React Tooling

Development and Production Setup

NODESEN.SE

Dependencies

- Node.js (6.11 or 8.9 LTS is good)
- NPM - Node Package Manager [Comes along with Node.js]

Create Project

```
> mkdir  react-app
> cd react-app
```

```
> npm  init -y
```

Creates **package.json** file, that shall have project dependencies

Install React, React-DOM

```
>   npm install   react   react-dom   --save
```

download packages from registry.npm.org

react specific packages

update package.json

package.json

```
"dependencies": {
  "react": "^15.4.2",
  "react-dom": "^15.4.2"
}
```

React Package

```
import React, {Component,
               PureComponent} from "react";
```

Contains classes needed for createElement, Component, PureComponents, the core library for react implementation. Creates and returns Virtual DOM on render Methods

React-DOM Package

```
import {render} from “react-dom”;
```

```
render( <App>
        </App>
        , document.getElementById("root"))
```

- Bootstrap react component into Browser DOM
- Manages Diffs between real and virtual DOM

NODESEN.SE

prop-types

```
>   npm install   prop-types --save
```

```
import PropTypes from "prop-types";
```

```
Address.propTypes = {
  pincode: PropTypes.number.isRequired
}
```

Useful for describing component props, context types, mandating required properties

React Router

> npm install react-router-dom —save

```
import {BrowserRouter,
        HashRouter,
        Route,
        Switch,
        NavLink } from "react-router-dom";
```

- Handling routing in the react app

Redux

```
> npm install redux —save
```

```
import {createStore} from “redux";
```

- State management library for managing application data

Redux-Thunk

> npm install redux-thunk —save

```
import thunk from “redux–thunk“;
```

- Managing Async Actions, like reading data from web service, storing the values to store

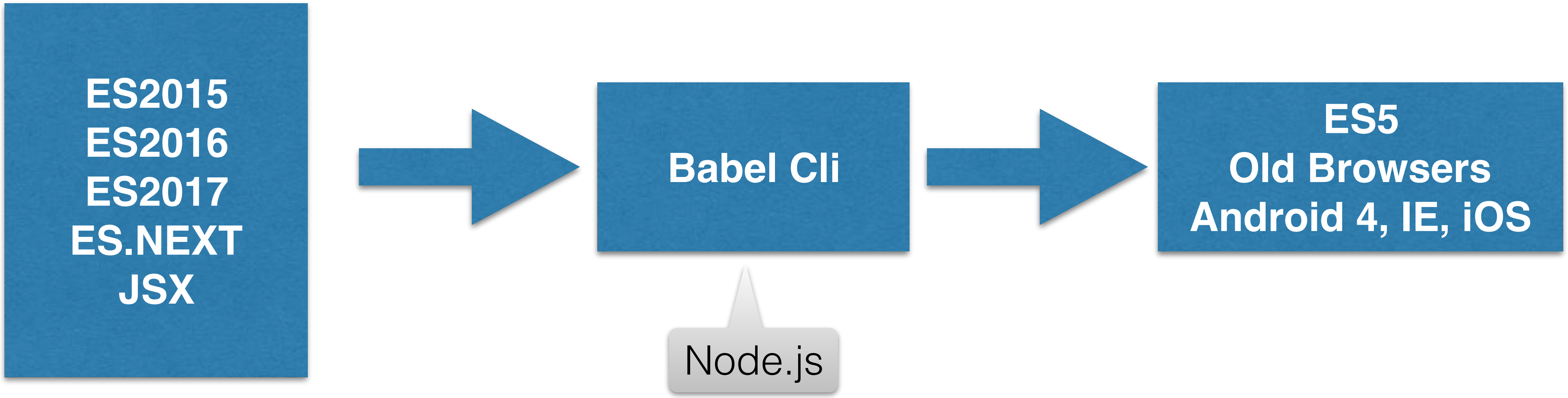
Development Dependencies

NODESEN.SE

ES6

- How to use new JavaScript ECMAScript 2015/ES6?
- Not all browsers support ES6, ES7, ES8, ES.NEXT
- Babel Transpiler

```
> npm install --save-dev babel-cli  -g
```



NODESEN.SE

Babel Presets ES6+

> npm install babel-preset-env —save-dev

- A Plugin for babel transpiler,
- Convert ES6 to ES5
- ES 2016 (ES7) to ES5
- ES 2017 to ES5
- To be stored in **.babelrc file**

```
{
     "presets": [“env”]
}
```

.babelrc

Babel Presets ES.NEXT

> npm install babel-preset-stage-2 —save-dev

- ES.NEXT (upcoming stage-2 language features to project)

.babelrc

```
{
    "presets": ["env",
                "stage-2"]
}
```

Static Variable inside
Inside class

Stage-3 presents
Dynamic import

https://babeljs.io/docs/plugins/preset-stage-2/

NODESEN.SE

JavaScript XML (JSX)

```
class App extends React.Component {
  render () {
   return  (
          <p> Hello React!</p>
   )}
}

render(<App/>,

          document.getElementById('root'));
```

XML
Inside JS code

Converted to
JS code later

NODESEN.SE

JSX Babel Presets

> npm install babel-preset-react —save-dev

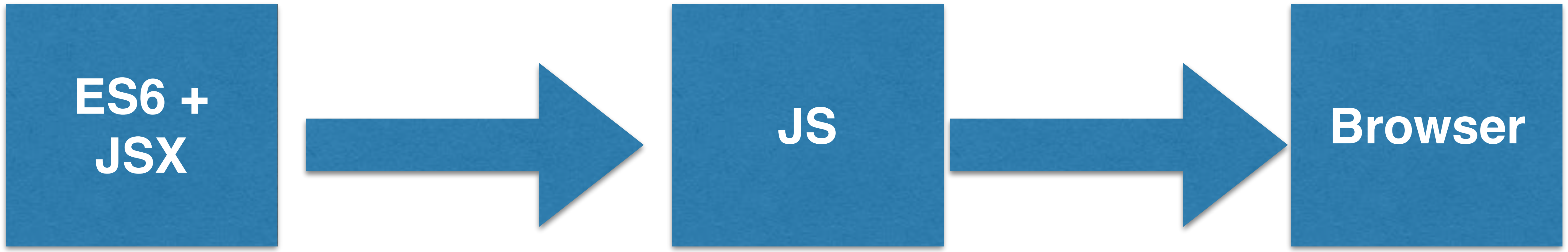
- A Plugin for babel transpiler, that converts JSX to JavaScript Code
- Used as command line option for babel-cli
- Or Often used with **.babelrc file**

.babelrc

```
{
    "presets": [“env”, “stage-2”, “react”]
}
```

NODESEN.SE

Babel Preset



babel
react-preset

Node.js
Development

Browser
No JSX

NODESEN.SE

Webpack

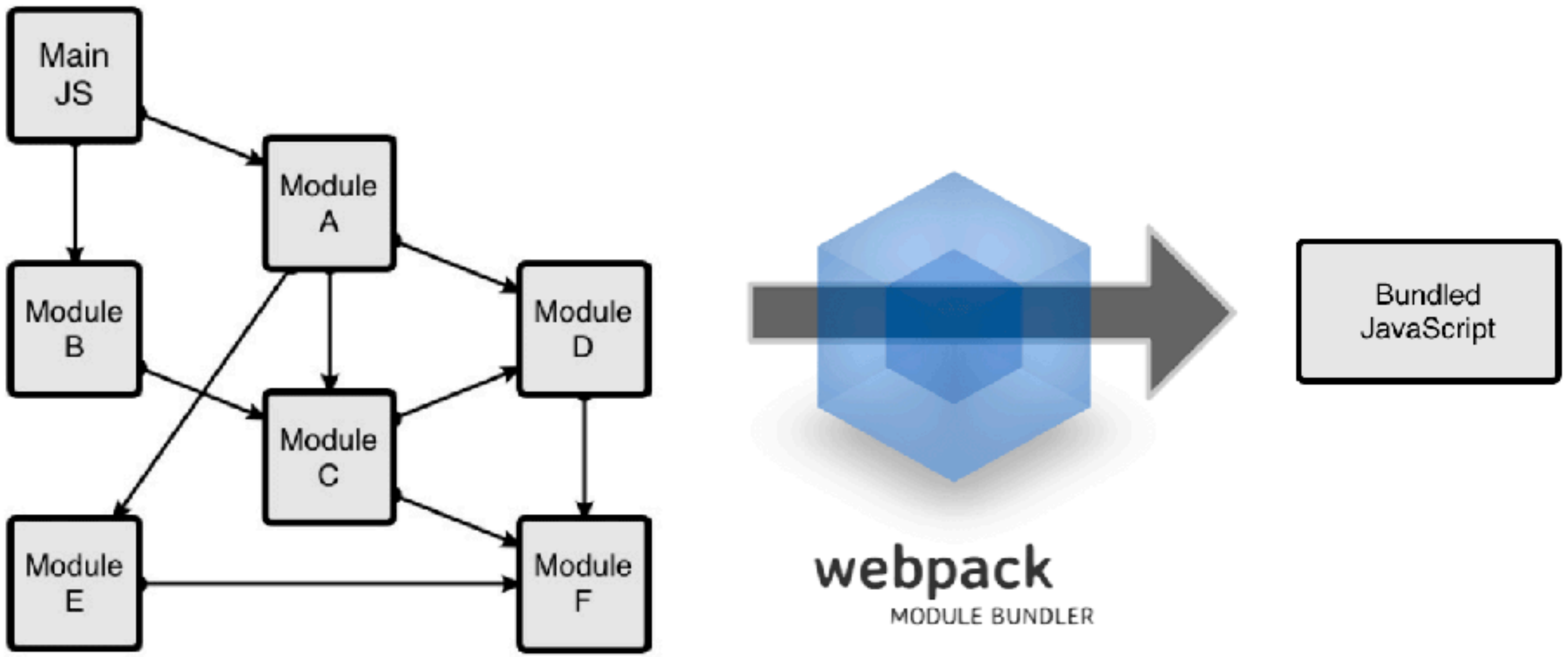
- The Module Bundler
- Put all JavaScript files into one single bundle
- Webpack can split large bundle into small bundles
- Easy for browser to download javascript, css files

Web Pack

```
> npm install webpack —save-dev
```

- Creates the bundle of JavaScript Files
- A De-facto module bundler for JavaScript
- Plug-ins for minification, code splitting
- Plugs for assets file synching with distribute
- Source map for debugging

Module refers to a single JavaScript file



Development Environment

Browser

Web Pack

Create a webpack.config.js for development mode

```
var webpack = require('webpack');
var path = require(‘path');
var APP_DIR = path.resolve(__dirname, 'src');
var BUILD_DIR = path.resolve(__dirname, 'dist');

var config = {
  entry: {
    app: APP_DIR + '/main.js'
  },

 output: {
    path: BUILD_DIR
  },

  devtool: 'source-map'
};

module.exports = config;
```

Entry file for bundle

Output bundle file stored in Dist

Generate .map for debugging

NODESEN.SE

Environments

- Three Options
- Development Environment
- Production Environment
- Optionally Testing Environment

Development Environment

- Webpack-Dev-Server
- Re-bundle files on file save
- Serve Files to Browser on 8080 or other ports
- Reload Web page when developer save files and automatic refreshment
- Hot Module Loading

WebPack-Dev-Server

```
> npm install webpack-dev-server —save-dev
```

- Generate bundle files in memory only
- Serve files from in-memory
- Hot Module Runtime, send modified file(s) to browser, instead of sending entire bundle

Webpack Dev Server

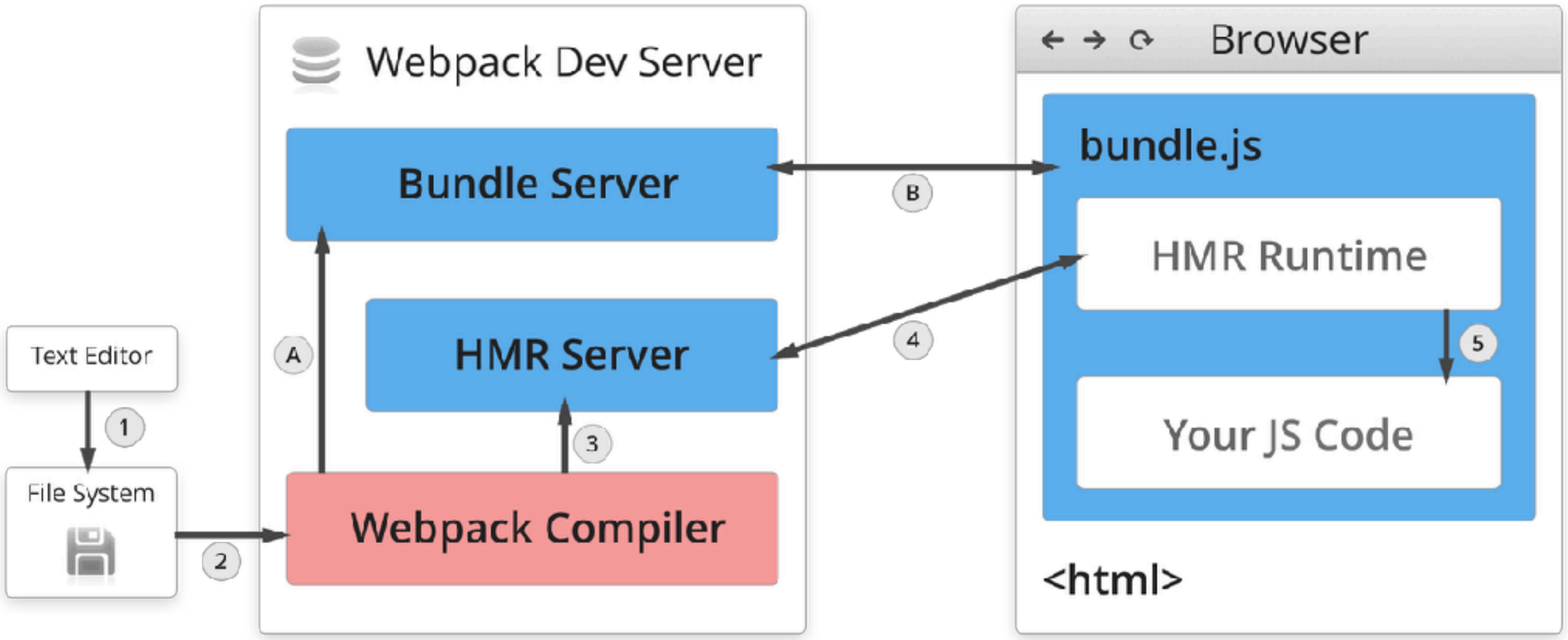


Image credit: http://andrewhfarmer.com/

NODESEN.SE

Babel-Loader

> npm install babel-core babel-loader —save-dev

- Babel Loader is a web pack loader module
- It loads JavaScript files, process them
- Babel-Loader to convert es6 files to es5 files
- Babel core is core library used by babel-loader

```
module : {
    loaders : [
      {
        test : /\.js?/,
        include : APP_DIR,
        loaders: [ "babel-loader"]
      }
    ]
  }
```

Web pack config file
Loads files with extension .js, .jsx

NODESEN.SE

Webpack npm script

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

> npm run build

Creates a bundles, copies the files into “dist” folder

Webpack-dev-server npm script

```
"scripts": {
    ...
    "start": “webpack–dev–server ––config
webpack.config.js ––inline ––hot ––open”,
    ...
  }
```

> npm start

Starts webpack-dev-server on port 8080 or available one

NODESEN.SE

webpack-dev-server

```
webpack-dev-server --config
webpack.config.js --inline --hot --open
```

- -- inline include web pack runtime, websocket, browser sync
- —hot enable hot module runtime, swap modified files in dev server with browser, faster reload time
- — open just open the browser

Webpack Babel

>   npm install babel-loader --save-dev

```
// Existing Code ....
var config = {
  // Existing Code ....
  module : {
    loaders : [
      {
        test : /\.js?/,
        include : APP_DIR,
        loader : ‘babel-loader'
      }
    ]
  }
}
```

Fetch

For **Ajax** Needs, Fetch is getting into Browsers, replacement for XMLHttpReqequest, provides higher level abstraction for Ajax support, eliminate needs for 3rd party library support

https://developer.mozilla.org/en/docs/Web/API/Fetch_API

Fetch Polyfill

All latest Browsers support fetch api

For older browser, we need polyfill

https://github.com/github/fetch

```
> npm install whatwg-fetch —save
```

In code, typically in main.js/index.js files

```
import “whatwg-fetch”
```