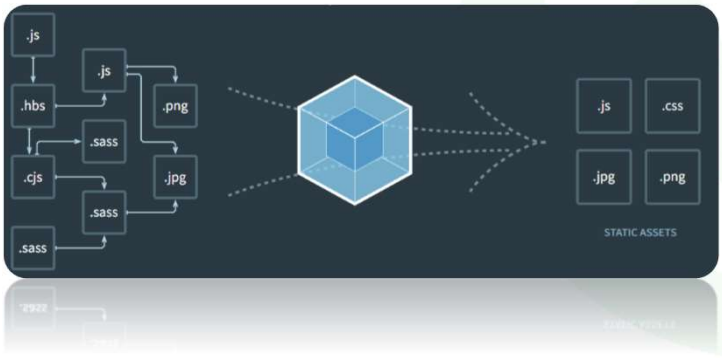

Webpack

Fundamentals

- Webpack ne işe yarar.
- Dokümantasyon
- Installing webpack
- Webpack.config
- Development & production mode

Webpack

Javascript modüllerini birleştirip tek bir bundle dosyası oluşturur



Webpack

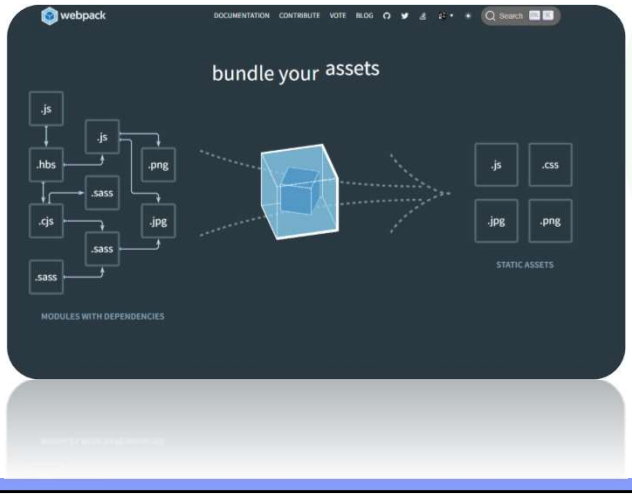
Dependency management

3th party library

Development server

Code splitting

Dokümantasyon



webpack.js.org

Installing webpack

package.json dosyasını oluşturur.

Paketleri development tarafı için kurar

```
npm init -y
npm i --save-dev webpack webpack-cli
```

```
"scripts": {
   "start": "webpack"
}
```

package.json

webpack.config

```
const path = require("path");
module.exports = {
    entry:"./src/index.js",
    output: {
        filename: "main.js",
        path: path.resolve(__dirname, "dist")
    }
}
```

Dosya ve klasör yollarına erişmek için kullanılan bir library

Webpack in başlangıç dosyası

Webpack in bundle dosyası

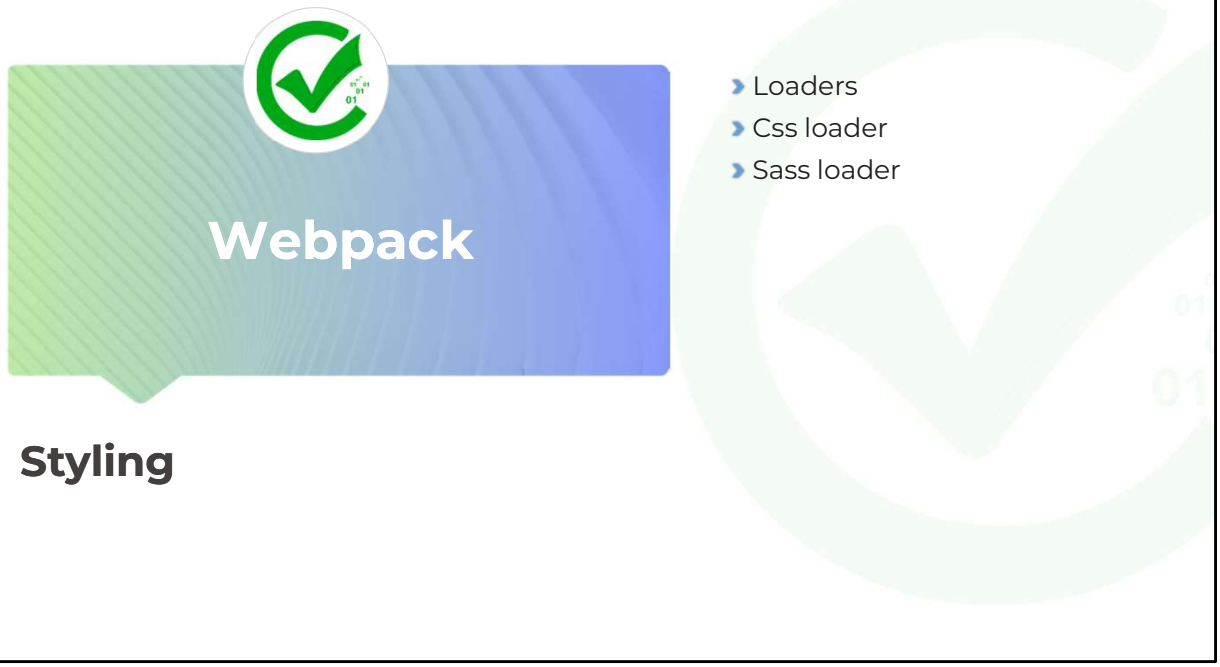
Webpack in bundle dosyasının bulunacağı klasör

Development & Production

Webpack development ve production süreçlerini farklı ayarlarla yürütebilir. Bunun için mode parametresi bulunmaktadır.

```
module.exports = {
    mode : "development",
    entry: "./src/index.js"
    …
}
```

Default değeri production dır.


Webpack
Styling
Loaders
Css loader
Sass loader

Loaders

«Loader» lar, dosyaları bir dilden başka bir dile dönüştürmeyi sağlayan yapılardır.

loader

https://webpack.js.org/loaders/

CSS Loader

CSS dosyalarının alınıp işlenmesini sağlayan loader tipidir.

```
npm i css-loader style-loader --save-dev
```

```
module: {
   rules: [
      {
         test: /\.css$/,
         use: [
          'style-loader',
          'css-loader']
      }
   ]
}
```

Use içinde kullanılan loader lar ters sırada çalıştırılır.

Css kodlarını DOM a yerlerştirir.

İmport edilen css dosyalarındaki css kodlarını alır.

SASS Loader

SASS (.sass ve .scss) dosyalarının alınıp işlenmesini sağlayan loader tipidir.

```
npm i sass-loader node-sass --save-dev
```

```
module: {
   rules: [
      {
         test: /\.s(a|c)ss$/,
         use: [
             'style-loader',
             'css-loader',
             'sass-loader'
         ]
      }
   ]
}
```

Use içinde kullanılan loader lar ters sırada çalıştırılır.

Css kodlarını DOM a yerlerştirir.

css dosyaları yükler ve css kodlarını alır.

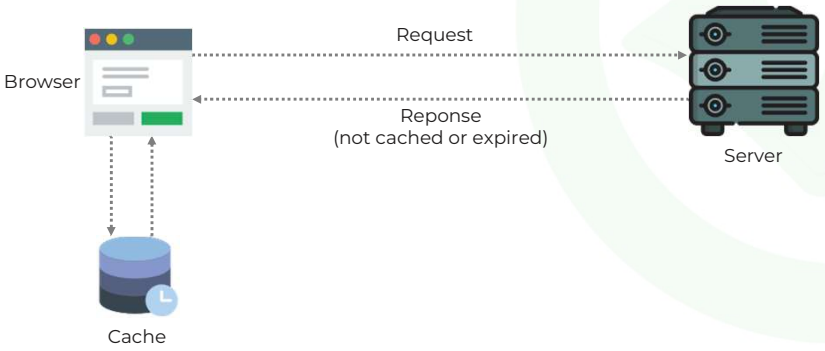
Sass dosyalarını yükler ve css e çevirir.

Caching

- Caching nedir
- Webpack caching
- HtmlWebpackPlugin

Caching

Browser lar, harici css, js ve image dosyalarını ilk yüklemeden sonra cache lerinde saklarlar. Sonraki yüklemelerde cache den yüklerler.

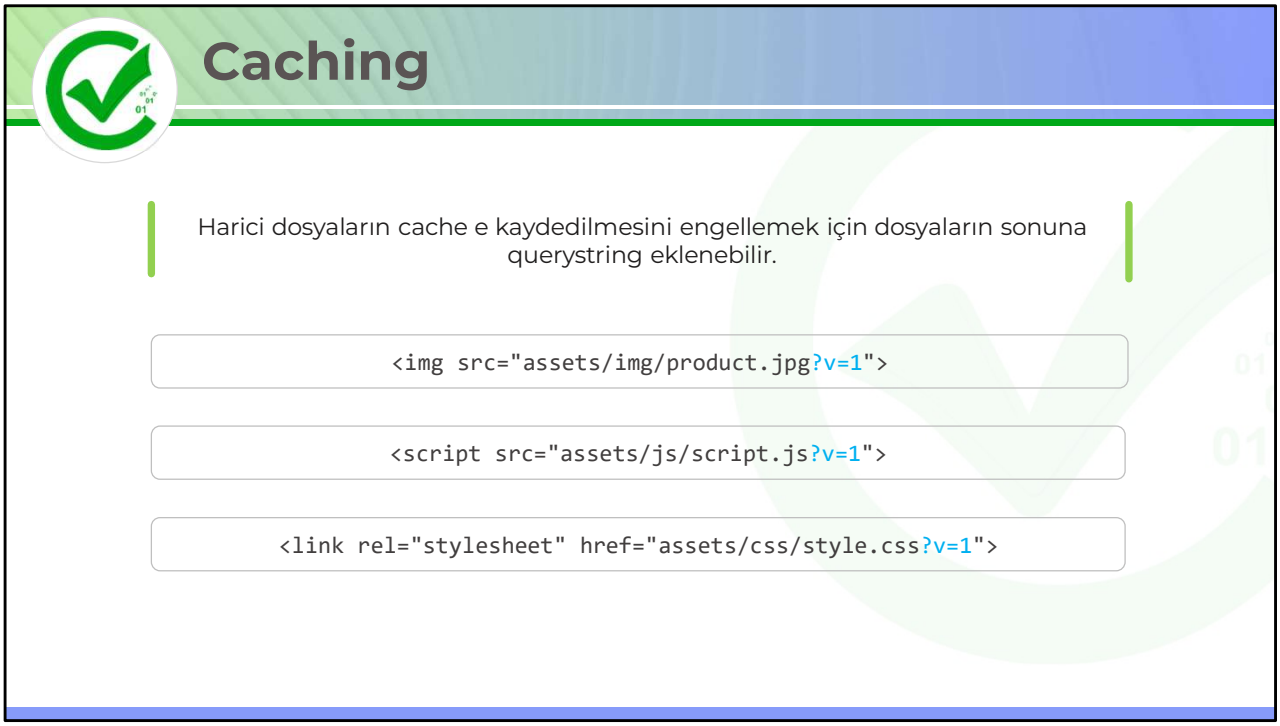


Caching

Tarayıcıların default davranışlarını değiştirmek için, html içinden çeşitli şekillerde ayarlar yapılabilir. Bu işlem sadece html dokümanına etki eder. İçindeki fotoğraf, js ve css harici dosyalarına etki etmez.

```
<meta http-equiv="Cache-Control" content="no-store" />
```

```
<meta http-equiv="expires" content="Fri, 18 Jul 2014 1:00:00 GMT" />
```


Caching
Harici dosyaların cache e kaydedilmesini engellemek için dosyaların sonuna querystring eklenebilir.
<img src="assets/img/product.jpg?v=1">
<script src="assets/js/script.js?v=1">
<link rel="stylesheet" href="assets/css/style.css?v=1">

Webpack Caching

Yapılan değişikliklerin production ortamına aktarılması sırasında değişiklik olan dosyaların isimlerinin değiştirilmesi, tarayıcıların eski versiyonları göstermesini engelleyecektir.

```
output: {
    fileName: "main.[contenthash].js"
}
```

contenthash, dosya adına bir hash string ekleyerek benzersiz olmasını sağlar.

Ancak html dosyası içindeki js referansının güncellenmesi sorunu ortaya çıkar. Bunun için **HtmlWebpackPlugin** kullanılır.

HtmlWebpackPlugin

Development tarafında html template ler oluşturularak, bu template ler temel alınarak, bundle içindeki html dosyalarının oluşturulması sağlanır.

```
npm install --save-dev html-webpack-plugin
```

```
const HtmlWebpackPlugin = require('html-webpack-plugin');

plugins: [new HtmlWebpackPlugin()]
```

HtmlWebpackPlugin

```
plugins:[new HtmlWebpackPlugin({
    template: "./src/index.html",
    inject: 'body'
})],
```

Template dosyası baz alınarak bundle içindeki html oluşturulur.

Script tagını body nin bitimine ekler.

```
output: {
    clean: true
}
```

Dist klasörünü temizlemek için kullanılır

Development & Production

- Development & production modes
- Farklı config dosyaları oluşturmak
- Config dosyalarının birleştirilmesi
- webpack-dev-server

Development & Production

Geliştirme yapılan ortama **development**, projenin yayına hazır olduğu ortama **production** denir. Bu iki ortama özel ayar yapabilmek için webpack.config dosyaları ayrılabilir.

```
const path = require("path");
module.exports = {
   mode: "development",
   output: {
     filename: "main.js",
     path: path.resolve(__dirname, "dist"),
   },
};
```

webpack.dev.js

```
const path = require("path");
module.exports = {
   mode: "production",
   output: {
     filename: "main.[contenthash].js",
     path: path.resolve(__dirname, "dist"),
   },
};
```

webpack.prod.js

[Ortak olan özellikler ise webpack.comon.js dosyasında tutulabilir]

Development & Production

Oluşturulan farklı config dosyalarını birleştirebilmek için **webpack-merge** paketinden faydalanılır.

```
npm i --save-dev webpack-merge
```

```
const path = require("path");
const common = require("./webpack.common");
const { merge } = require("webpack-merge");
module.exports = {
   …
};
```

dev ve prod

```
scripts: {
  "start": "webpack --config webpack.dev.js",
  "build": "webpack --config webpack.prod.js"
}
```

package.json

Webpack-dev-server

Sürekli npm start yapmamak için development ortamı için live-server benzeri bir server kurulabilir.

```
npm i --save-dev webpack-dev-server
```

```
scripts: {
  "start": "webpack-dev-server --config
webpack.dev.js --open",
}
```

package.json

```
devServer: {
  static: {
    directory: path.join(__dirname,
'public'),
  },
  port: 9000,
  watchFiles: ["src/*.html", , "src/*.scss"]
},
```

webpack.dev.js

Webpack

Image processing

- Image processing

Image processing

Development ortamında kullanılan görsellerin production a aktarılması için html-loader kullanılır.

```
npm i --save-dev html-loader
```

```
output:{
    assetModuleFilename: "img/[hash][ext]",
}
```

dev ve prod

```
rules:{
    {
        test: /\.html$/,
        use: ["html-loader"],
    },
    {
        test: /\.(png|svg|jpg|jpeg|gif)$/i,
        type: 'asset/resource',
    }
}
```

webpack-common

- Multiple entry points

Multiple entry points

Multiple entry points

Uygulama içinde birbirinden bağımsız script yapıları varsa, bunlar için ayrı giriş noktaları oluşturulabilir. Webpack bu yapılar için ayrı dosyalar oluşturur. En çok harici kütüphaneleri (vendors) ayırmak için kullanılır.

```
entry: {
        main: "./src/index.js",
        vendor: "./src/vendor.js"
},
```

webpack.common

```
output: {
    filename: "./js/[name].js",
}
```

webpack-dev

```
output: {
    filename: "./js/[name].[contenthash].js",}
```

webpack-prod

Managing CSS

- Extracting CSS
- Minimizing CSS

Extracting CSS

CSS dosyaları default olarak bundle.js dosyasının içine enjekte ediliyor. Bunu engelleyip, css dosyalarını ayırmak böylece bundle.js dosya boyutunu azaltmak için extract pllugin kullanılır.

```
npm install --save-dev mini-css-extract-plugin
```

```
module: {
   rules: [
      {
         test: /\.s(a|c)ss$/,
         use: [
             'style-loader',
             'css-loader',
             'sass-loader'
         ]
      }
   ]
}
```

webpack.dev

Extracting CSS

```
const MiniCssExtractPlugin = require("mini-css-extract-plugin");

plugins: [
  new MiniCssExtractPlugin({
    filename: "./css/[name].[contenthash].css",
  }),
],


rules: [{
   test: /\.s(a|c)ss$/,
   use: [
      MiniCssExtractPlugin.loader,
      "css-loader",
      "sass-loader"
   ],
}],
```

webpack.prod

Minifing CSS

JS kodları default olarak minify yapılır. Ancak CSS dosyalarının minify edilmesi için ayrıca plugin kullanmak gereklidir.

```
npm install css-minimizer-webpack-plugin --save-dev
```

Minifing CSS

```
optimization: {
    minimizer: [
        `...`,
        new CssMinimizerPlugin(),
    ],
},
plugins: [
    new MiniCssExtractPlugin({
        filename: "./css/[name].[contenthash].css",
    }),
],
```

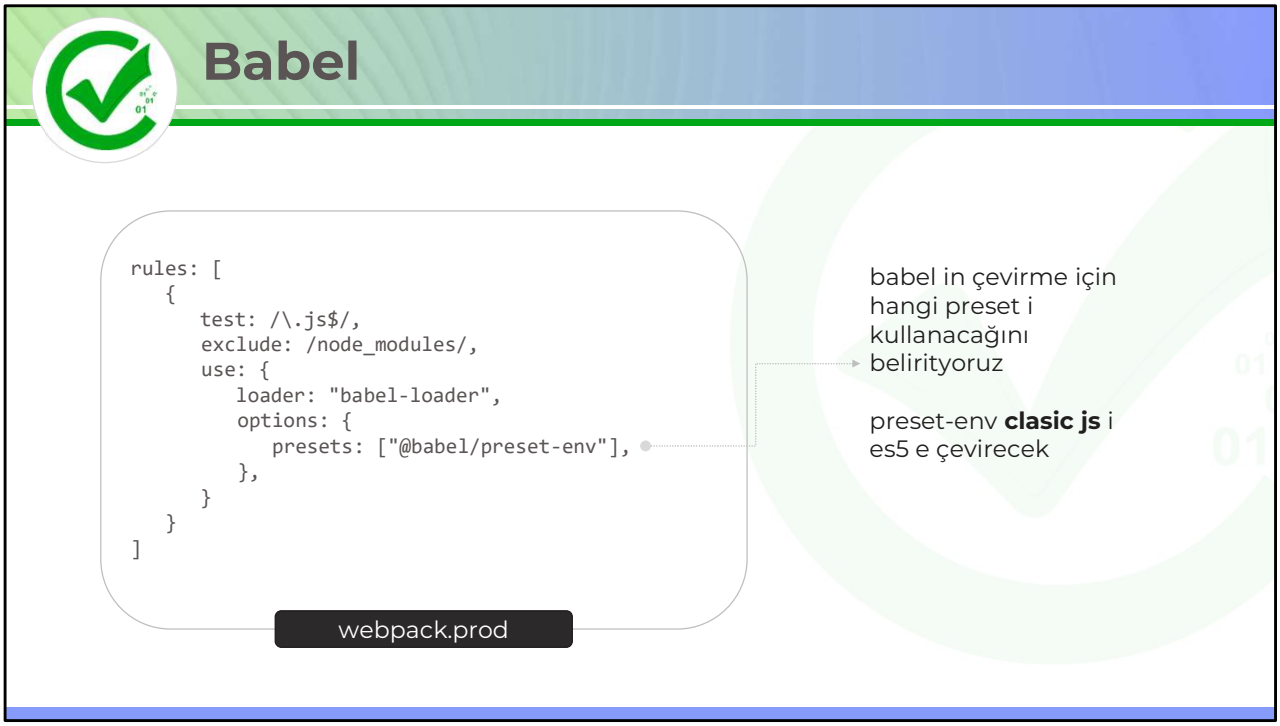
webpack.prod

Spread operatörü ile default minimizer lar buraya açılır. Bu yapılmazsa sonraki satır default javascript minimizer ı ezer.

Babel

Modern java script kodlarının tüm tarayıcılar tarafından desteklenmesi için ES5 standardına çevrilmesi gerekir. Bunun için babel plugini kullanılır.

```
npm install babel-loader @babel/core @babel/preset-env --save-dev
```


Babel
rules: [
{
test: /\.js$/,
exclude: /node_modules/,
use: {
loader: "babel-loader",
options: {
presets: ["@babel/preset-env"],
},
}
}
]
webpack.prod
babel in çevirme için hangi preset i kullanacağını belirityoruz
preset-env **clasic js** i es5 e çevirecek