モダンな
フロントエンド開発環境
を作ろう

前回のおさらい

前回、最近のフロントエンドフレームワーク

「Vue.js」「React」「Angular」

を使って開発すると

「見た目をオブジェクトで操作できる」し
「ビューとロジックを分離できる」ので

開発しすいよ！　って話をしましたね？

本日のお題

実際に

どんな開発環境を用意したらいいの？

今までと変わっちゃうの？

について今日は説明します！

レガシー／モダンな環境の違い

レガシーな開発では...

基本的に次の２つのツールを使っていたはず。

- **エディタ - HTML、CSS、JavaScript**を記述する
- **ブラウザ -** 動作確認やデバッグする

レガシーな開発では…

プロダクトは人によって使っていたものが変わってくると思いますが…

- エディタ - **HTML、CSS、JavaScript**を記述する
  - **IDE**付属のエディタ**(Eclipse、Visual Studio**の標準エディタ**)**
  - テキストエディタ(サクラエディタ、メモ帳)
- ブラウザ - 動作確認やデバッグする
  - **Chrome**
  - **IE**
  - **FireFox**
  - **Safari**

モダンな開発では…どう変わる？

基本的に前者の「エディタ」と「ブラウザ」は一緒。

違いとして、新しく3つのツールが登場！

モダンな開発では…どう変わる？

基本的に前者の「エディタ」と「ブラウザ」は一緒。

違いとして、新しく**3**つのツールが登場！

- **サーバーサイドJavaScript環境**

モダンな開発では…どう変わる？

基本的に前者の「エディタ」と「ブラウザ」は一緒。

違いとして、新しく**3**つのツールが登場！

- **サーバーサイドJavaScript環境**
- **パッケージ管理ツール**

モダンな開発では…どう変わる？

基本的に前者の「エディタ」と「ブラウザ」は一緒。

違いとして、新しく**3**つのツールが登場！

- **サーバーサイドJavaScript環境**
- **パッケージ管理ツール**
- **バンドルに必要なツール群**

モダンな開発では…どう変わる？

基本的に前者の「エディタ」と「ブラウザ」は一緒。

違いとして、新しく**3**つのツールが登場！

- **サーバーサイドJavaScript環境 -** バンドルツールの実行基盤
- **パッケージ管理ツール - JavaScript**パッケージのバージョン管理ツール
- **バンドルに必要なツール群 -** モジュールのバンドルツール

モダンな開発では…どう変わる？

基本的に前者の「エディタ」と「ブラウザ」は一緒。

違いとして、新しく3つのツールが登場！

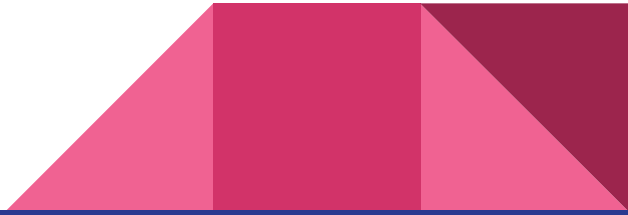
すでに
疑問をお持ちの方もいるはず。。。
「バンドル」って一体何者！？

- サーバーサイドJavaScript環境 - バンドルツールの実行基盤
- パッケージ管理ツール - JavaScriptパッケージのバージョン管理ツール
- バンドルに必要なツール群 - モジュールのバンドルツール

最近のフロントエンド界隈では...

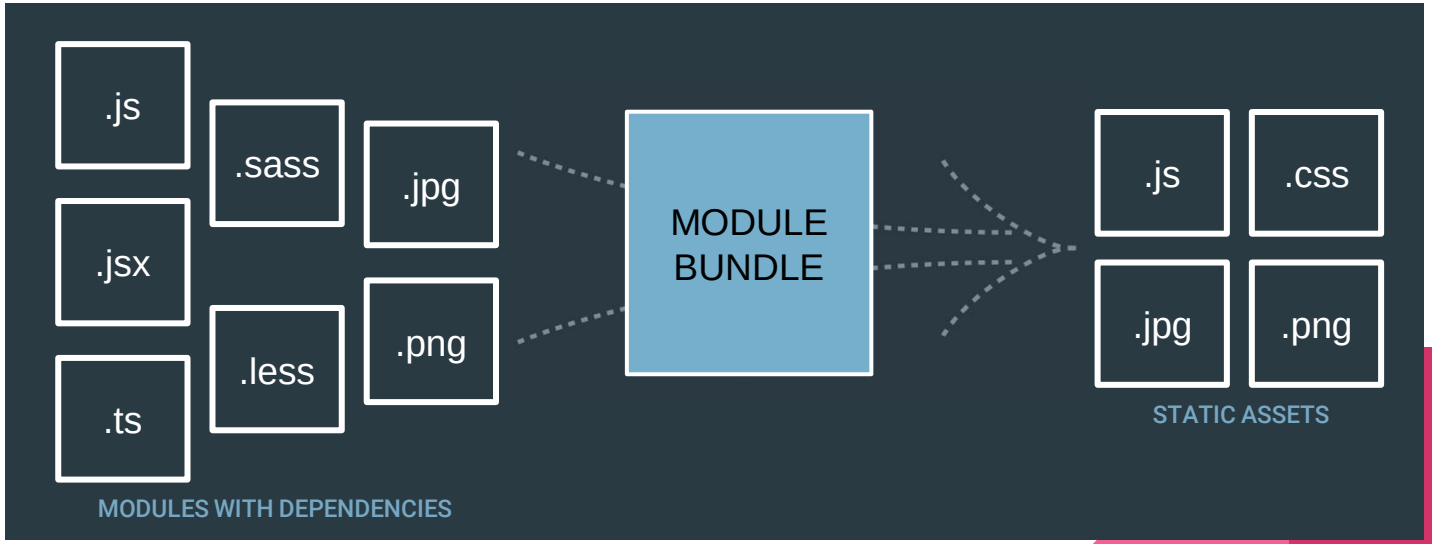
バンドル(ビルド)

が必須になりました。



バンドルとは？

(モジュール)バンドルとは、依存関係のあるモジュール群を
ビルドしたり、**依存関係を解決して１つにまとめたり**ミニファイしたりして
ブラウザ上で動作させるのに最適な**静的資産**に変換すること

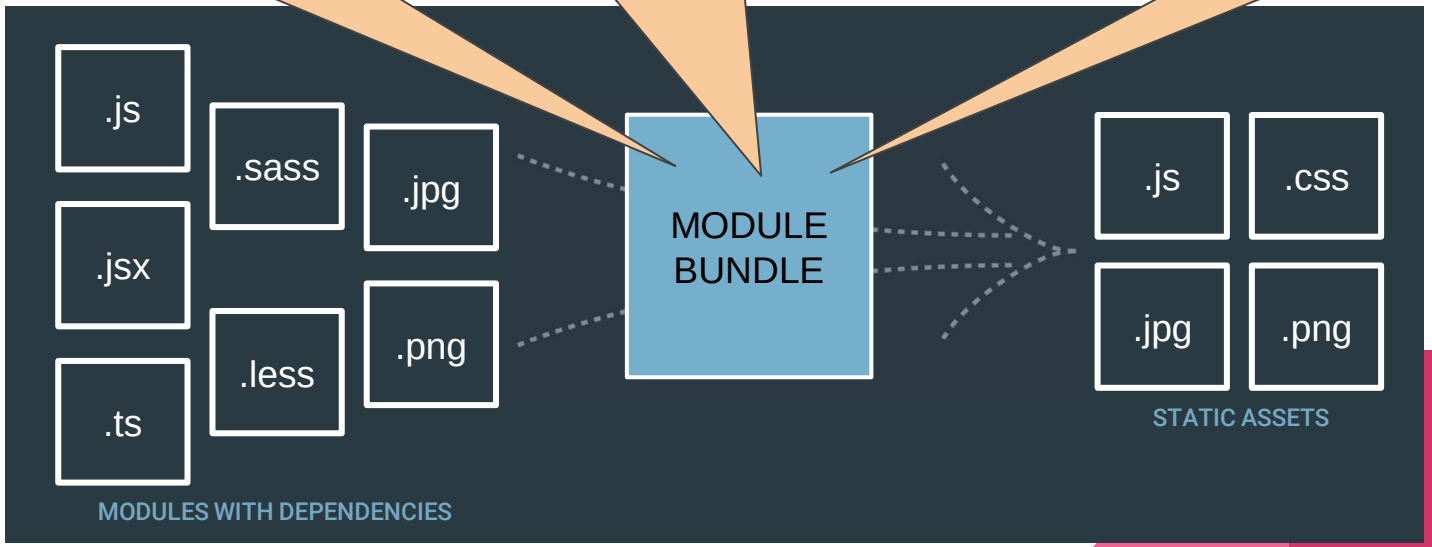


バンドルとは？

.jsx(JSXファイル)
.ts(TypeScriptファイル)
を.jsにビルドしたり
.sassや.lessを.cssにビルド
してくれる

ビルドしてできた.jsや複数に
別れた.jsを依存関係を解決し
た状態で１つにまとめて
くれる

.jsや.cssを
ミニファイ(圧縮)
してくる



いろんな形式で書かれたファイルを
ブラウザで動くようにしてくれるし

なんかバンドルって便利そうなツールかも？？

ただ、ここで
疑問をお持ちの方いませんか！？

本日は

「Vue.js」 「React」 「Angular」

の開発環境についての話だったはず。。。

疑問：
「Vue.js」「React」「Angular」
を使うのにバンドルって必要？

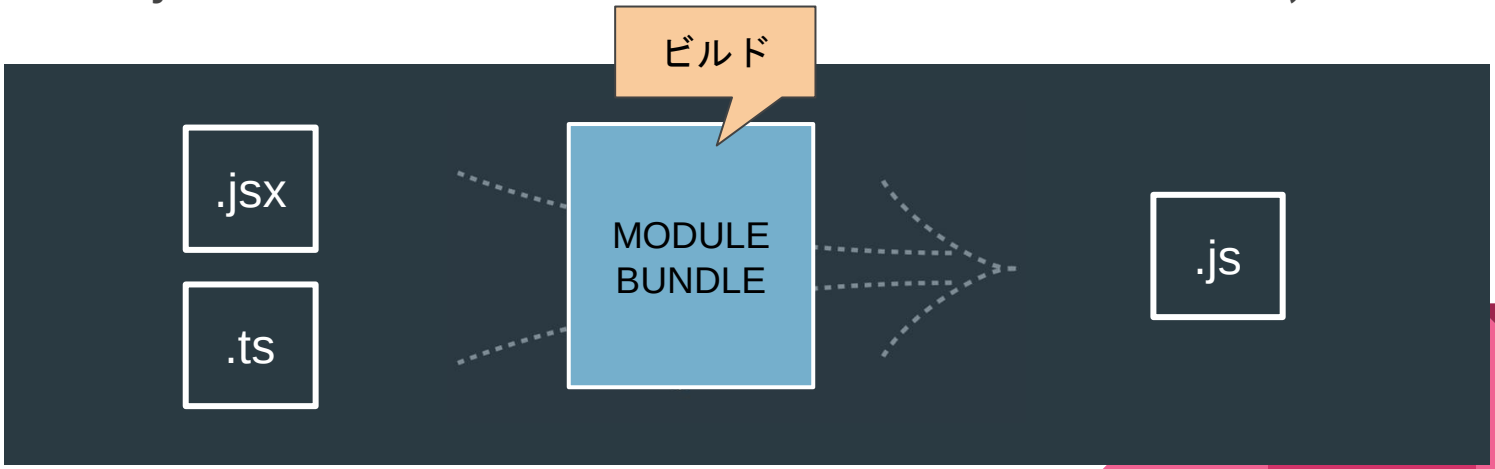
疑問：
「Vue.js」「React」「Angular」を使うのにバンドルって必要？

答え：

「**React**」「**Angular**」を使うには基本的に必須！
「**Vue.js**」はなくても大丈夫だが、開発効率を上げるには使ったほうが良い！

バンドルが必要！という解説

Reactで開発する上で標準となる「**JSX(.jsx)**」や
Angularで開発する上で必要となる「**TypeScript(.ts)**」
は、そのままではブラウザは認識しない！なのでビルドが必須！
(ちなみに**Vue.js**も**.vue**ファイルで開発できるので、その際は必須！)



大事なのでもう一度言います

最近のフロントエンド開発では

バンドル(ビルド)

が必須になりました。

大事なのでもう一度言います

なので
バンドル使って
開発しましょう！

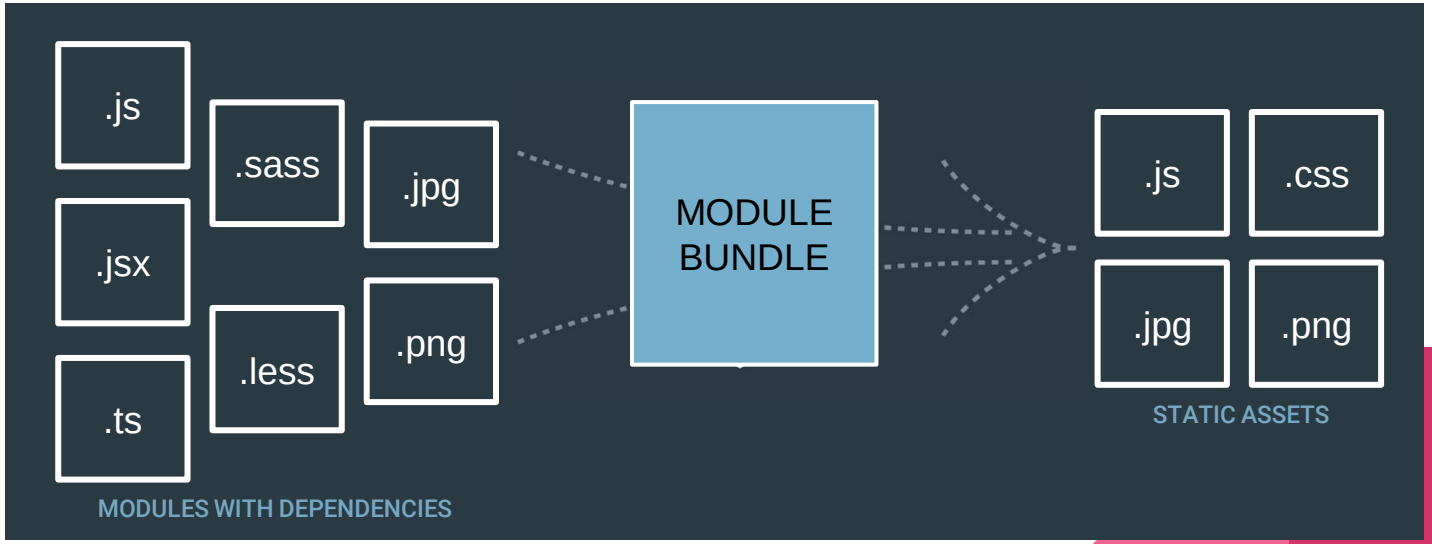
最近のフロントエンド開発では

バンドル(ビルド)

が必須になりました。

ちなみに。。。

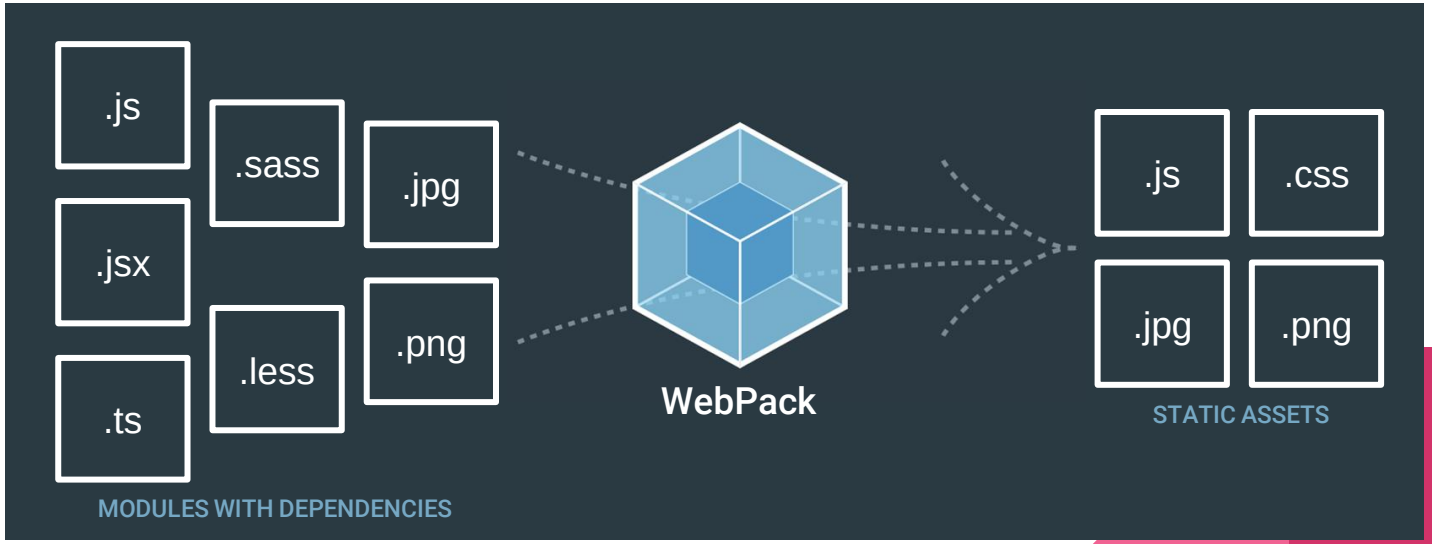
この後詳しく話しますが。。。
今回はバンドルツールに



ちなみに。。。

この後詳しく話しますが。。。
今回はバンドルツールに
「WebPack」というツールを使う！

最近のデファクトなので
覚えておくように！！



モダンな開発環境を作ろう

モダンな開発環境

- エディタ - **HTML、CSS、JavaScript**を記述する
  - **Visual Studio Code**
- ブラウザ - 動作確認やデバッグする
  - **Chrome**
  - **(FireFox、Safari、IE、Edge、etc...)**
- サーバーサイド**JavaScript環境** - バンドル(ビルド)ツールの実行基盤
  - **Node.js(nodist、nodenv)**
- パッケージ管理ツール - **JavaScript**パッケージのバージョン管理する
  - **NPM(Node.js)**
  - **(yarn)**
- バンドル(ビルド)に必要なツール(パッケージ)群 - モジュールをバンドルする
  - **WebPack、Babel、TypeScriptCompiler、etc…**

エディタ

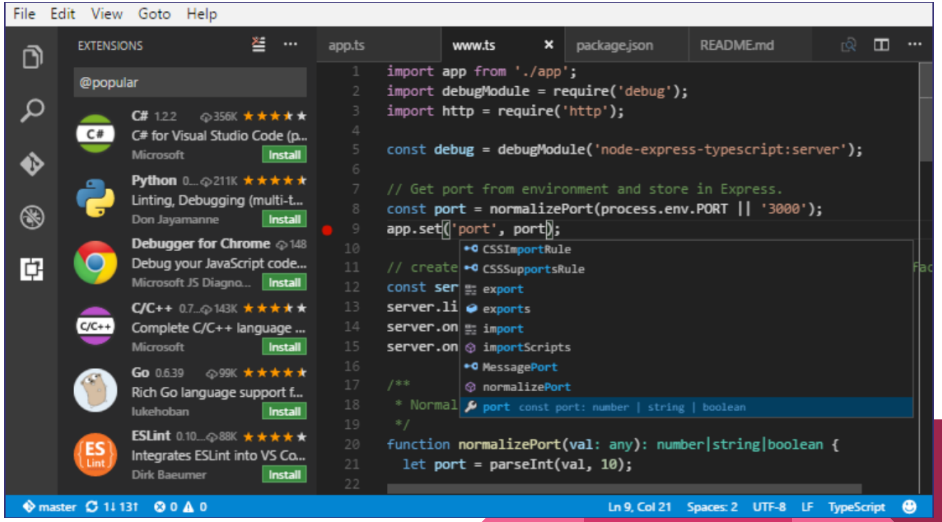
HTML、CSS、JavaScriptを記述するツール

オススメのプロダクト

Visual Studio Code

オススメのポイント

JSXや**TypeScript**などモダンなフロントエンド開発に欠かせないプログラミング言語に標準で対応している(プラグイン導入済)
無償で使える！



ブラウザ

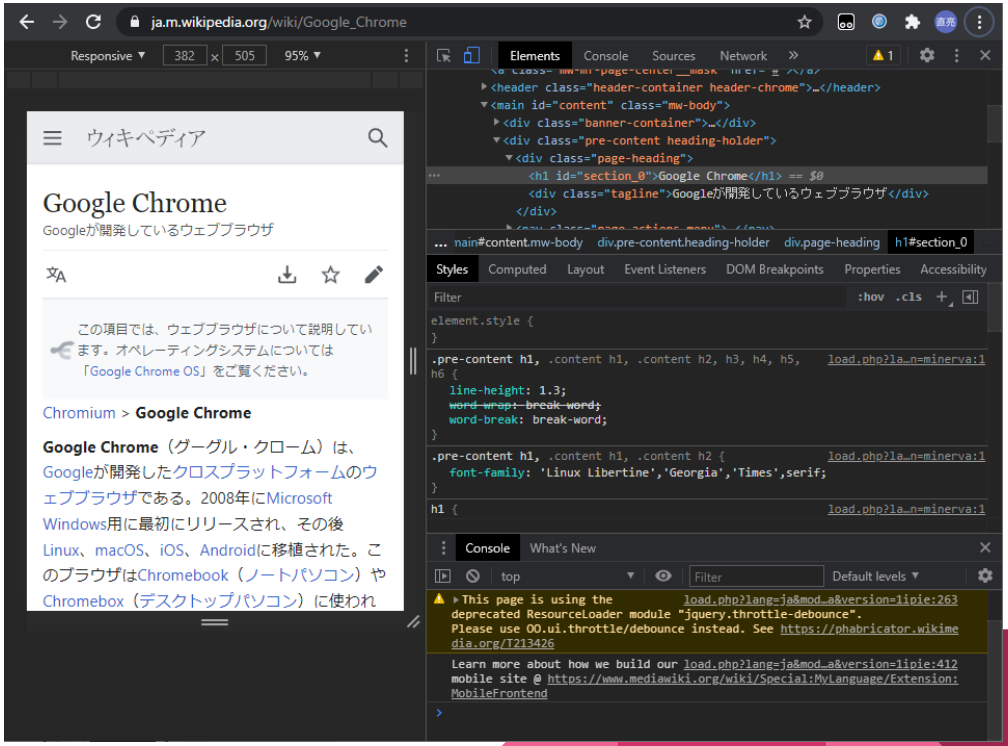
動作確認やデバッグするツール

オススメのプロダクト

Chrome

オススメのポイント

言わずと知れた**Google**製の**Web**ブラウザ。
特にデバッグする際はオススメ！標準で強力なデバッグ機能が搭載されている。
もちろんのこと無償で使える！



サーバーサイドJavaScript環境

バンドル(ビルド)ツールの実行基盤

オススメのプロダクト

Node.js

オススメのポイント

オススメというよりも、**Node.js**一択。
利用するバージョンは
LTS(Long-term support)版を利用しよう。
これまた無償です！

サーバーサイドJavaScript環境

オススメでもなくなってきたのでスキップ

補足）**Node.js**をインストールするには？

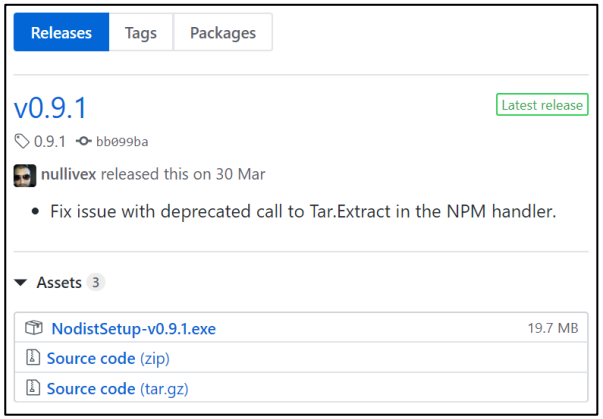
公式サイトのインストーラを使ってもいいですが。。。
Node.jsのバージョン管理ツール(無償)があるのでそちらの利用をオススメ！
バージョンの切り替えやアップデートなどコマンドで実行できる！

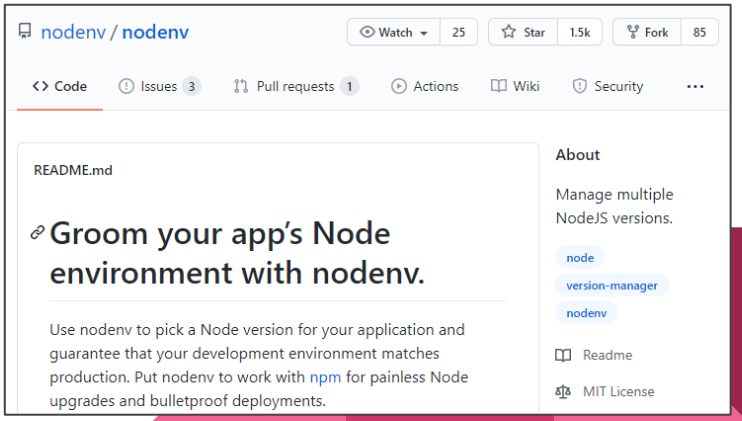
Windows

[nodist]

Mac

[nodenv]





パッケージ管理ツール

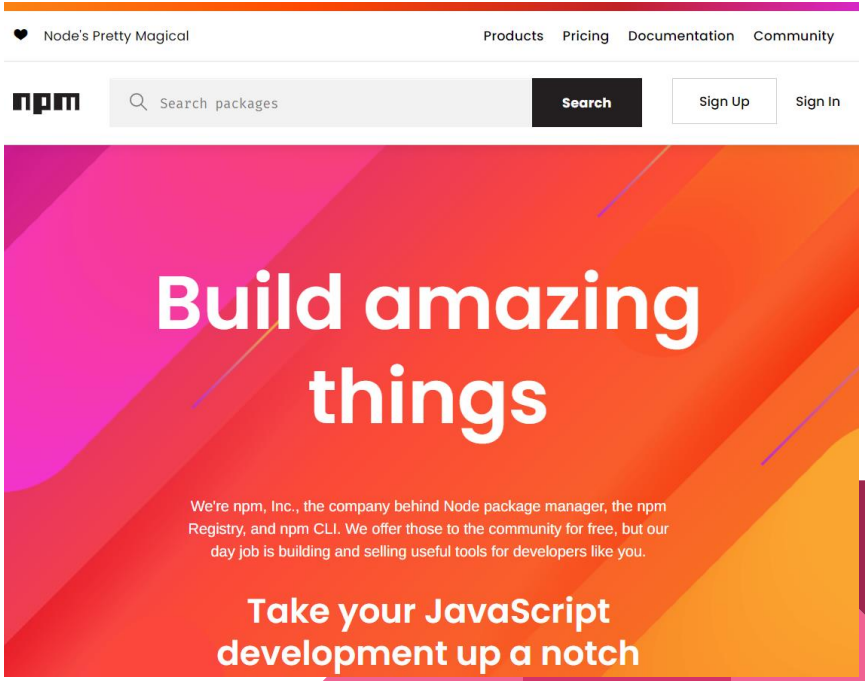
JavaScriptパッケージのバージョン管理する

オススメのプロダクト

NPM (Node Package Manager)

オススメのポイント

こちらもオススメというより**NPM**一択。
NPMは**JavaScript**のパッケージを管理している「リポジトリ」とそのリポジトリからパッケージを取得する「**npm**コマンド」というクライアントツールで構成される。
「**npm**コマンド」は**Node.js**をインストールすると一緒にインストールされる(つまり無償)



パッケージ管理ツール

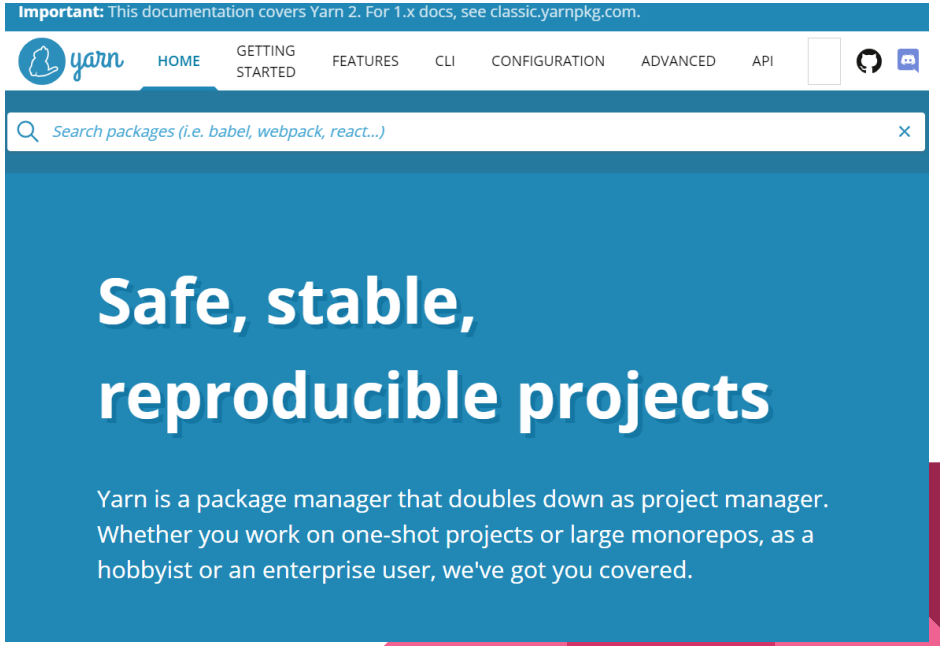
補足）**npm**コマンドの代替プロダクト

オススメのプロダクト

yarn

オススメのポイント

このプロダクトは必須ではないが、メジャーなプロダクトなので紹介。
Facebook、**Google**、**Exponent**、**Tilde**が作った**JavaScript**パッケージマネージャー(無償)
パッケージの取得先リポジトリは「**npm**コマンド」と同じ。パッケージのダウンロードなど高速になるので興味ある方はどうぞ！



バンドル(ビルド)に必要なツール(パッケージ)群

JavaScriptパッケージ管理(**npm**コマンド)を使って
バンドルに必要なパッケージを取得！
(もちろん無償で使えます)

webpack

TS
TypeScript

BABEL

バンドル(ビルド)に必要なツール(パッケージ)群

JavaScriptパッケージ管理(**npm**コマンド)を使って
バンドルに必要なパッケージを取得！
(もちろん無償で使えます)

これらの使い方は次の章で！

webpack

TS
TypeScript

BABEL

バンドル(ビルド)環境を作ろう

WebPackを使ってバンドル環境を作る

環境を作るには、**WebPack**の設定**(webpack.config.js)**を作成する。

```
var path = require('path');

module.exports = {
  mode: 'development’,
  entry: './foo.js’,
  output: {
    path: path.resolve(__dirname, 'dist’),
    filename: 'foo.bundle.js’
  }
};
```

今回は

Vue.js向けの
バンドル環境を作ってみよう！

Vue.jsのバンドル環境を作る

Vue.jsでは、
単一ファイルコンポーネント(.vueファイル)
というHTML、JavaScript、 CSSを１つのファイルに合わせて部品を作れる機能を持っている。

今回は、この.vueファイルをビルドできる環境を作る。

App.vue

```
<template>
  <div id="app">
    <img src="./assets/logo.png">
    <router-view/>
  </div>
</template>

<script>
export default {
  name: 'App'
}
</script>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
}
</style>
```

WebPackを使ってバンドル環境を作る

Vue.js向けの設定例(webpack.dev.conf.js)

```
'use strict'
const utils = require('./utils')
const webpack = require('webpack')
const config = require('../config')
const merge = require('webpack-merge')
const path = require('path')
const baseWebpackConfig = require('./webpack.base.conf')
const CopyWebpackPlugin = require('copy-webpack-plugin')
const HtmlWebpackPlugin = require('html-webpack-plugin')
const FriendlyErrorsPlugin = require('friendly-errors-webpack-plugin')
const portfinder = require('portfinder')
```

まだまだ続くよ

WebPackを使ってバンドル環境を作る

Vue.js向けの設定例(webpack.dev.conf.js)

```
const devWebpackConfig = merge(baseWebpackConfig, {
module: {
rules: utils.styleLoaders({ sourceMap: config.dev.cssSourceMap, usePostCSS:
true })
},
// cheap-module-eval-source-map is faster for development
devtool: config.dev.devtool,

// these devServer options should be customized in /config/index.js
devServer: {
```

まだまだまだ続くよ

WebPackを使ってバンドル環境を作る

Vue.js向けの設定例(webpack.dev.conf.js)

```
clientLogLevel: 'warning',
historyApiFallback: {
  rewrites: [
    { from: /.*/, to: path.posix.join(config.dev.assetsPublicPath,
'index.html') },
  ],
},
hot: true,
contentBase: false, // since we use CopyWebpackPlugin.
compress: true,
host: HOST || config.dev.host,
port: PORT || config.dev.port,
open: config.dev.autoOpenBrowser,
overlay: config.dev.errorOverlay
  ? { warnings: false, errors: true }
  : false,
```

まだまだまだまだ続くよ

WebPackを使ってバンドル環境を作る

Vue.js向けの設定例(webpack.dev.conf.js)

```
  publicPath: config.dev.assetsPublicPath,
  proxy: config.dev.proxyTable,
  quiet: true, // necessary for FriendlyErrorsPlugin
  watchOptions: {
    poll: config.dev.poll,
  }
},
plugins: [
  new webpack.DefinePlugin({
    'process.env': require('../config/dev.env’)
  }),
  new webpack.HotModuleReplacementPlugin(),
  new webpack.NamedModulesPlugin(), // HMR shows correct file names in console
on update.
  new webpack.NoEmitOnErrorsPlugin(),
```

まだまだまだまだまだ続くよ

WebPackを使ってバンドル環境を作る

Vue.js向けの設定例(webpack.dev.conf.js)

```
    // https://github.com/ampedandwired/html-webpack-plugin
    new HtmlWebpackPlugin({
      filename: 'index.html',
      template: 'index.html',
      inject: true
    }),
    // copy custom static assets
    new CopyWebpackPlugin([
      {
        from: path.resolve(__dirname, '../static'),
        to: config.dev.assetsSubDirectory,
        ignore: ['.*']
      }
    ])
  ]
})
```

まだまだまだまだまだまだ続くよ

WebPackを使ってバンドル環境を作る

Vue.js向けの設定例(webpack.dev.conf.js)

```
module.exports = new Promise((resolve, reject) => {
  portfinder.basePort = process.env.PORT || config.dev.port
  portfinder.getPort((err, port) => {
    if (err) {
      reject(err)
    } else {
      // publish the new Port, necessary for e2e tests
      process.env.PORT = port
      // add port to devServer config
      devWebpackConfig.devServer.port = port
```

まだまだまだまだまだまだまだまだ続くよ

WebPackを使ってバンドル環境を作る

Vue.js向けの設定例(webpack.dev.conf.js)

```
// Add FriendlyErrorsPlugin
devWebpackConfig.plugins.push(new FriendlyErrorsPlugin({
  compilationSuccessInfo: {
    messages: [`Your application is running here:
http://${devWebpackConfig.devServer.host}:${port}`],
  },
  onErrors: config.dev.notifyOnErrors
  ? utils.createNotifierCallback()
  : undefined
}))

resolve(devWebpackConfig)
```

長っ！

まだまだまだまだまだまだまだまだまだ続くよ

WebPackを使ってバンドル環境を作る

正直、自分でイチから設定するのは厳しい！

しかも、ちゃんとしたバンドル環境を作ろうとすると
WebPackの設定だけでなく、他にも数多くの設定が必要となる！！！

WebPackを使ってバンドル環境を作る

こんなのイチイチ
作ってられないorz

正直、自分でイチから設定するのは厳しい！

しかも、ちゃんとしたバンドル環境を作ろうとすると
WebPackの設定だけでなく、他にも数多くの設定が必要となる！！！

そこで登場するのが...

我らが救世主

「**CLI**」

CLIを使ってバンドル環境を作ろう

イチから環境を作るのは大変なので、
各フロントエンドフレームワークはバンドル環境(アプリの雛形)を
構築するCLIを用意してくれている！

- Vue.js
  - Vue CLI
- React
  - React Create App
- Angular
  - Angular CLI

CLIを使ってバンドル環境を作ろう

イチから環境を作るのは大変なので、
各フロントエンドフレームワークはバンドル環境(アプリの雛形)を
構築するCLIを用意してくれている！

- Vue.js
  - Vue CLI
- React
  - React Create App
- Angular
  - Angular CLI

素直にCLIを
使いましょう^^

CLIをインストール

npmコマンドを使ってCLIをインストールできる！
例：Vue CLIをインストール

```
# npm install -g @vue/cli
```

CLIをインストール

インストールに成功すると「**CLI**」コマンドが実行できるようになる！
例：**Vue CLI**コマンドを実行してバージョンを確認

```
# vue --version
@vue/cli 4.5.11
```

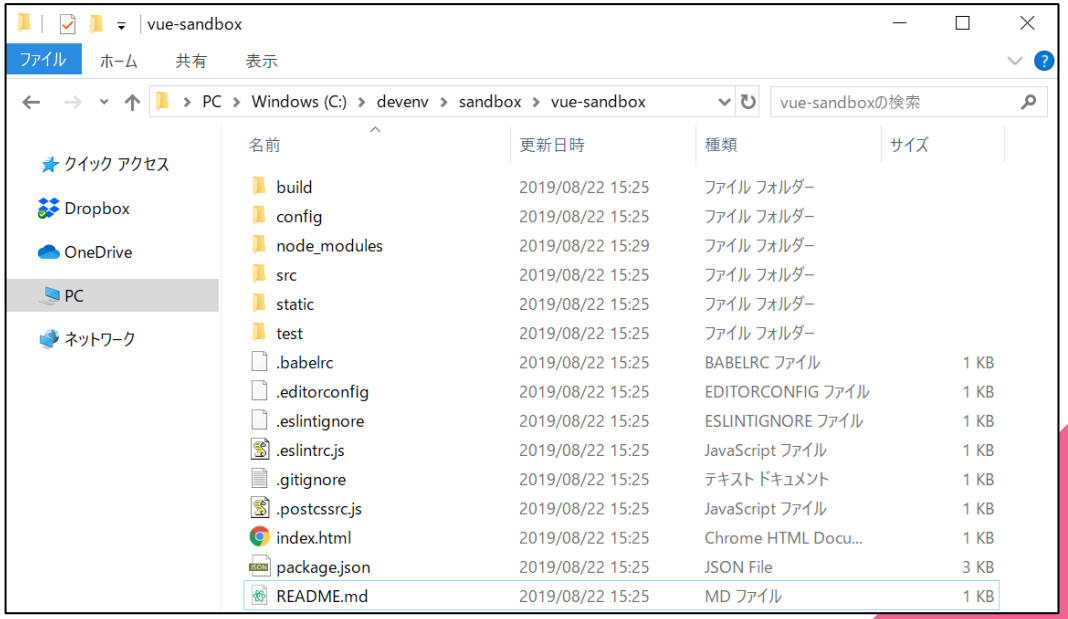
CLIでバンドル環境を作成

「**vue init webpack 【プロジェクト名】**」を実行して**Vue.js**向けのバンドル環境(アプリの雛形)を作成する。

```
# vue init webpack vue-sandbox
```

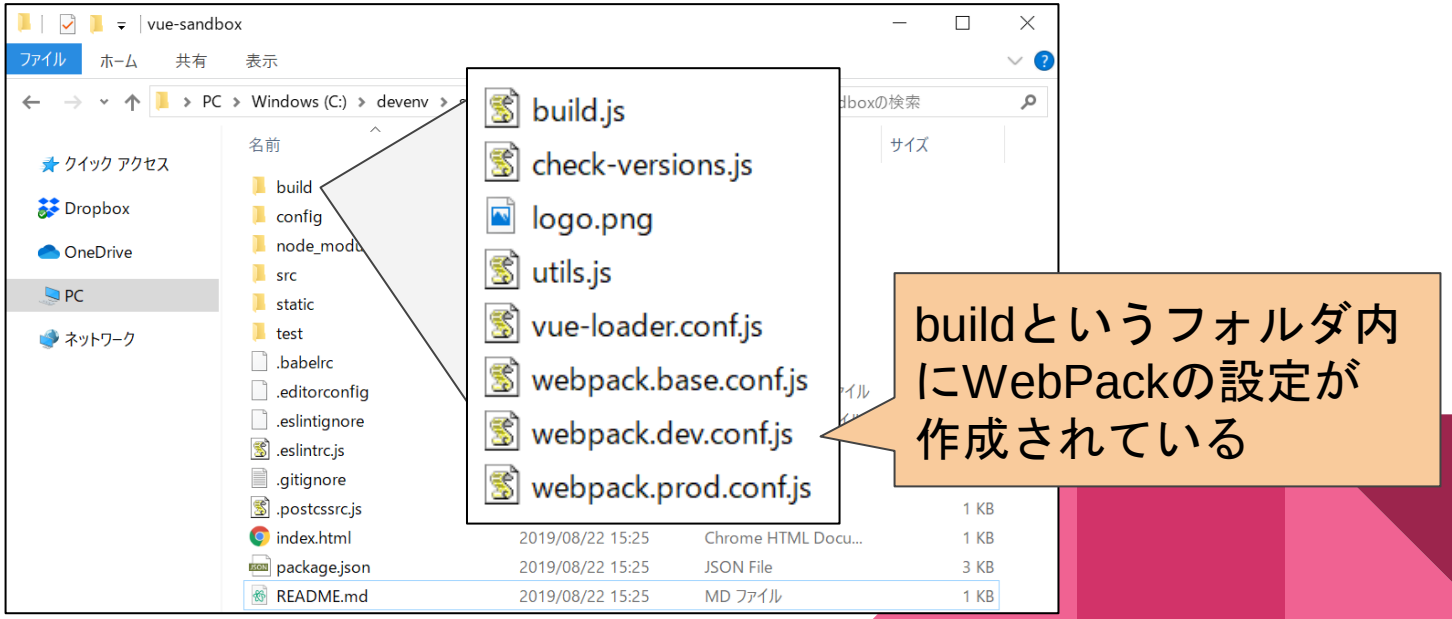
CLIでバンドル環境を作成

成功するとプロジェクトフォルダの中に
バンドルに必要な設定ファイルやサンプルコードなどが作成される！



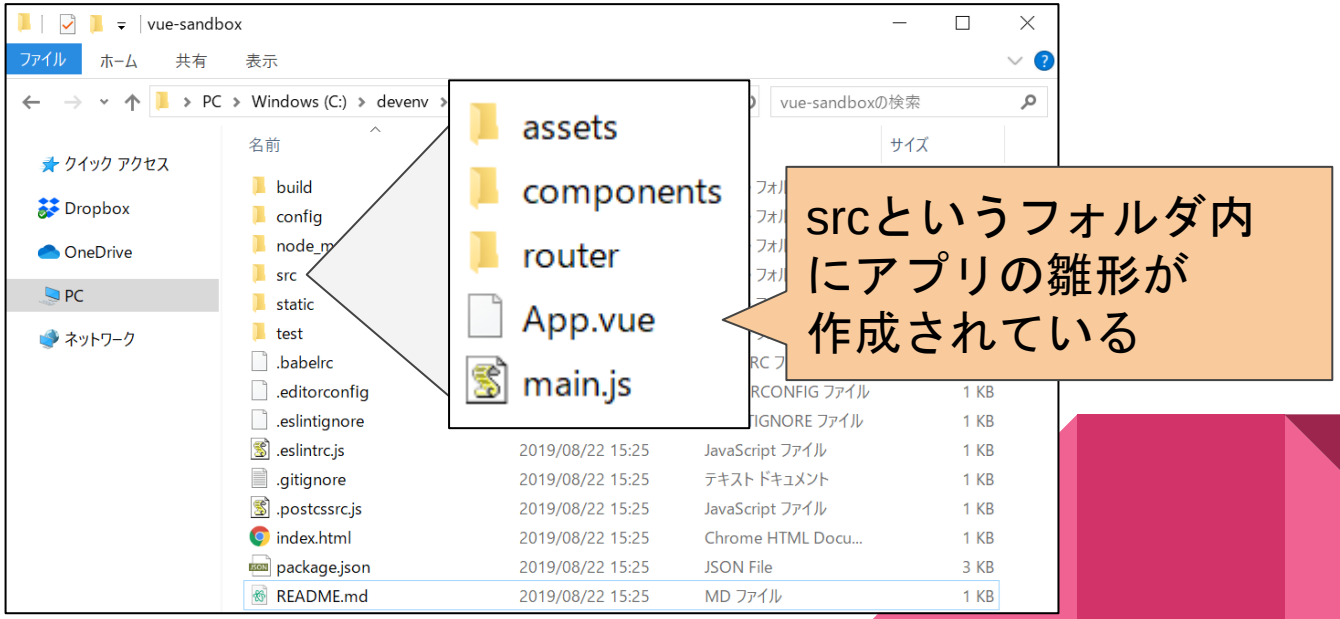
CLIでバンドル環境を作成

成功するとプロジェクトフォルダの中に
バンドルに必要な設定ファイルやサンプルコードなどが作成される！



CLIでバンドル環境を作成

成功するとプロジェクトフォルダの中に
バンドルに必要な設定ファイルやサンプルコードなどが作成される！



バンドル(ビルド)実行

プロジェクトのフォルダに移動して「**npm run dev**」を実行すると
WebPackを使ったバンドル(ビルド)が実行されます。

```
# cd vue-sandbox
# npm run dev

> vue-sandbox@1.0.0 dev C:¥devenv¥sandbox¥vue-sandbox
> webpack-dev-server --inline --progress --config
build/webpack.dev.conf.js

 10% building modules 0/1 modules 1 active ... (以下略)
```

バンドル(ビルド)実行

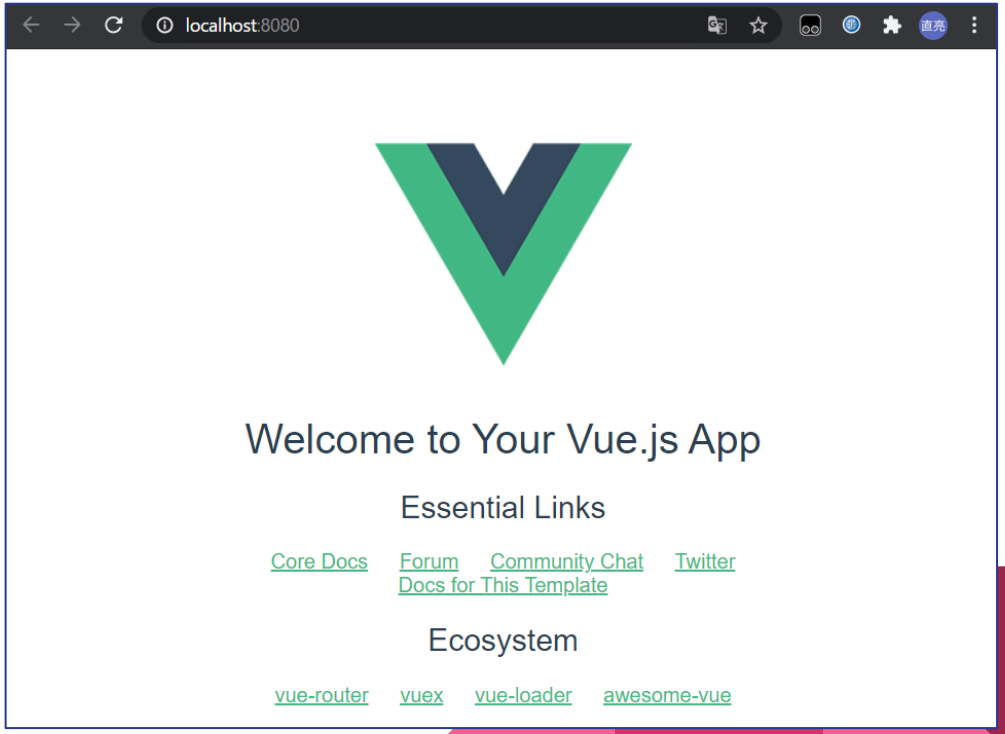
バンドルに成功すると下記のようにアプリが
http://localhost:8080
で実行されていると表示される！

```
(上略)
95% emitting DONE  Compiled successfully in 6522ms16:01:10

I  Your application is running here: http://localhost:8080
```

バンドル(ビルド)実行

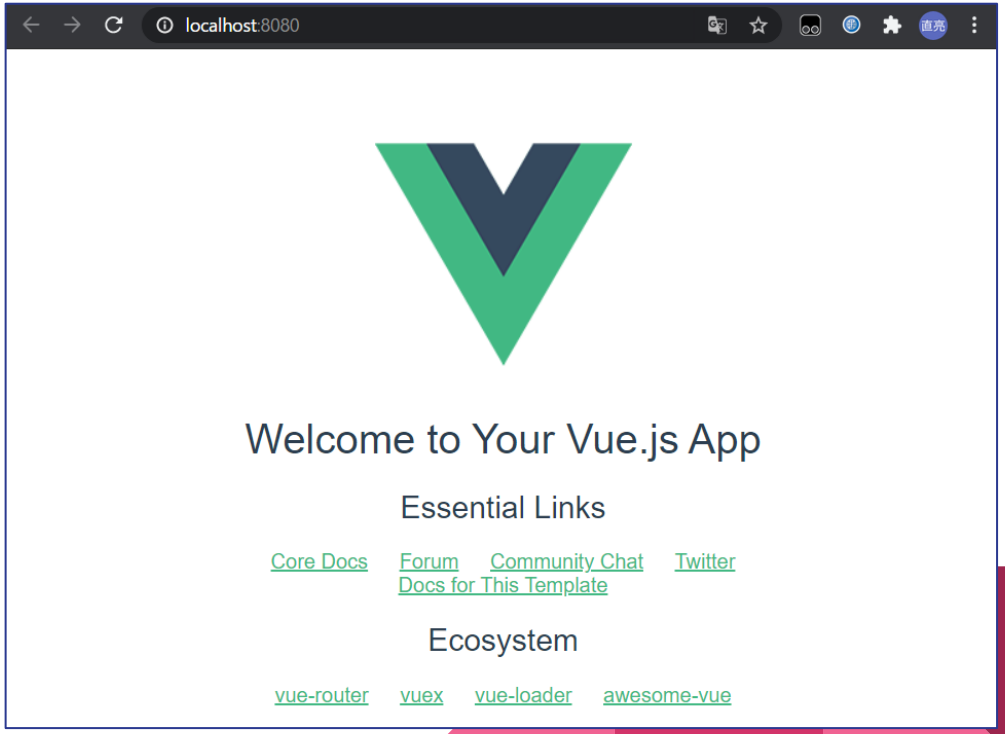
試しにブラウザで
http://localhost:8080
にアクセスすると、
アプリが起動していることを
確認できる。



バンドル(ビルド)実行

試しにブラウザで
http://localhost:8080
にアクセスすると、
アプリが起動していることを
確認できる。

CLIを使うと
簡単にバンドル環境が作れる^^
(React、AngularもCLIを使うと
環境が簡単に作れる！！！)



本日のまとめ

まとめ

モダンなフロントエンド開発では

バンドル(ビルド)が必須になりました！

まとめ

モダンなフロントエンド開発では

バンドル(ビルド)が必須になりました！

バンドル(ビルド)環境を作るには

サーバーサイドJavaScript環境(Node.js)をインストールしよう！

まとめ

モダンなフロントエンド開発では

バンドル(ビルド)が必須になりました！

バンドル(ビルド)環境を作るには

サーバーサイドJavaScript環境(Node.js)をインストールしよう！

バンドル(ビルド)には

WebPack(と関連するパッケージ)を使います！

まとめ

モダンなフロントエンド開発では

バンドル(ビルド)が必須になりました！

バンドル(ビルド)環境を作るには

サーバーサイドJavaScript環境(Node.js)をインストールしよう！

バンドル(ビルド)には

WebPack(と関連するパッケージ)を使います！

WebPackの設定はイチから手組すると大変なので

各フロントエンドフレームワークのCLIを使いましょう！

ご静聴ありがとうございました。

終

制作・著作

SWB