BLOB container ile ilişkilendiriniz. Azure storage içerisinde CDN etkinleştirip, yeni bir CDN profil oluşturunuz. Detay için [linke](#) tıklayınız.



CDN profilinizi oluşturduktan sonra endpoint oluşturmanız gerekmektedir. Detay için [linke](#) tıklayınız.



Blog container ile ilişkilendirdiğimizden dolayı
“<http://spfxsamples.azureedge.net/azurehosted-webpart/>” endpoint oluşacaktır.

Solution Package Konfigürasyonu

“config” folder içerisinde yer alan “package-solution.json” dosyasını açınız. “includeClientSideAssets” property “false” olarak güncelleyiniz. Bu durumda asset ‘leri paket içerisine dahil etmeyecektir. Çünkü biz Azure CDN üzerinden çekmesini istiyoruz.

```
{  
  "$schema": "https://developer.microsoft.com/json-schemas/spfx-build/package-solution  
.schema.json",  
  "solution": {  
    "name": "azurehosted-webpart-client-side-solution",  
    "id": "a4e95ed1-d096-4573-8a57-d0cc3b52da6a",  
    "version": "1.0.0",  
    "manifest": "manifest.json",  
    "files": [{"path": "index.html", "size": 123456789}, {"path": "script.js", "size": 987654321}],  
    "dependencies": [{"name": "nodejs", "version": "14.17.0"}, {"name": "spfx-build", "version": "1.1.0"}]  
  }  
}
```

```
    "version": "1.0.0.0",
    "includeClientSideAssets": false,
},
"paths": {
    "zippedPackage": "solution/azurehosted-webpart.sppkg"
}
}
```

Azure Storage Account Konfigürasyonu

“config” folder içerisinde yer alan “deploy-azure-storage.json” dosyayı açınız.

```
{
    "$schema": "https://dev.office.com/json-schemas/spfx-build/deploy-azure-storage.schema.json",
    "workingDir": "./temp/deploy/",
    "account": "<!-- STORAGE ACCOUNT NAME -->",
    "container": "azurehosted-webpart",
    "accessKey": "<!-- ACCESS KEY -->"
}
```

“account, container, accessKey” ayarlarınızı, oluşturmuş olduğunuz Azure storage account içerisinde yer alan bilgilerle güncelleyiniz.

“workingDir” ise assets ‘in bulunduğu path adıdır.

```
{
    "workingDir": "./temp/deploy/",
    "account": "spfxsamples",
    "container": "azurehosted-webpart",
    "accessKey": "q1UsGwocj+Cn1Luv9Zpri0Cj46ikgBbDBCaQ0FfE8+qKVbDTVSbRGj41avlG73rynbvKiz
ZpIKK9XpnpA=="
}
```

CDN Path Güncelleme

“config” folder içerisinde yer alan “write-manifest.json” dosyasını açınız. “cdnBasePath” içerisinde oluşturmuş olduğunuz Azure CDN url ile güncelleyiniz.

```
{  
  "cdnbasePath": "https://spfxsamples.azureedge.net/azurehosted-webpart/"  
}
```

SharePoint Framework Deploy Süreci

Aşağıdaki komutu çalıştırınız. “gulp bundle –ship” solution ‘ı bundle etmenizi sağlayacaktır..

```
gulp bundle --ship
```

Sonra aşağıdaki kodu çalıştırarak CDN ‘e “temp/deploy” içerisinde yer alan CSS, JS gibi assets yükleyiniz.

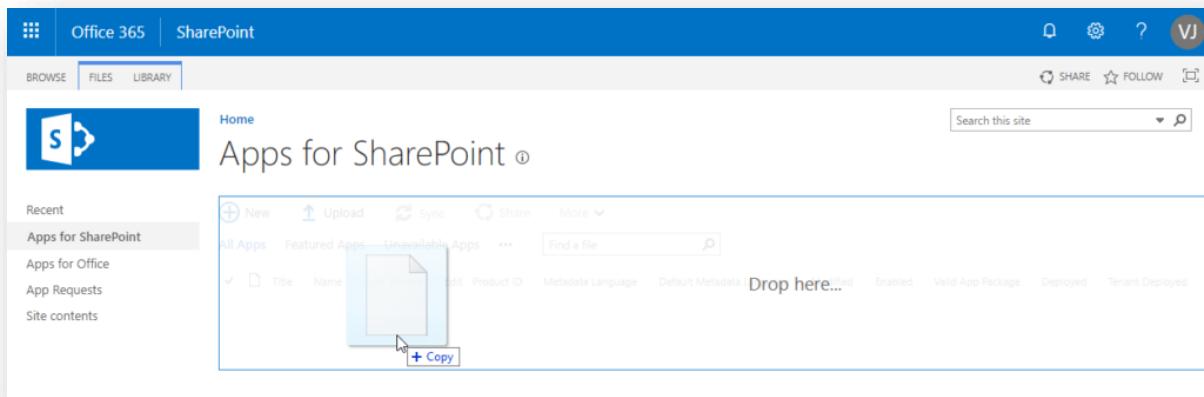
```
gulp deploy-azure-storage
```

Solution ‘ı paket haline getirmek için aşağıdaki komutu çalıştırınız. Komut işlemini tamamladıktan sonra config için belirtmiş olduğunuz path içerisinde oluşturmaktadır. (solution klasörü)

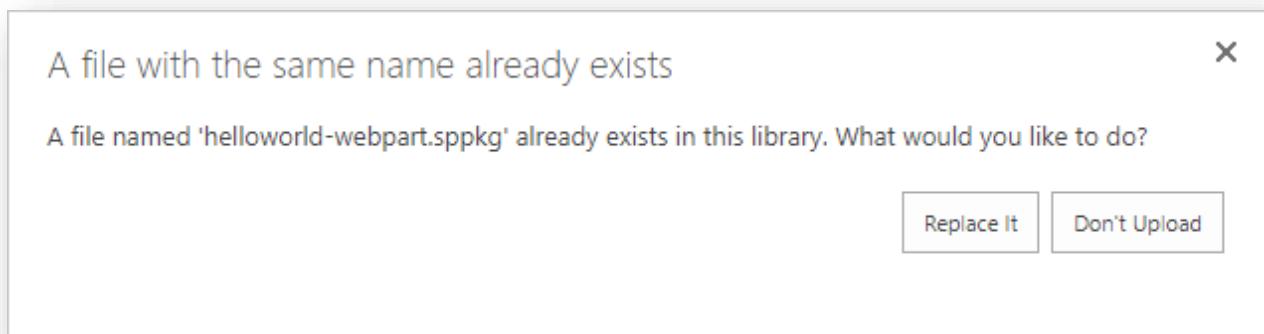
```
gulp package-solution --ship
```

Paket içerisinde görüntülemek isterseniz “solution/debug” klasörüne gidip, görüntüleyebilirsiniz.

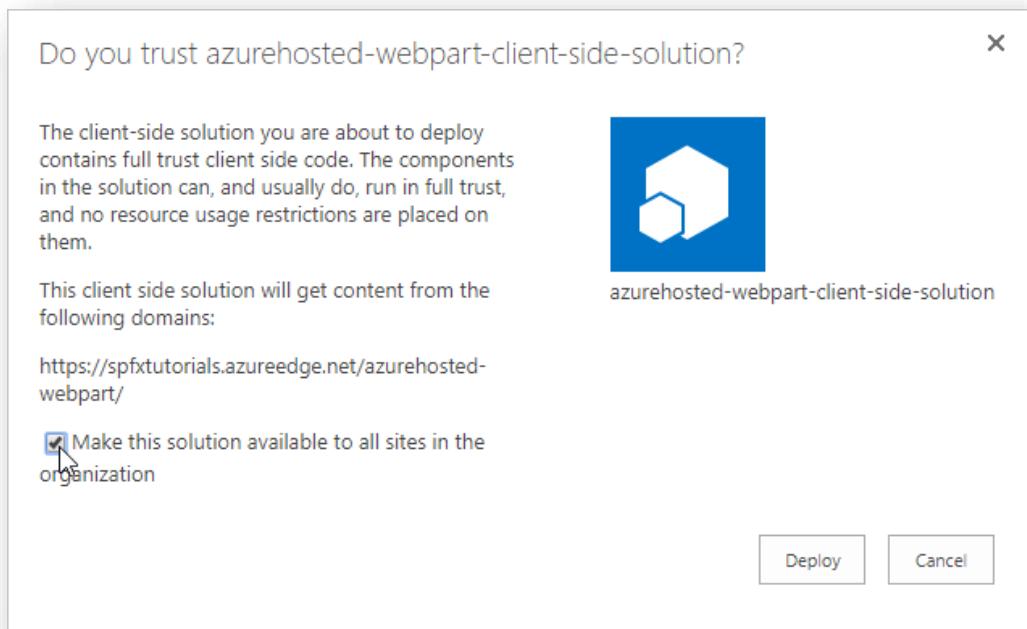
Paketleme işlemimiz bu kadar. Şimdi ilgili “.sppkg” dosyasını SharePoint App Katalog içerisinde yükleyiniz.



Eğer daha önceden aynı isimde bir app mevcut ise size değiştirip değiştirmeyeceğini soracaktır. “Replace it” butonuna tıklayarak uygulamayı upload edebilirsiniz.

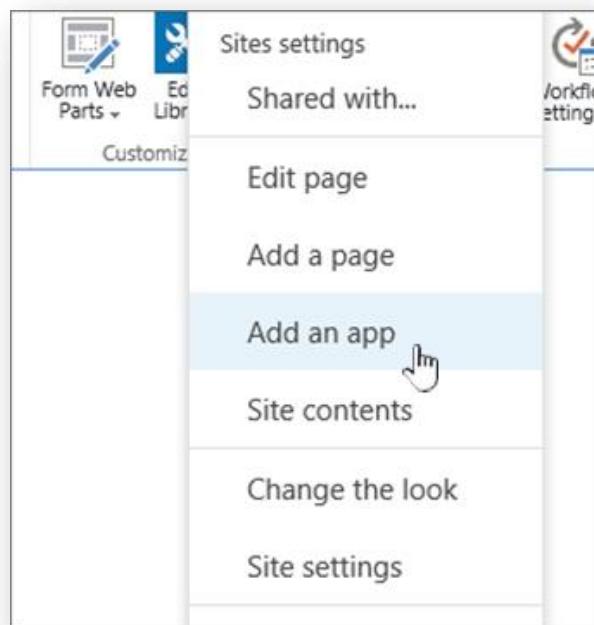


Başarı ile upload ettikten sonra “Deploy” butonuna tıklayınız. (domain olarak Azure CDN url olduğuna dikkat ediniz.)

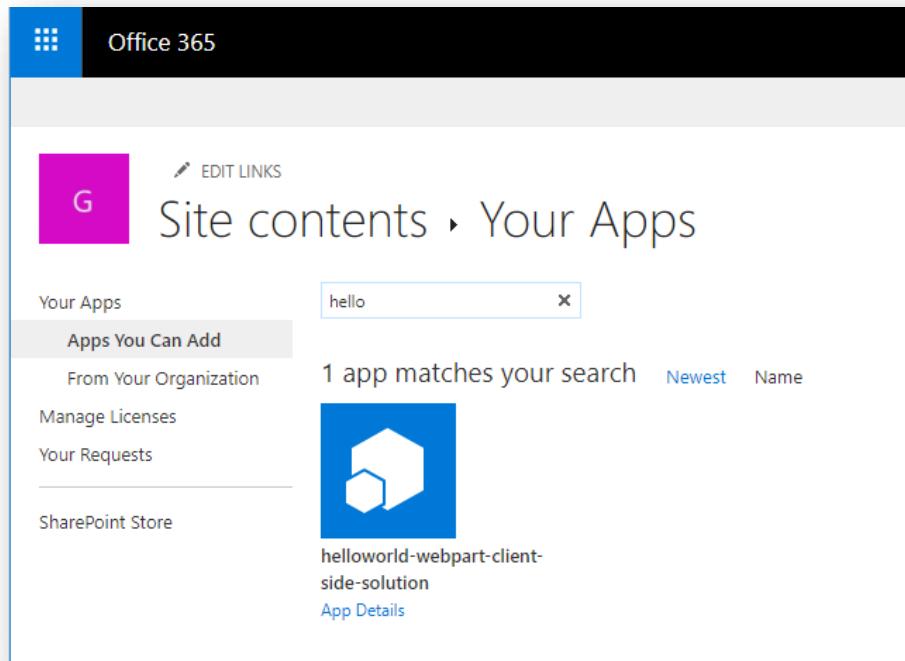


SharePoint Framework Uygulama Yükleme

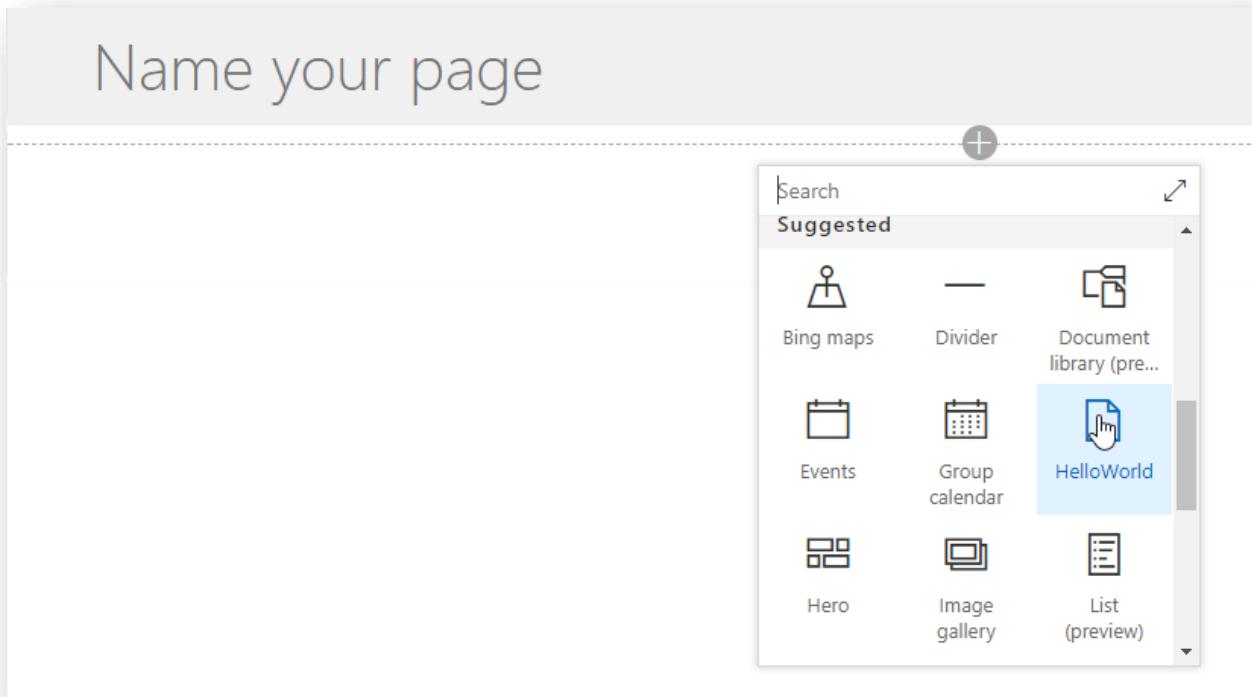
- ✓ Developer site ya da uygulamanızı kullanmak istediğiniz site collection 'a gidiniz.
- ✓ Sağ üst nav bar üzerinde “Add an app” butonuna tıklayınız.



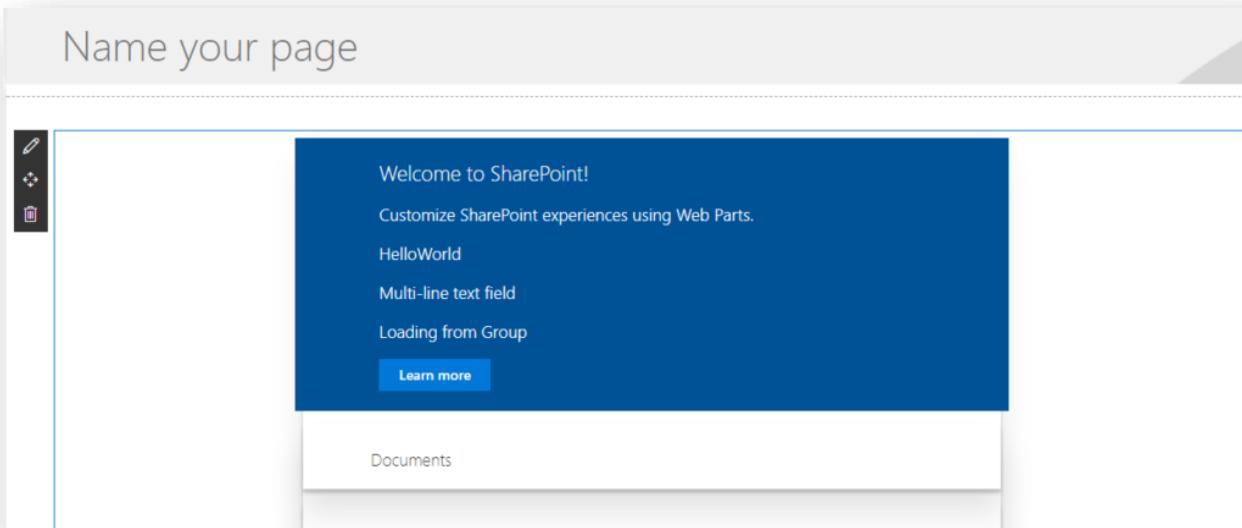
- ✓ Search box üzerinde kendi uygulamanızı aratıp, yükleyiniz.



- ✓ Uygulamanızı yükledikten sonra WebPart mevcut ise görüntülemek istediğiniz sayfayı açınız ve WebPart ekleyiniz.



Local üzerinde node.js çalışmasa bile uygulamanız görüntülenmeye devam edecektir.



F12 tuşuna basıp kontrol ettiğinizde Azure CDN üzerinden yüklediğini göreceksiniz.

