Modern Web Stack

Visual Studio Code

Free, Open Source, lightweight cross plattform Editor - Built on top of GitHubs Electron platform using TypeScript

Out-of-box integration of GitHub & Node.js

Optimized to enable almost any kind of Application and Task

- HTML, TS, JS, C#, Java, Angular, PowerShell, Phyton, …
- Docker, DevOps Pipelines, Azure, …
- SPFx, Teams, Office Add-Ins, …

Extensions (Plugins) for almost anything

Get from https://code.visualstudio.com/

Visual Studio Code

Node.js – Development time hosting

- Development-time hosting platform & Local JavaScript runtime environment
  - Can be considered as the IIS Express in typical Microsoft stack
- Can also work as backend system with server-side code, if needed
- Using differen Node Versions on the same machine
  - nvm-windows supports running several Node versions on the same windows machine
    - https://github.com/coreybutler/nvm-windows
    - nvm install 6.3.1
    - nvm use 6.3.1
    - For mac / linux use : http://nvm.sh/

Nodes Role in modern Web Stack Dev

Node.js accts as a Runtime Host for our Dev Toolset

It provides:

- Package Management
- TS /Sass Compilation
- Bundling
- …

npm – Node Package Manager

- npm is a package manager for JavaScript
- Used with Node.js for local JavaScript development
- Downloads 3rd party libraries used in your customizations
  - You can also share your own packages in npm for the benefit of others

Global Installation -> Tools

- npm install xxx –g

https://www.npmjs.com/

package.json

package.json is the configuration file for node.js

packages are saved to node_modules

Installation in the local project

- npm install xxx -S (Runtime Dependency) or -D (Dev Dependency)

```
{
  "version": "1.0.0",
  "name": "asp.net",
  "private": true,
  "dependencies": {
    "gulp-sourcemaps": "^2.4.1",
    "lodash": "^4.17.4",
    "webpack": "^2.6.1",
    "babel-core": "^6.2.1",
    "babel-loader": "^6.2.0",
    "babel-preset-es2015": "^6.1.18"
  },
  "devDependencies": {
    "babel-preset-es2015": "^6.24.1",
    "gulp": "3.9.1",
    "gulp-babel": "^6.1.2",
    "gulp-concat": "2.6.0",
    "gulp-cssmin": "0.1.7",
    "gulp-sass": "^3.1.0",
    "gulp-sourcemaps": "^2.6.0",
    "gulp-uglify": "1.5.3"
  }
}
```

Gulp

Gulp is a Plugin based Task Runner

- SPSave can be used to upload to SharePoint

Gulp can be used to automate

- Bundling, minification or
- the cleansing of a development environment

Tasks are defined in gulpfile.js


```
var spsaveCoreOption = {
    siteUrl: 'http://sp2019',
    folder: 'Shared%20Documents',
    checkin: true,
    checkinType: 1,
    flatten: false,
    notification: true
};

var spsaveCredential = {
    username: 'spdom\\administrator',
    password: 'Pa$$w0rd'
};

gulp.task('upload', function() {
    return gulp.src('./dist/**/*').pipe(spsave(spsaveCoreOption, spsaveCredential));
});
```

Babel

JavaScript downlevel Compiler

- Ensures Browser Compatibility when combined with Polyfills

Can be automated using Gulp, Webpack, ...

Documentation at https://babeljs.io/

Sass Compilation

Sass Compilation is hosted by Node.js

Compilation makes Sass avilable to Browsers

Can be automated using Watchers and:

- Webpack
- Gulp


```
gulp.task('compile:sass', function() {
    gulp.src(paths.scss).pipe(sass()).pipe(gulp.dest(paths.dist));
});
```

Webpack

A module bundler for the static assets in your web application

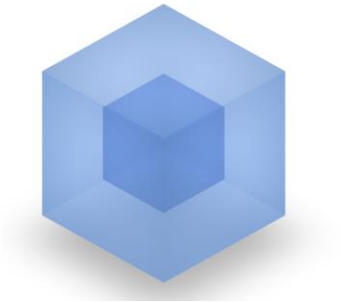
Bundles:

- JavaScript, Typescript,
- CSS, Images
- …

Transforms & creates bundles provided to the browser

Supports Source-Maps & watch-mode for dynamic recompilation:

- webpack --watch



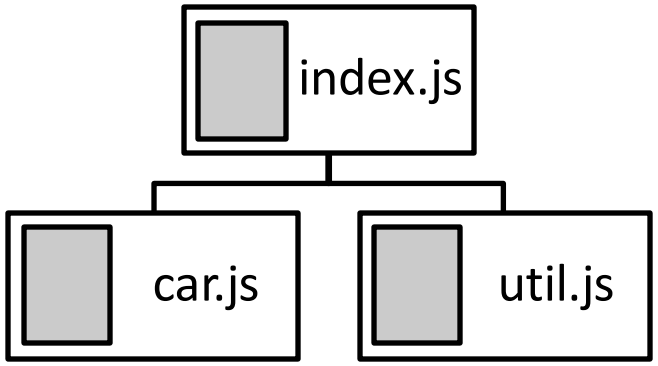
Dependency Graphs

require keyword is used to build dependency graphs

Use import when writing ES 6 / Typescript instead

Example:

- require(' './wwwroot/app.js ');




```
import { SPUtil } from "./util";
import { Car } from "./car";

let spUtil = new SPUtil();
spUtil.log();

spUtil
  .getWebTitleCSOM()
  .then(data => console.log(`Web Title received by CSOM is: ${data}`));

spUtil.getWebTitleREST();
```

webpack.config.js

Defines the Webpack Build

- __dirname refers to the directory where this webpack.config.js lives
- Entry: entry file of the app
- Output: file to generate
- Watch: use watcher

TS to JS transpilation is done using the TS-Loader Plugin

```
const path = require("path");

module.exports = {
  entry: "./src/index",
  mode: "development",
  devtool: "inline-source-map",
  module: {
    rules: [
      {
        test: /\.ts$/,
        use: "ts-loader"
      }
    ]
  },
  resolve: {
    extensions: [".ts"]
  },
  output: {
    filename: "bundle.js",
    path: path.resolve(__dirname, "./dist/")
  }
};
```