Imágenes

https://github.com/webpack-contrib/url-loader

```
test: /\.(jpg|png|gif)$/,
use: {
  loader: 'url-loader',
  options: {
    limit: 100000, // maximo peso de la imagen en bytes
  }
}
```

Necesitamos cargar este loader. npm install --save-dev url-loader

Fuentes.

www.fontsquirrel.com

una vez seleccionada la fuente deseada vamos a la pestaña de webfont kit, seleccionamos todos los formatos, y lo descargamos.

En el webpack, le damos soporte a las fuentes tipo woff, eot, ttf, y svg

```
test: /\.(jpg|png|gif|woff|eot|ttf|svg)$/,
use: {
  loader: 'url-loader',
  options: {
    limit: 100000, // maximo peso de la imagen en bytes
  }
```

Debemos ahora copiar el archivo css de las fuentes en nuestros estilos.css

Las fuentes las copiamos en una carpeta llamada fonts.

Podríamos tener un error si las fuentes que tenemos pesan mas de lo que estipulamos en el limite.

Archivos JSON

https://github.com/webpack-contrib/json-loader

```
{
  test: /\.json$/,
  use: 'json-loader'
},
```

Vídeos.

https://github.com/webpack-contrib/file-loader

```
{
  test: /\.(webm|mp4)$/,
  use: {
    loader: 'file-loader',
    options: {
      limit: 300000, // maximo peso del video en bytes
      name: 'videos/[name].[hash].[ext]'
    }
  }
},
```

Los vídeos que no entran en el margen de tamaño máximo permitido sera ubicados en la carpeta llamada vídeos/...

También debemos configurar el output, añadiendo cual es el directorio publico desde el que se buscaran todos los demás archivos.

```
output: {
  path: path.resolve(__dirname, 'dist'),
  filename: 'bundle.js',
  publicPath: './dist/'
```

SASS

https://github.com/webpack-contrib/sass-loader

```
{
  test: /\.scss$/,    //sass
  use: extractTextPlugin.extract({
    use: ["css-loader", "sass-loader"]
  }),
}
```

npm run build:sass -- --progress –watch

Stylus

https://github.com/shama/stylus-loader

```
{
  test: /\.styl$/,
  use: ExtractTextPlugin.extract({
    // ['style-loader','css-loader']
    // fallback: 'style-loader',
    use: [
      "css-loader",
      {
        loader: 'stylus-loader',
        options: {
          use: [
            require('nib'),
            require('rupture')
          ],
          import: [
            '~nib/lib/nib/index.styl',
            '~rupture/rupture/index.styl'
          ]
        }
      }
    ]
  }),
}
```

Less

https://github.com/webpack-contrib/less-loader

```
{
  test: /\.less$/,
  use: ExtractTextPlugin.extract({
    use: ["css-loader", {
      loader: 'less-loader',
      options: {
        noIeCompat: true,
      }
    }]
  }),
},
```

PostCss

https://github.com/postcss/postcss-loader

https://www.npmjs.com/package/postcss-cssnext

https://github.com/MoOx/postcss-cssnext

```
{
  test: /\.css$/,
  use: ExtractTextPlugin.extract({
    use: [
      {
        loader: 'css-loader',
        options: {
          modules: true,
          importLoaders: 1
        }
      },
      'postcss-loader'
    ]
  }),
},
```

Prevenir duplicación

Cuando tenemos multiples entryPoints,

```
entry: {
  home: path.resolve(__dirname, 'src/js/index.js'),
  contact: path.resolve(__dirname, 'src/js/contact.js')
},
```

si varios están asociados a componentes de react o angular, por ejemplo, multiplicaremos el código de react/angular por cada entryPoint ya que en ambos estamos importando los mismos paquetes en algunos casos, como por ejemplo react, o react-dom. Para evitar esto, en nuestro webpack importamos webpack:

```
const webpack = require('webpack')
```

luego debemos añadir un plugin:

```
plugins: [
  // aquí van los plugins
  new ExtractTextPlugin("css/[name].css"),
  new webpack.optimize.CommonsChunkPlugin({
    name: 'common'
  })
]
```

este, plugin lo que hace es capturar todo el código común y empaquetarlo en un nuevo archivo, que en este caso hemos llamado ‘common’, así solo lo tendremos una vez.

Cuando tenemos múltiples entryPoints debemos ponerle nombres dinámicos a las salidas.

```
output: {
  path: path.resolve(__dirname, 'dist'),
  filename: '[name].js'
```

El nombre dinámico esta en la ultima linea.

Ahora en el html debemos cargar el modulo principal de cada entryPoint, y los archivos comunes.

```
<section id="container"></section>
<script src="dist/common.js"></script>
<script src="dist/home.js"></script>
<script src="dist/contac.js"></script>
```

Es importante que el primer archivo sea el de los archivos comunes.

https://gist.github.com/sokra/1522d586b8e5c0f5072d7565c2bee693

https://webpack.js.org/plugins/dll-plugin/