Cheat Sheet for comprehensive Webpack

Installation and Setup

- Global Installation

```
npm install -g webpack webpack-cli
```

- Local Installation

```
npm install --save-dev webpack webpack-cli
```

- Basic Configuration File

- Create `webpack.config.js` in the root directory.

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

module.exports = {
  entry: './src/index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```

Core Concepts

- Entry

- Specifies the entry point of the application.

```
entry: './src/index.js'
```

- Output

- Defines where the bundled files will be output.

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

- **Loaders**

- Transform files (e.g., CSS, images) into modules.

```
module: {
  rules: [
    { test: /\.css$/, use: ['style-loader', 'css-loader'] },
    { test: /\.(png|svg|jpg|gif)$/, use: ['file-loader'] },
  ],
}
```

- **Plugins**

- Perform broader tasks like optimization, asset management, etc.

```
const HtmlWebpackPlugin = require('html-webpack-plugin');

plugins: [
  new HtmlWebpackPlugin({ template: './src/index.html' }),
]
```

Configuration Examples

- **Multiple Entry Points**

```
entry: {
  app: './src/app.js',
  admin: './src/admin.js',
}
```

- **Code Splitting**

```
optimization: {
  splitChunks: {
    chunks: 'all',
  },
}
```

- **Environment-Specific Configurations**

- Use `webpack-merge` for merging common and environment-specific configs.

```
const { merge } = require('webpack-merge');
const common = require('./webpack.common.js');

module.exports = merge(common, {
  mode: 'development',
  devtool: 'inline-source-map',
});
```

Development Tools

- DevServer

- Provides a development server with live reloading.

```
devServer: {
  contentBase: './dist',
  hot: true,
}
```

- Hot Module Replacement (HMR)

- Updates modules in the browser without a full reload.

```
plugins: [
  new webpack.HotModuleReplacementPlugin(),
]
```

Optimization

- Minification

- Minifies the output bundle.

```
optimization: {
  minimize: true,
  minimizer: [new TerserPlugin()],
}
```

- Tree Shaking

- Removes unused code.

```
  optimization: {
    usedExports: true,
  }
```

Loaders

- **CSS**

```
  { test: /\.css$/, use: ['style-loader', 'css-loader'] }
```

- **SCSS/SASS**

```
  { test: /\.scss$/, use: ['style-loader', 'css-loader', 'sass-loader']
}
```

- **Images**

```
  { test: /\.(png|svg|jpg|gif)$/, use: ['file-loader'] }
```

- **Fonts**

```
  { test: /\.(woff|woff2|eot|ttf|otf)$/, use: ['file-loader'] }
```

Plugins

- **HTML Webpack Plugin**

```
  const HtmlWebpackPlugin = require('html-webpack-plugin');

  plugins: [
    new HtmlWebpackPlugin({ template: './src/index.html' }),
  ]
```

- **Clean Webpack Plugin**

```
  const { CleanWebpackPlugin } = require('clean-webpack-plugin');

  plugins: [
    new CleanWebpackPlugin(),
  ]
```

- **Mini CSS Extract Plugin**

```
const MiniCssExtractPlugin = require('mini-css-extract-plugin');

plugins: [
  new MiniCssExtractPlugin({
    filename: '[name].[contenthash].css',
  }),
]
```

CLI Commands

- **Build for Production**

```
webpack --mode production
```

- **Build for Development**

```
webpack --mode development
```

- **Watch Mode**

```
webpack --watch
```

- **Serve with DevServer**

```
webpack serve --open
```

Advanced Features

- **Lazy Loading**

• Load modules only when they are needed.

```
import(/* webpackChunkName: "lodash" */ 'lodash').then(_ => {
  console.log(_.join(['Hello', 'webpack'], ' '));
});
```

- **Environment Variables**

- Use environment variables in your configuration.

```
webpack --env.production



module.exports = env => {
  return {
    mode: env.production ? 'production' : 'development',
  };
};
```

- **Custom Webpack Configuration**

- Extend Webpack's capabilities with custom configurations.

```
const config = {
  // Your custom configuration
};

module.exports = (env, argv) => {
  if (argv.mode === 'development') {
    config.devtool = 'source-map';
  }
  return config;
};
```

Tips and Tricks

- **Use `webpack-bundle-analyzer`**

- Analyze the size of your bundles.

```
npm install --save-dev webpack-bundle-analyzer



  const BundleAnalyzerPlugin = require('webpack-bundle-
analyzer').BundleAnalyzerPlugin;

  plugins: [
    new BundleAnalyzerPlugin(),
  ]
```

- **Cache Busting**

- Use content hashes to cache bust.

```
output: {
  filename: '[name].[contenthash].js',
}
```

- **Use `webpack-dev-middleware`**

- Integrate Webpack with an existing Node.js server.

```
const webpack = require('webpack');
const middleware = require('webpack-dev-middleware');
const compiler = webpack(config);

app.use(middleware(compiler, {
  publicPath: config.output.publicPath,
}));
```

Common Issues and Solutions

- **Module Not Found**

- Ensure all paths are correct and dependencies are installed.

- **HMR Not Working**

- Check if `HotModuleReplacementPlugin` is added and `devServer.hot` is set to `true`.

- **Slow Builds**

- Optimize loaders and use `thread-loader` for parallel processing.

Resources

- **Official Documentation**

- [Webpack Documentation](https://webpack.js.org/concepts/)

- **Community Plugins**

- [Awesome Webpack](https://github.com/webpack-contrib/awesome-webpack)

- **Tutorials**

- [Webpack Guides](https://webpack.js.org/guides/)

This cheat sheet provides a comprehensive overview of Webpack, covering installation, core concepts, configuration examples, development tools, optimization techniques, loaders, plugins, CLI commands, advanced features, tips and tricks, common issues, and additional resources.

By Ahmed Baheeg Khorshid

ver 1.0