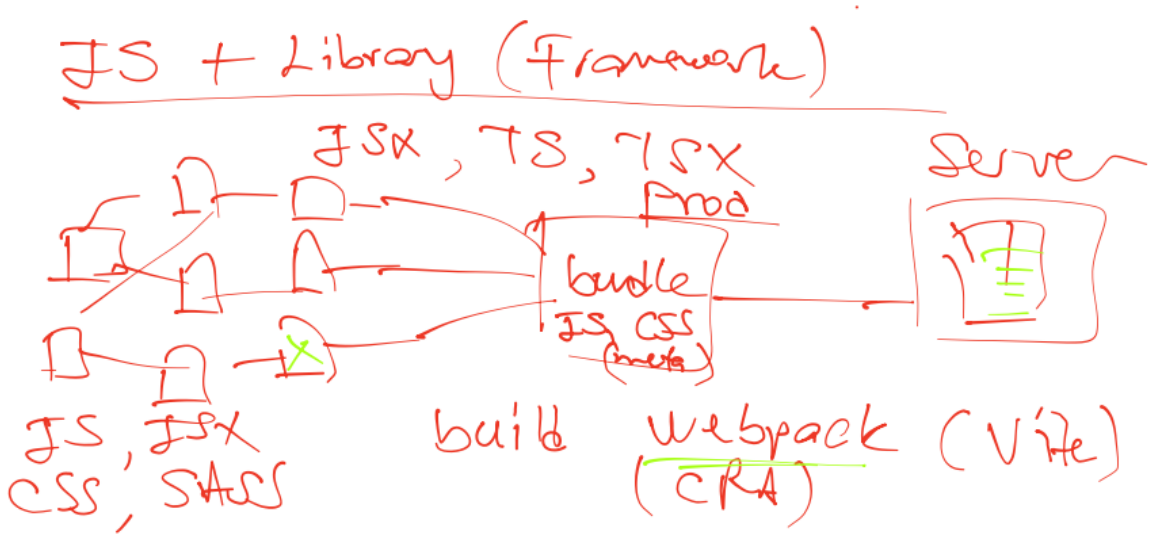
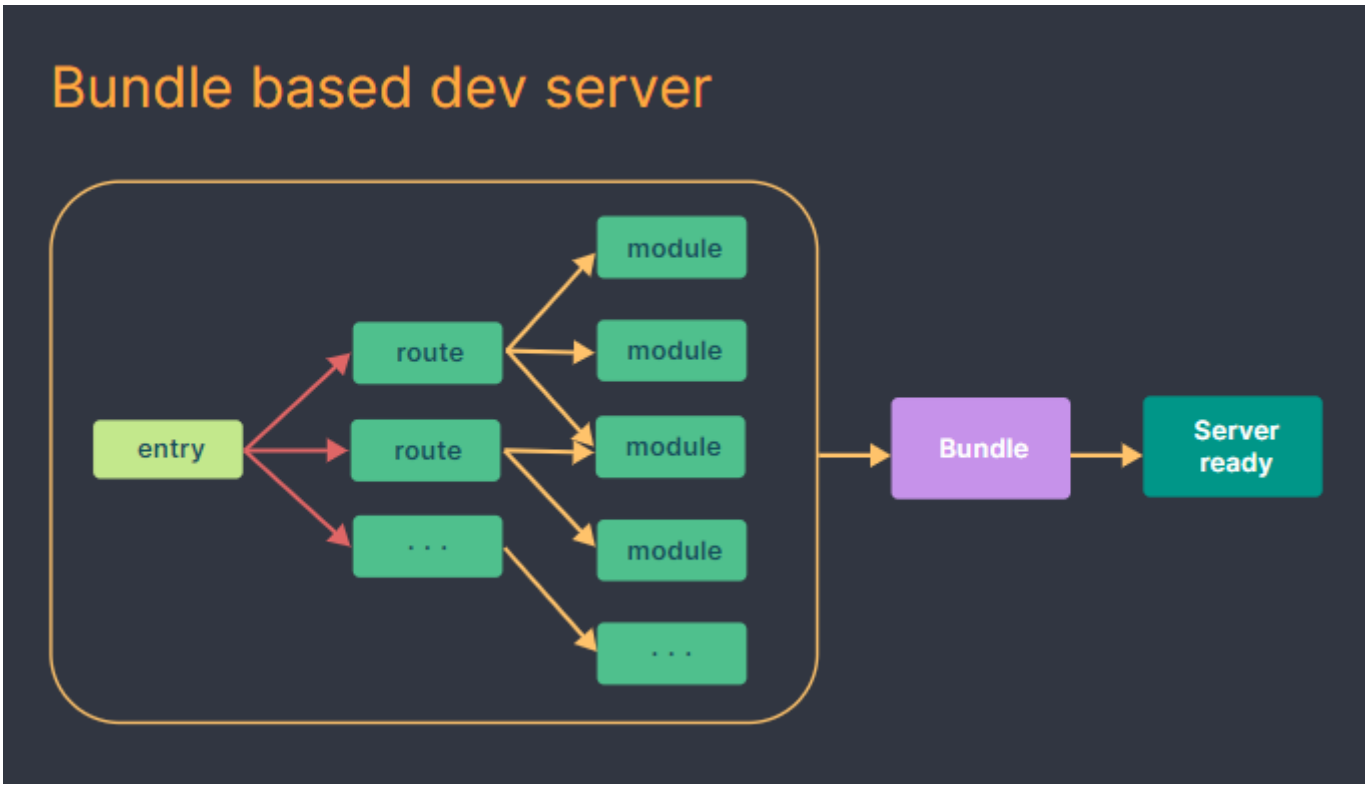
Vite

- Vite=> Yeni Nesil Frontend Toolu. Yani derleme toolu.(Next Generation Frontend Tool)
- Building Tools => webpack(cra), parcel, vite vb.

Javascript + Library(Framework) ile yazdığımız projelerde moduller üzerinden ilerliyoruz. React da yazdığımız componentleri birer modül olarak düşünebiliriz. JSX-TSX-SCSS-CSS bunları servera yansıtabilmek için bundle işlemi gerçekleşiyor. Yani yazdığımız kodlar birleştiriliyor. Ondan sonra servera sunuluyor.



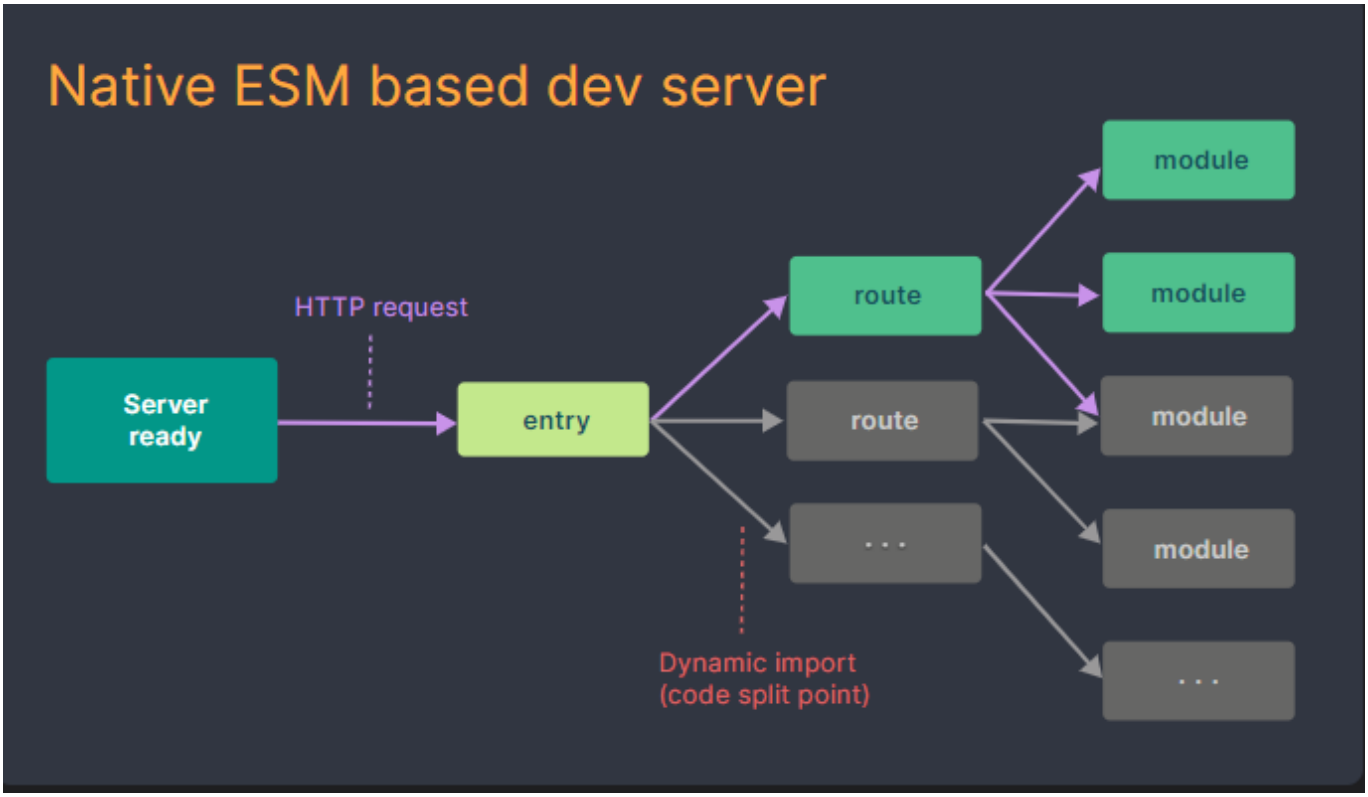


Şimdiye kadar biz bu bundle işlemini webpack ile yapıyorduk. Vite da aslında aynı işlemi gerçekleştiriyor ama izlediği yol farklı ve daha hızlı.

Vite neden hızlı?

Örneğin; webpack ile bir module(component) üzerinde değişiklik yaptığımızda webpack tüm modulleri alıyor tekrardan hepsini bundle ediyor ve servera son halini sunuyor. Küçük projelerde bu işlem çok ciddi bir zaman almıyor ama proje büyüdüğünde,moduller,dependencyler arttığında bu işlem uzun sürüyor. Mui örneğinde de bunu görüyoruz🙂.

Vite ise webpackden farklı olarak **Native Module** yapısını kullanıyor. Javscript ES2020 ile brlikte popüler browserlar(chrome-safari gibi) artık Native Module yapısını desteklemeye başladı. Yani artık bizim yazdığımız moduller doğrudan ayrı ayrı browserlara sunulabilir hale gelmiş oldu. Vite de bu native yapıyı kullanarak artık modulleri doğrudan browserlara sunuyor. Webpack gibi development aşamasında sürekli bundle işlemi gerçekleştirmiyor. Sadece değişikliğin gerçekleştiği module e odaklanıp değişikliği daha hızlı ekrana yansıtmış oluyor. Tabi ki prod aşamasında değişen bir durum yok, aynı webpackte olduğu gibi bundle edip build alıyor. Daha detaylı bilgiler için dökümanda Why Vite? kısmına bakabilirsiniz.



Start Project Vite

```
Command => pnpm create vite or yarn create vite or npm create vite@latest
Framework => React
Variant => Javscript or Typescript

# or with template
=> for javascript:
yarn create vite todoapp --template react
npm create vite todoapp --template react
pnpm create vite todoapp --template react
=> for typescript:
yarn create vite todoapp --template react-ts
npm create vite todoapp --template react-ts
pnpm create vite todoapp --template react-ts
```

Start Server

```
cd todoapp
=> add dependencies
for npm => npm install
for yarn => yarn install
for pnpm => pnpm install
=> run server
for npm => npm run dev
for yarn => yarn dev
for pnpm => pnpm dev
```

Vite Terminal Commands

```
press r to restart the server
press u to show server url
press o to open in browser
press c to clear console
press q to quit
```

Vite Features

1. Hot Module Replacement (HMR)

   Vite, native ESM (ECMAScript Modülleri) üzerinden Hot Module Replacement (HMR) desteği sunar. Vue ve React gibi kütüphaneler için entegre bir destek sağlar.

2. TypeScript

   - Vite ile hemen .ts uzantılı dosyaları kullanmaya başlayabilirsiniz.
   - Sadece modül aktarımını gerçekleştirir ve tip kontrolünü size bırakır. Bu, yapılandırmanıza bağlıdır.
   - Vite, TypeScript derlemesi için esbuild'i kullanır ve bu, tsc'ye göre 20-30 kat daha hızlıdır.

3. PostCSS

   - Projenizde bir PostCSS yapılandırma dosyası varsa, bu yapılandırma otomatik olarak tüm CSS kodlarına uygulanır.

4. CSS Modülleri

   - Dosyaların adları .module.css ile bitiyorsa, CSS Modülleri kullanımına izin verir.

5. CSS Önişlemcileri (CSS Pre-Processors)

   -Vite, scss, sass, less, style, stylus dosyaları için destek sunar. Ayrıca, yukarıda bahsedilen önişlemcilere CSS Modül yapısını da uygulayabilirsiniz.

6. Statik Varlıklar (Static Assets)
    - JSON dosyaları doğrudan içe aktarılabilir ve named export kullanabilirsiniz.
7. Ortam Değişkenleri (Environment)
    - Vite projelerinde, import.meta.env kullanarak çevresel değişkenlere erişebilirsiniz.
    - Üretim aşamasında, çevresel değişkenler kod içinde statik olarak değiştirilir ve bunun için Vite dotenv paketini kullanır.
8. Sunucu Tarafı İşleme (Server Side Rendering - SSR)
    - Vite, kendi içinde Sunucu Tarafı İşleme (SSR) desteği sunar. Ayrıntılar dokümantasyonda mevcuttur.
9. Hangi Çalışma Çerçevelerini Kullanabiliriz?
    - Vite ile Vue, React, Preact, Lit, Svelte gibi çeşitli çalışma çerçeveleri kullanarak geliştirme yapabilirsiniz.

Ortam Değişkenleri (Environment)

Vite, çevresel değişkenlere erişmek için dotenv paketini kullanır. Projede import.meta.env kullanarak çevresel değişkenlere erişebilirsiniz. İşte bazı örnek değişkenler:

- import.meta.env.MODE: Uygulamanın çalışma modunu içerir (production, development...).
- import.meta.env.PROD: Uygulama production modunda ise true döndürür.
- import.meta.env.DEV: Uygulama development modunda ise true döndürür.
- import.meta.env.SSR: Uygulama sunucu tarafında çalışıyorsa true döndürür.
- .env dosyalarını uygulamanın çalışma moduna göre kullanabilirsiniz:
    - .env: Her durumda yüklenir.
    - .env.[mode]: Örneğin, .env.production sadece Production modunda kullanılır. Not: .env dosyanızda TITLE adında bir çevresel değişken tutarsanız, Vite bunu görmeyecektir. Bunun yerine VITE_APP_TITLE kullanmalısınız. Bu sorunu çözmek için Vite config dosyanızda envPrefix değerini değiştirmeniz gerekmektedir.

1. env dosyalarinda genelde tum uygulama icerisinde tekrar tekrar kullanilacak degiskenler (API sifreleri, URL'ler vb.) saklanilir. Dolayisiyla bir kere tanimlayip tum proje icersinde erisebilmek tekrari azaltmak adina avantaj saglar.

2. .env degiskenlerine her asamada (dev,prod,test) erisilebilir. Bu dosya github'a pushlandigi icin gizli bilgiler paylasmak dogru olmaz.

3. VITE de guvenlik nedeniyle degisken isimlerinin basina VITE_ kelimesi eklenmelidir.

4. bir env degiskenini okumak icin `import.meta.env.degiskenAdi` seklinde erisim yapilir.

```
VITE_BASE_URL = "https://10001.fullstack.clarusway.com"
```

5. .env.local degiskenlerine her asamada erisilebilir. local uzantili dosyalar github'a pushlanmaz.

```
VITE_API_KEY = "dsfafare11323"
```

6. .env.production.local degiskenlerine sadece production asamasinda yani canliya alindigi durumda erisilebilir.

```
VITE_API_KEY_PROD = "32rgdrgr2132"
```

Not: Vite da .local uzantılı dosyalar git takibine girmezler. Default olarak .gitignore dosyasında `*.local` şeklinde gelir. Dolayısıyla .env.local dosyasında yazdığınız değişkenler githuba pushlanmaz. Diğer .env dosyalarını githuba pushlamak istemiyorsanız onları da .gitignore dosyasına manuel olarak eklemeniz gerekir.

Vite Config Ayarları

```
// Default config dosyası
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
})

// build klasör adı değiştirme
export default defineConfig({
  plugins: [react()],
  build:{outDir:'./build'},
})

// port değiştirme
export default defineConfig({
  plugins: [react()],
  server: {
    port: 3000,
  },
})

// .env prefix değiştirme
export default defineConfig({
  plugins: [react()],
  envPrefix: 'CUSTOM_',
})
```

Daha detaylı bilgiler için dökümana bakabilirsiniz.