



Vue CLI



Vue CLI の前に

JavaScript 開発環境

アフターES6

Google Chrome ・ ・ ブラウザ

Visual Studio Code ・ ・ エディタ

Babel ・ ・ ES6->ES5への変換

BABEL



Node.js(npm) ・ ・ JSライブラリ管理

webpack ・ ・ 1つにまとめる(読み込み速度up)

polyfill ・ ・ 下位互換

各loader ・ ・ css,fileなど



フロントエンド開発環境

新コード->旧コード対応

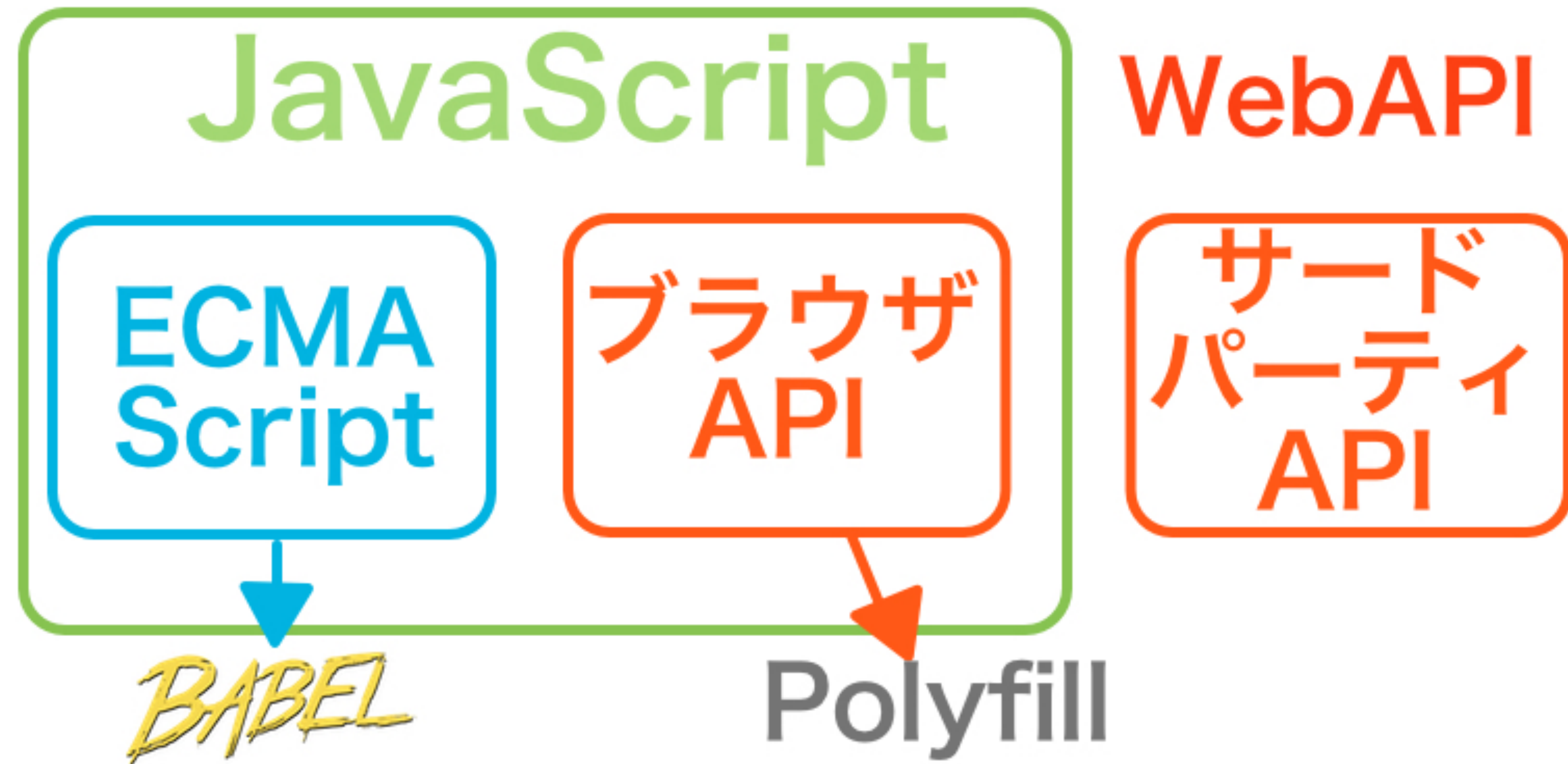
IE11への対応

(2020年2月時点でシェア11%)

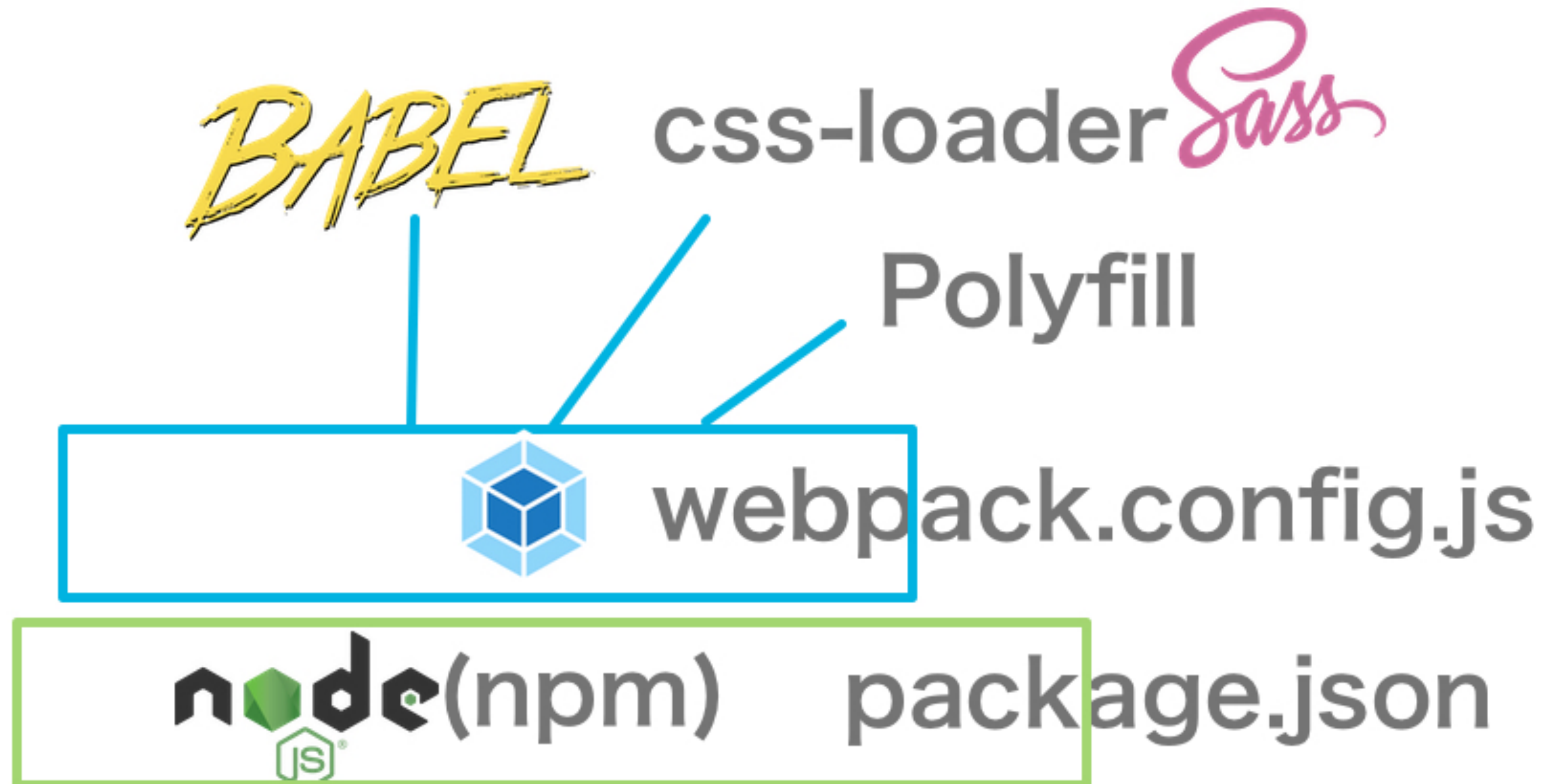
パフォーマンス向上(読み込み速度up)

開発環境向上(読みやすさ・メンテしやすさ)

新コード->旧コードへの変換



フロントエンドの環境(最低限)



Vuejsの環境構築方法

- Webpack + vue-loader
npm i -D vue-loader vue-template-compiler
- Vue-CLI (webpackのラッパー)



Vue CLI

Vue CLI



Command Line Interface

インストール

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

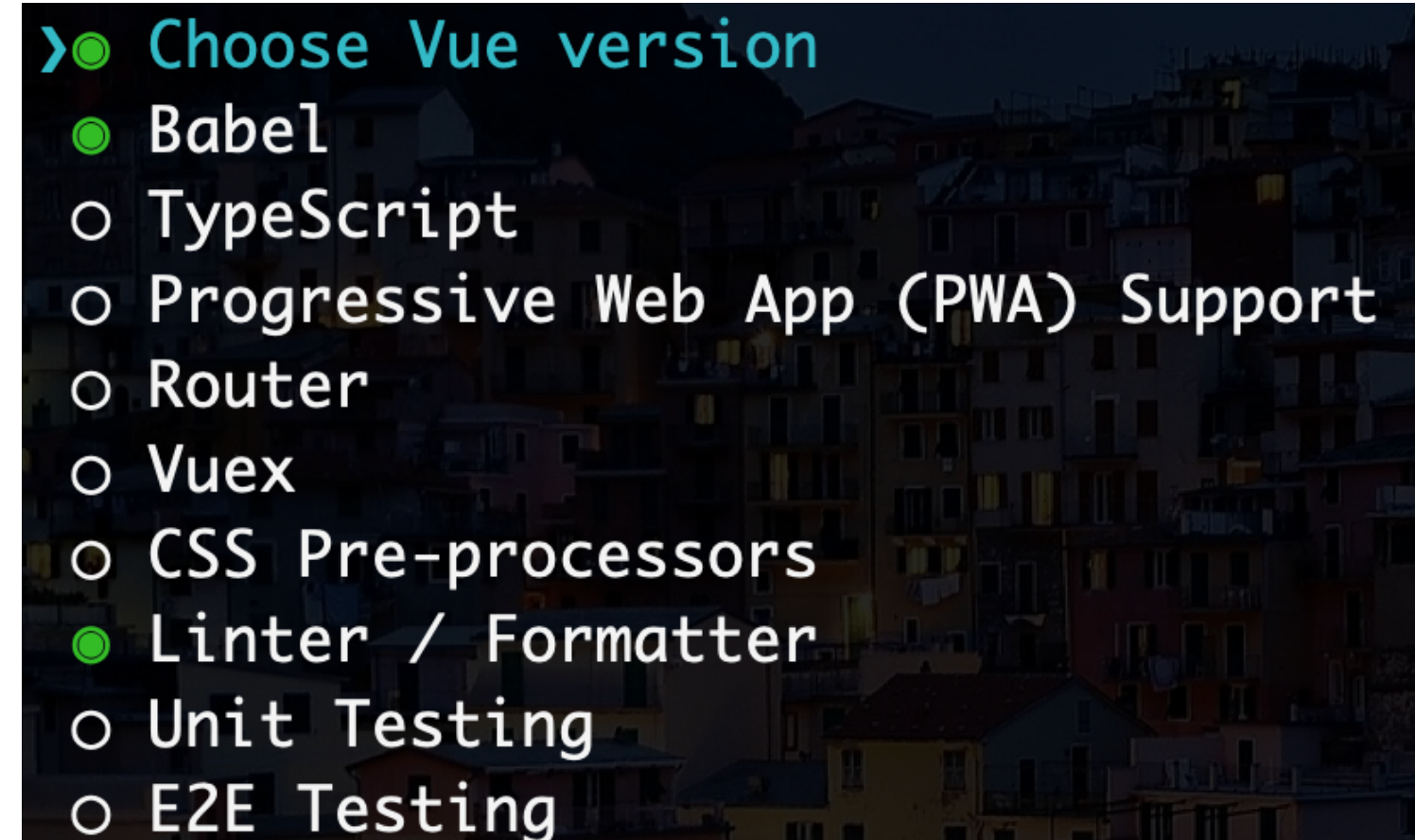
バージョン情報

```
vue --version
```

Vue CLI 新規作成

vue create xxx // CUI コマンド
vue ui // GUI グラフィック

VueVersion / Babel
/ TypeScript / PWA
/ Router / Vuex
/ CSS / Linter / Test

A screenshot of the Vue CLI create wizard terminal output. The background is a dark, stylized image of a city at night. The text is white and green. The first line is a green prompt character followed by 'Choose Vue version'. Below it is a list of options, each preceded by a green circle. The options are: Babel, TypeScript, Progressive Web App (PWA) Support, Router, Vuex, CSS Pre-processors, Linter / Formatter, Unit Testing, and E2E Testing. The 'Linter / Formatter' option is highlighted with a green circle.

```
> Choose Vue version
  Babel
  TypeScript
  Progressive Web App (PWA) Support
  Router
  Vuex
  CSS Pre-processors
  Linter / Formatter
  Unit Testing
  E2E Testing
```

Vue CLI 新規作成



npm run serve 簡易サーバー起動
->開発環境

npm run build ビルド
->本番環境 distフォルダに本番用ファイルが
生成される



Vuterインストール

VSCode プラグイン



Vetur ・ ・ Vueファイルを見やすく
<vueと書く と最低限のコードを作成

ESLint ・ ・ コード解析
Prettier ・ ・ コード整形



Vue CLI フォルダ構成

フォルダ・ファイル構成

node_modules ・ ・ 各種ライブラリ

dist ・ ・ コンパイル後のフォルダ

public ・ ・ テンプレート

src ・ ・ 開発フォルダ

main.js ・ ・ エントリーポイント

package.json ・ ・ npmの設定ファイル


vue.config.js ・ ・ vueの設定ファイル(要作成)

他環境でも実施するなら



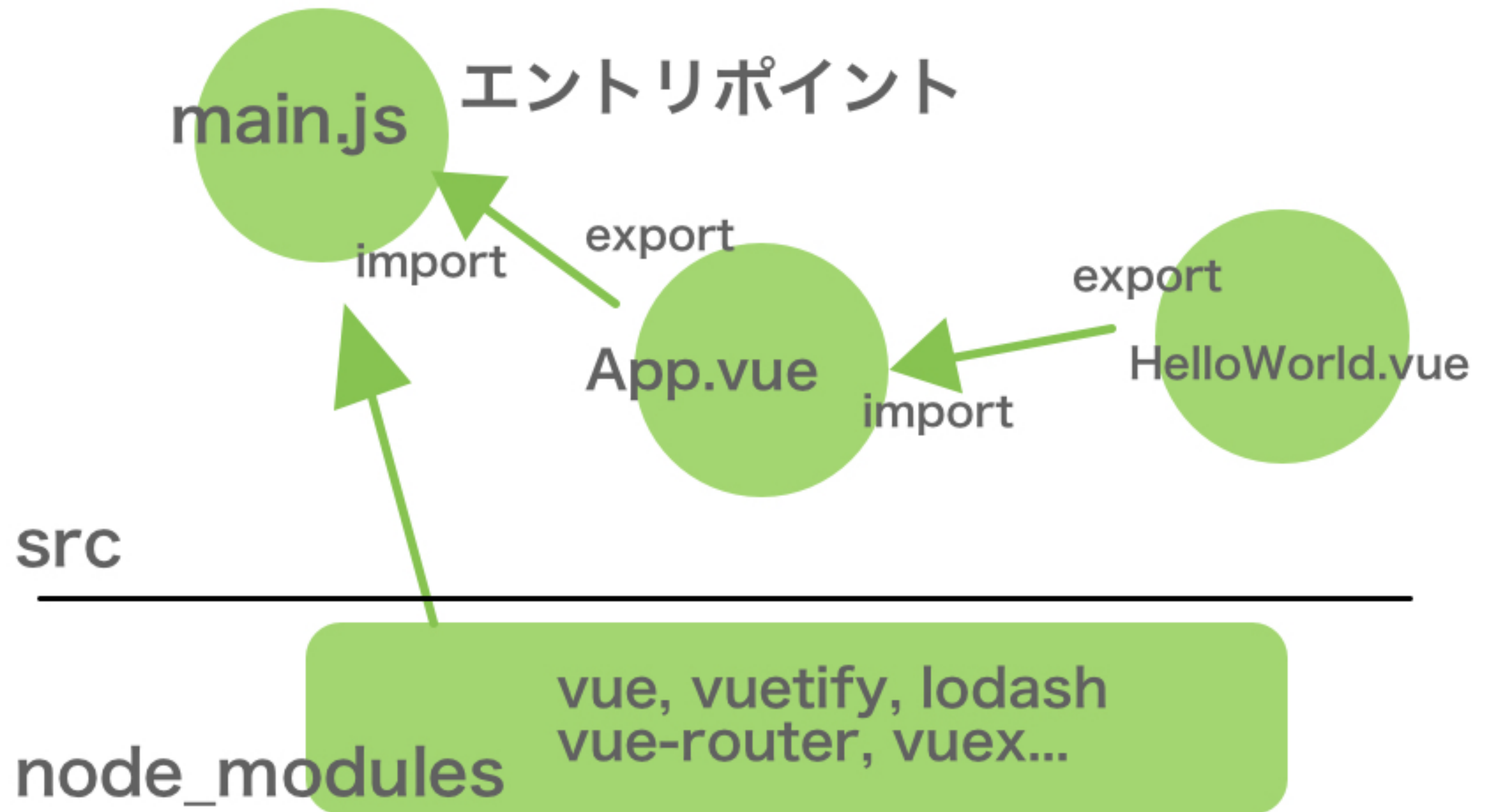
node_modules / dist は不要
gitHubなどにもこれらは含まない

Node.js / vue-cli インストール済みで
package.jsonがあれば
npm install で node_module生成
vue run build で dist 生成



エントリーポイント とVueファイル

初期の構成(import/export)



import



import TestCom from './TestCom.vue' 同じ階層
import TestCom from '../Folder/TestCom.vue'
一つ上のフォルダ

import TestCom from '@Folder/TestCom.vue'
絶対パス (@はsrcの意味)

SFCはファイル名パスカルケース、単語2文字以上
import名もファイル名と同じにしておく

render, \$mount, h



render

templateの代わり。関数でHTMLを書ける

\$mount elの代わり。

仮想DOMにあたる箇所を指定。 (#appなど)

h hyperscriptからきている、仮想DOM実装で使われる

https://qiita.com/teinen_qiita/items/ed1bb1909a17f9ca9daa

Vueファイル (SFC)



```
<template></template>
<script>
import xxx from 'xxx.vue'

export default{
  name: 'yyy',
  components: {
    xxx}
  }
</script>
<style></style>
```

Scoped CSS



`<style></style>` //グローバル 非推奨

これで書くならBEMなど

他コンポーネントと重ならないように整理要

`<style scoped></style>` //ローカル

`<style lang="scss"></style>`

追加設定必要



vue.config.js

vue.config.js vueの設定ファイル

```
module.exports = {  
  publicPath: '' // distのパス変更  
}
```




補足1 SCSS

VueCliでScss



vue create時に追加するか

npmで後から追加

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

各コンポーネントで書く場合


```
<style lang="scss">
```

```
</style>
```

VueCliでScss グローバル

src/assets/sass/main.scss を作成したとして
vue.config.js に下記を追記

```
module.exports = {  
  css:{  
    loaderOptions:{  
      scss:{  
        additionalData: `@import "@/assets/sass/main.scss";`  
      }  
    }  
  }  
}
```



補足2 マルチページ モード

マルチページモード

エントリーポイントをページ毎に作成

vue.config.js に

```
pages{ index: {
```

```
  entry:xxx, template:xxx, filename:xxx,  
  title:xxx, chunks:xxx }
```

などとページ毎に記載

npm install —save-dev html-webpack-plugin

preload-webpack-plugin

vue inspect でエラーないか確認