hot module replacement

> 此功能允许在运行时更新所有类型的模块，而无需完全刷新，适用于开发环境

使用webpack-dev-server开启HMR

- 在webpack.config.js中引入webpack
- 在devServer配置中配置hot选项
- 在插件plugins中应用webpack自带的HotModuleReplacementPlugin

```
//webpack.config.js

const path=require('path');
const HtmlWebpackPlugin=require('html-webpack-plugin')
const {CleanWebpackPlugin}=require('clean-webpack-plugin')
const webpack=require('webpack');
module.exports={
    mode:'development',//默认produnction,decelopment||none,设置为development时,
    entry:'./src/index.js',
    devServer:{
        open:true,
        hot:true,//开启hot module replacement
        hotOnly:true,//即便html功能没生效，浏览器也不自动刷新
    },
    output:{
        filename:'dist.js',//打包后的文件名，默认是main.js
        path:path.resolve(__dirname,'dist'),
    },
    plugins:[
            new webpack.HotModuleReplacementPlugin(),
            new HtmlWebpackPlugin({
                template:'src/index.html'
            }),
            new CleanWebpackPlugin()],
}
```

- 修改入口文件，传入更新的模块，在这些模块修改后，告诉webpack接受updated module

```
//index.js
import counter from './counter.js';
import number from './number.js';
counter();
number();
//如果当前模块开启了hot module replacement,接受的number如果发生变化，移除之前的number,number重新执行
if(module.hot){
    module.hot.accept('./number',function(){
        document.body.removeChild(document.getElementById('number'));
        number();
    });
}
```

通过node.js API开启HMR

使用webpack-dev-middleware和webpack-hot-middleware

- 配置server.js,并使用webpack-dev-middleware和webpack-hot-middleware
- 修改webpack.config.js,删除derServer配置，将此配置对象写在server.js中，并使用HotModuleReplacementPlugin
- 在webpack.config.js中配置开启HMR的入口起点

```
//webpack.config.js
//设置开启HMR的入口起点
entry:{
    main: ['webpack-hot-middleware/client', './src/index.js']
},
//在plugins选项中添加热更新模块
 plugins:[
            new webpack.HotModuleReplacementPlugin(),
            new HtmlWebpackPlugin({
                template:'src/index.html'
            }),
            new CleanWebpackPlugin()
        ],

--------------------------------------------------------------------------------
//server.js
//引入webpack-dev-middleware
const webpackHotMiddleware=require('webpack-hot-middleware');
const express=require('express');
const webpack= require('webpack');
const webpackMiddleware=require('webpack-dev-middleware');
//引入webpack的配置文件
const config=require('./webpack.config.js');
const complier=webpack(config);
//服务器实例
const app=express();
//使用中间件,只要文件变化，compiler就会重新编译，打包输出的路径就是config文件下的output的publicPath
app.use(webpackMiddleware(complier));
//使用热更新模块
app.use(webpackHotMiddleware(complier));
app.listen(8080,()=>{
    console.log('server is running');
});
```

使用webpack-dev-server

- 将webpack.config.js的devServer配置去除，写在server.js中
- 不需要再webpack.config.js中使用HotModuleReplacementPlugin
- 配置server.js
- webpack-dev-server package 中具有一个叫做addDevServerEntrypoints 的方法,想要启用 HMR，使用该方法添加 HMR 入口起点。

```
//使用webpack dev server
const webpackDevServer=require('webpack-dev-server');
const webpack=require('webpack');
const config=require('./webpack.config.js');
const option={
    contentBase:'./dist',
    hot:true,
    host:'localhost'
};
/**
 * webpack-dev-server package 中具有一个叫做addDevServerEntrypoints 的方法
 * 想要启用 HMR，使用该方法添加 HMR 入口起点。
*/
webpackDevServer.addDevServerEntrypoints(config,option);
const compiler=webpack(config);
const server=new webpackDevServer(compiler,option);
server.listen(5000,'localhost',()=>{
    console.log('dev server listening on port 5000');
});
```