link: null
title: 珠峰架构师成长计划
description: webpack.config.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=184 sentences=764, words=3853

1. 什么是HMR

- Hot Module Replacement是指当我们对代码修改并保存后，webpack将会对代码进行重新打包，并将新的模块发送到浏览器端，浏览器用新的模块替换掉旧的模块，以实现在不刷新浏览器的前提下更新页面

2.使用HMR

2.1 安装

```
yarn add webpack webpack-cli webpack-dev-server html-webpack-plugin socket.io socket.io-client events mime fs-extra --dev
```

2.2 使用

2.2.1 webpack.config.js

webpack.config.js

```
let path = require("path");
let webpack = require("webpack");
let HtmlWebpackPlugin = require("html-webpack-plugin");
let HotModuleReplacementPlugin = require('webpack/lib/HotModuleReplacementPlugin');
module.exports = {
    mode: "development",
    entry:"./src/index.js",
    output: {
        filename: "[name].js",
        path: path.resolve(__dirname, "dist")
    },
    devServer:{
        hot:true,
        port:8000,
        contentBase:path.join(__dirname,'static')
    },
    plugins: [
        new HtmlWebpackPlugin({
            template:'./src/index.html'
        }),
        new HotModuleReplacementPlugin()
    ]
}
```

2.2.2 src\index.html

src\index.html

```

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>hmrtitle>
head>
<body>
    <input/>
    <div id="root">div>
body>
html>
```

2.2.3 src\index.js

src\index.js

```
let render = () => {
    let title = require("./title.js");
    root.innerText = title;
}
render();

if (module.hot) {
    module.hot.accept(["./title.js"], render);
}
```

2.2.4 title.js

src\title.js

```
module.exports = "title";
```

2.2.5 package.json

```
"scripts": {
  "build": "webpack",
  "dev": "webpack serve"
}
```

2.2.6 debugger

```
  "scripts": {
    "build": "webpack",
    "dev": "webpack serve",
+    "debug": "webpack serve"
  },
```

3.基础知识

3.1 module和chunk

- 在 webpack里有各种各样的模块
- 一般一个入口会依赖多个模块
- 一个入口一般会对应一个chunk,这个chunk里包含这个入口依赖的所有的模块

3.2 HotModuleReplacementPlugin

- webpack\lib\HotModuleReplacementPlugin.js
- 它会生成两个补丁文件
    - 上一次编译生成的hash.hot-update.json,说明从上次编译到现在哪些代码块发生成改变
    - chunk名字.上一次编译生成的hash.hot-update.js,存放着此代码块最新的模块定义,里面会调用 `webpackHotUpdate`方法
- 向代码块中注入HMR runtime代码,热更新的主要逻辑,比如拉取代码、执行代码、执行accept回调都是它注入的到chunk中的
- `hotCreateRequire`会帮我们给模块 module的 `parents`、 `children`赋值

3.3 webpack的监控模式

- 如果使用监控模式编译webpack的话,如果文件系统中有文件发生了改变,webpack会监听到并重新打包
- 每次编译会产生一个新的hash值

4.工作流程

4.1. 服务器部分

1. 启动webpack-dev-server服务器
2. 创建webpack实例
3. 创建Server服务器
4. 添加webpack的 `done`事件回调，在编译完成后会向浏览器发送消息
5. 创建express应用app
6. 使用监控模式开始启动webpack编译,在 webpack 的 watch 模式下，文件系统中某一个文件发生修改，webpack 监听到文件变化，根据配置文件对模块重新编译打包，并将打包后的代码通过简单的 JavaScript 对象保存在内存中
7. 设置文件系统为内存文件系统
8. 添加webpack-dev-middleware中间件
9. 创建http服务器并启动服务
10. 使用sockjs在浏览器端和服务端之间建立一个 websocket 长连接，将 webpack 编译打包的各个阶段的状态信息告知浏览器端,浏览器端根据这些 `socket`消息进行不同的操作。当然服务端传递的最主要信息还是新模块的 `hash`值，后面的步骤根据这一 `hash`值来进行模块热替换

步骤 代码位置 1.启动webpack-dev-server服务器
webpack-dev-server.js#L159 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/bin/webpack-dev-server.js#L83)

2.创建webpack实例
webpack-dev-server.js#L89 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/bin/webpack-dev-server.js#L89)

3.创建Server服务器
webpack-dev-server.js#L100 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/bin/webpack-dev-server.js#L107)

4.更改config的entry属性
webpack-dev-server.js#L157 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js#L57)

entry添加dev-server/client/index.js
addEntries.js#L22 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/utils/addEntries.js#L22)

entry添加webpack/hot/dev-server.js
addEntries.js#L30 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/utils/addEntries.js#L30)

5. setupHooks
   Server.js#L122 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js#L122)
6. 添加webpack的 `done`

事件回调
Server.js#L183 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js#L183)

编译完成向websocket客户端推送消息，最主要信息还是新模块的hash值，后面的步骤根据这一hash值来进行模块热替换
Server.js#L178 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js#L178)

7.创建express应用app
Server.js#L169 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js#L169)

8. 添加webpack-dev-middleware中间件
   Server.js#L208 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js#L208)

以watch模式启动webpack编译，文件系统中某一个文件发生修改，webpack 监听到文件变化，根据配置文件对模块重新编译打包
index.js#L41 (https://github.com/webpack/webpack-dev-middleware/blob/v3.7.2/index.js#L41)

设置文件系统为内存文件系统
index.js#L65 (https://github.com/webpack/webpack-dev-middleware/blob/v3.7.2/index.js#L65)

返回一个中间件，负责返回生成的文件
middleware.js#L20 (https://github.com/webpack/webpack-dev-middleware/blob/v3.7.2/lib/middleware.js#L20)

app中使用webpack-dev-middlerware返回的中间件
Server.js#L128 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js#L128)

9. 创建http服务器并启动服务
   Server.js#L135 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js#L135)
10. 使用sockjs在浏览器端和服务端之间建立一个 websocket 长连接
    Server.js#L745 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js#L745)

创建socket服务器并监听connection事件
SockJSServer.js#L33 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/servers/SockJSServer.js#L33)

4.2. 客户端部分

1. `webpack-dev-server/client-src/default/index.js`端会监听到此 `hash`消息,会保存此hash值
2. 客户端收到 `ok`的消息后会执行 `reloadApp`方法进行更新
3. 在reloadApp中会进行判断，是否支持热更新，如果支持的话发射 `webpackHotUpdate`事件,如果不支持则直接刷新浏览器
4. 在 `webpack/hot/dev-server.js`会监听 `webpackHotUpdate`事件,然后执行 `check()`方法进行检查
5. 在check方法里会调用 `module.hot.check`方法
6. 它通过调用 `JsonpMainTemplate.runtime`的 `hotDownloadManifest`方法，向 server 端发送 Ajax 请求，服务端返回一个 `Manifest`文件，该 `Manifest` 包含了所有要更新的模块的 `hash` 值和chunk名
7. 调用 `JsonpMainTemplate.runtime`的 `hotDownloadUpdateChunk`方法通过JSONP请求获取到最新的模块代码
8. 补丁JS取回来后会调用 `JsonpMainTemplate.runtime.js`的 `webpackHotUpdate`方法，里面会调用 `hotAddUpdateChunk`方法,用新的模块替换掉旧的模块
9. 然后会调用 `HotModuleReplacement.runtime.js`的 `hotAddUpdateChunk`方法动态更新模块代 码
10. 然后调用 `hotApply`方法进行热更新

步骤 代码 1.连接websocket服务器
socket.js#L25 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/client-src/default/socket.js#L25)

2.websocket客户端监听事件
socket.js#L53 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/client-src/default/socket.js#L53)

监听hash事件，保存此hash值
index.js#L55 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/client-src/default/index.js#L55)

3.监听ok事件，执行reloadApp方法进行更新
index.js#L93 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/client-src/default/index.js#L93)

4. 在reloadApp中会进行判断，是否支持热更新，如果支持的话发射 `webpackHotUpdate`

事件,如果不支持则直接刷新浏览器
reloadApp.js#L7 (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/client-src/default/utils/reloadApp.js#L7)

5. 在 `webpack/hot/dev-server.js`

会监听 `webpackHotUpdate`

事件
dev-server.js#L55 (https://github.com/webpack/webpack/blob/v4.39.1/hot/dev-server.js#L55)

6. 在check方法里会调用 `module.hot.check`

方法
dev-server.js#L13 (https://github.com/webpack/webpack/blob/v4.39.1/hot/dev-server.js#L13)

7. 调用 `hotDownloadManifest`

，向 server 端发送 Ajax 请求，服务端返回一个 Manifest文件(lastHash.hot-update.json)，该 Manifest 包含了本次编译hash值 和 更新模块的chunk名
HotModuleReplacement.runtime.js#L180 (https://github.com/webpack/webpack/blob/v4.39.1/lib/HotModuleReplacement.runtime.js#L180)

8. 调用 `JsonpMainTemplate.runtime` `hotDownloadUpdateChunk`

方法通过JSONP请求获取到最新的模块代码
JsonpMainTemplate.runtime.js#L14 (https://github.com/webpack/webpack/blob/v4.39.1/lib/web/JsonpMainTemplate.runtime.js#L14)

9. 补丁JS取回来后会调用 `JsonpMainTemplate.runtime.js` `webpackHotUpdate`

方法
JsonpMainTemplate.runtime.js#L8 (https://github.com/webpack/webpack/blob/v4.39.1/lib/web/JsonpMainTemplate.runtime.js#L8)

10. 然后会调用 `HotModuleReplacement.runtime.js` `hotAddUpdateChunk`

方法动态更新模块代码
HotModuleReplacement.runtime.js#L222 (https://github.com/webpack/webpack/blob/v4.39.1/lib/HotModuleReplacement.runtime.js#L222)

11.然后调用 `hotApply`

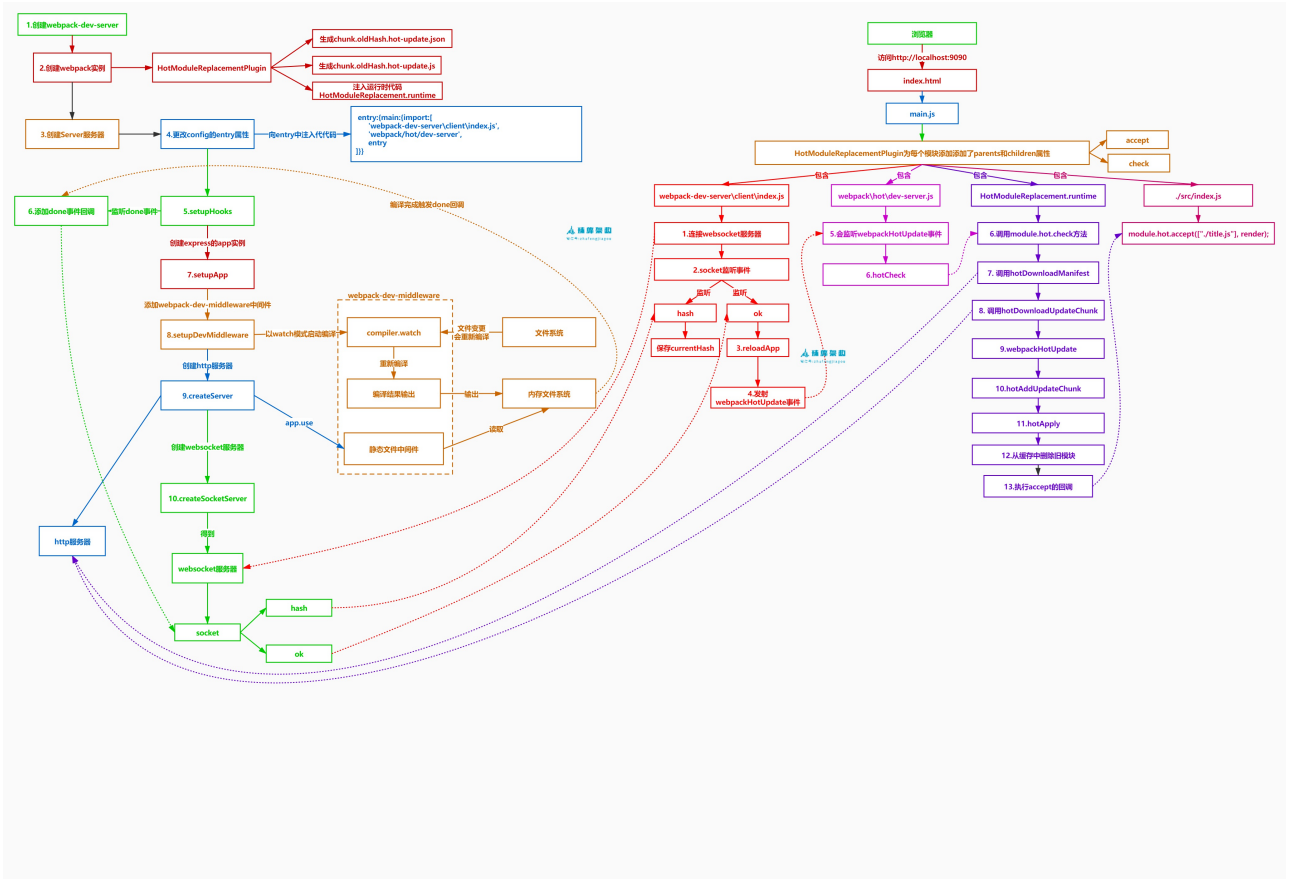
方法进行热更新
HotModuleReplacement.runtime.js#L257 (https://github.com/webpack/webpack/blob/v4.39.1/lib/HotModuleReplacement.runtime.js#L257) HotModuleReplacement.runtime.js#L278 (https://github.com/webpack/webpack/blob/v4.39.1/lib/HotModuleReplacement.runtime.js#L278)

12.从缓存中删除旧模块
HotModuleReplacement.runtime.js#L510 (https://github.com/webpack/webpack/blob/v4.39.1/lib/HotModuleReplacement.runtime.js#L510)

13.执行accept的回调
HotModuleReplacement.runtime.js#L569 (https://github.com/webpack/webpack/blob/v4.39.1/lib/HotModuleReplacement.runtime.js#L569)

4.3 相关代码

- webpack-dev-server.js (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/bin/webpack-dev-server.js)
- Server.js (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/Server.js)
- webpack-dev-middleware/index.js (https://github.com/webpack/webpack-dev-middleware/blob/v3.7.0/index.js)
- SockJSServer.js (https://github.com/webpack/webpack-dev-server/blob/v3.7.2/lib/servers/SockJSServer.js)



5.启动开发服务器

5.1 startDevServer.js

startDevServer.js

```
const webpack = require("webpack")
const Server = require('./webpack-dev-server/lib/Server');
const config = require("./webpack.config")
function startDevServer(compiler,options) {
    const devServerOptions = options.devServer||{};
    const server = new Server(compiler, devServerOptions);
    const {host='localhost',port=8080}=devServerOptions;
    server.listen(port, host, (err) => {
        console.log(`Project is running at http://${host}:${port}`);
    });
}
const compiler = webpack(config);
startDevServer(compiler,config);
```

5.2 Server.js

webpack-dev-server\lib\Server.js

```
const express = require("express");
const http = require("http");
class Server {
    constructor(compiler,devServerOptions) {
        this.compiler = compiler;
        this.devServerOptions = devServerOptions;
        this.setupApp();
        this.createServer();
    }
    setupApp() {
        this.app = new express();
    }
    createServer() {
        this.server = http.createServer(this.app);
    }
    listen(port, host = "localhost", callback = ()=>{}) {
         this.server.listen(port, host, callback);
    }
}
module.exports = Server;
```

5.3 package.json

package.json

```
  "scripts": {
    "build": "webpack",
    "dev": "webpack-dev-server",
+   "start":"node ./startDevServer.js"
  },
```

6.给entry添加客户端

6.1 Server.js

webpack-dev-server\lib\server\Server.js

```
const express = require("express");
+const updateCompiler = require('./utils/updateCompiler');
const http = require("http");
class Server {
    constructor(compiler,devServerOptions) {
        this.compiler = compiler;
        this.devServerOptions = devServerOptions;
+       updateCompiler(compiler);
        this.setupApp();
        this.createServer();
    }
    setupApp() {
        this.app = new express();
    }
    createServer() {
        this.server = http.createServer(this.app);
    }
    listen(port, host = "localhost", callback = ()=>{}) {
         this.server.listen(port, host, callback);
    }
}
module.exports = Server;
```

6.2 updateCompiler.js

webpack-dev-server\lib\utils\updateCompiler.js

```
const path = require("path");
let updateCompiler = (compiler) => {
    const config = compiler.options;

    config.entry.main.import.unshift(require.resolve("../../client/index.js"),);

    config.entry.main.import.unshift(require.resolve("../../../webpack/hot/dev-server.js"));
    console.log(config.entry);
    compiler.hooks.entryOption.call(config.context, config.entry);
}
module.exports = updateCompiler;
```

6.3 client\index.js

webpack-dev-server\lib\client\index.js

```
console.log('webpack-dev-server\client\index.js');
```

6.4 dev-server.js

webpack-dev-server\lib\client\hot\dev-server.js

```
console.log('webpack-dev-server\lib\client\hot\dev-server.js');
```

7.添加webpack的done事件回调

7.1 Server.js

webpack-dev-server\lib\server\Server.js

```
const express = require("express");
const updateCompiler = require('./utils/updateCompiler');
const http = require("http");
class Server {
    constructor(compiler,devServerOptions) {
        this.compiler = compiler;
        this.devServerOptions=devServerOptions;
        updateCompiler(compiler);
+       this.sockets = [];
+       this.setupHooks();
        this.setupApp();
        this.createServer();
    }
+    setupHooks() {
+        this.compiler.hooks.done.tap('webpack-dev-server', (stats) => {
+            console.log("stats.hash", stats.hash);
+            this.sockets.forEach((socket) => {
+                socket.emit("hash", stats.hash);
+                socket.emit("ok");
+            });
+            this._stats = stats;
+        });
+    }
    setupApp() {
        this.app = new express();
    }
    createServer() {
        this.server = http.createServer(this.app);
    }
    listen(port, host = "localhost", callback = ()=>{}) {
         this.server.listen(port, host, callback);
    }
}
module.exports = Server;
```

8.webpack-dev-middleware中间件

- webpack-dev-middleware 实现webpack编译和文件相关操作

8.1 Server.js

webpack-dev-server\lib\Server.js

```
const express = require("express");
const updateCompiler = require('./utils/updateCompiler');
+const webpackDevMiddleware = require('../../webpack-dev-middleware');
const http = require("http");
class Server {
    constructor(compiler,devServerOptions) {
        this.compiler = compiler;
        this.devServerOptions = devServerOptions;
        updateCompiler(compiler);
        this.sockets = [];
        this.setupHooks();
        this.setupApp();
+       this.setupDevMiddleware();
        this.createServer();
    }
+   setupDevMiddleware() {
+        if(this.devServerOptions.contentBase)
+            this.app.use(express.static(this.devServerOptions.contentBase));
+        this.middleware = webpackDevMiddleware(this.compiler);
+        this.app.use(this.middleware);
+   }
    setupHooks() {
        this.compiler.hooks.done.tap('webpack-dev-server', (stats) => {
            console.log("stats.hash", stats.hash);
            this.sockets.forEach((socket) => {
                socket.emit("hash", stats.hash);
                socket.emit("ok");
            });
            this._stats = stats;
        });
    }
    setupApp() {
        this.app = new express();
    }
    createServer() {
        this.server = http.createServer(this.app);
    }
    listen(port, host = "localhost", callback = ()=>{}) {
         this.server.listen(port, host, callback);
    }
}
module.exports = Server;
```

8.2 webpack-dev-middleware\index.js

webpack-dev-middleware\index.js

```
const middleware = require("./middleware");
const MemoryFileSystem = require("memory-fs");
let memoryFileSystem = new MemoryFileSystem();
function webpackDevMiddleware(compiler) {
    compiler.watch({}, () => {
        console.log("start watching!");
    });
    let fs = compiler.outputFileSystem = memoryFileSystem;
    return middleware({
        fs,
        outputPath:compiler.options.output.path
    });
}

module.exports = webpackDevMiddleware;
```

8.3 middleware.js

webpack-dev-middleware\middleware.js

```
const mime = require('mime');
const path = require("path");
module.exports = function wrapper(context) {
    return function middleware(req, res, next) {
        let url = req.url;
        if (url === "/") { url = "/index.html"; }
        let filename = path.join(context.outputPath, url);
        try {
            let stat = context.fs.statSync(filename);
            if (stat.isFile()) {
                let content = context.fs.readFileSync(filename);
                res.setHeader("Content-Type", mime.getType(filename));
                res.send(content);
            } else {
                res.sendStatus(404);
            }
        } catch (error) {
            res.sendStatus(404);
        }
    };
};
```

9.创建消息服务器

9.1 Server.js

webpack-dev-server\lib\server\Server.js

```
const express = require("express");
const updateCompiler = require('./utils/updateCompiler');
const webpackDevMiddleware = require('../../webpack-dev-middleware');
const http = require("http");
+const WebsocketServer = require("socket.io");
class Server {
    constructor(compiler) {
        this.compiler = compiler;
        updateCompiler(compiler);
        this.sockets = [];
        this.setupHooks();
        this.setupApp();
        this.setupDevMiddleware();
        this.createServer();
+       this.createSocketServer();
    }
    setupDevMiddleware() {
        this.middleware = webpackDevMiddleware(this.compiler);
        this.app.use(this.middleware);
    }
    setupHooks() {
        this.compiler.hooks.done.tap('webpack-dev-server', (stats) => {
            console.log("stats.hash", stats.hash);
            this.sockets.forEach((socket) => {
                socket.emit("hash", stats.hash);
                socket.emit("ok");
            });
            this._stats = stats;
        });
    }
    setupApp() {
        this.app = new express();
    }
    createServer() {
        this.server = http.createServer(this.app);
    }
+    createSocketServer() {
+        const io = WebsocketServer(this.server);
+        io.on("connection", (socket) => {
+            console.log("client connected");
+            this.sockets.push(socket);
+            socket.on("disconnect", () => {
+                let index = this.sockets.indexOf(socket);
+                this.sockets = this.sockets.splice(index, 1);
+            });
+            if(this._stats){
+              socket.emit('hash', this._stats.hash);
+              socket.emit('ok');
+            }
+        });
+    }
    listen(port, host = "localhost", callback = ()=>{}) {
        this.server.listen(port, host, callback);
    }
}
module.exports = Server;
```

1.客户端连接消息服务器

1.1 index.html

src\index.html

```
    hmr

+
```

1.2 webpack\hot\emitter.js

webpack\hot\emitter.js

```
class EventEmitter {
  constructor() {
    this.events = {};
  }
  on(eventName, fn) {
    this.events[eventName] = fn;
  }
  emit(eventName, ...args) {
    this.events[eventName](...args);
  }
}
module.exports = new EventEmitter();
```

1.3 webpack-dev-server\client\index.js

webpack-dev-server\client\index.js

```
var hotEmitter = require("../../webpack/hot/emitter");
var socket = io();
var currentHash = "";
var initial = true;
socket.on("hash", (hash) => {
  currentHash = hash;
});
socket.on("ok", () => {
  console.log("ok");
  if(initial){
    return initial=false;
  }
  reloadApp();
});
function reloadApp() {
  hotEmitter.emit("webpackHotUpdate", currentHash);
}
```

1.4 webpack\hot\dev-server.js

webpack\hot\dev-server.js

```
var hotEmitter = require('../../webpack/hot/emitter');
hotEmitter.on("webpackHotUpdate", (currentHash) => {
   console.log('dev-server',currentHash);
})
```

2.打包后文件分析

2.1 static\hmr.html

static\hmr.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Documenttitle>
head>
<body>
    <input/>
    <div id="root">div>
    <script src="/socket.io/socket.io.js">script>
    [illegible]
[illegible]>
```

2.2 static\hmr.js

static\hmr.js

```
(() => {
  var modules = {
    "./src/title.js": (module) => {
      module.exports = "title3";
    },
    "./webpack/hot/emitter.js": (module) => {
      class EventEmitter {
        constructor() {
          this.events = {};
        }
        on(eventName, fn) {
          this.events[eventName] = fn;
        }
        emit(eventName, ...args) {
          this.events[eventName](...args);
        }
      }
      module.exports = new EventEmitter();
    },
  };
  var cache = {};
  function require(moduleId) {
    if (cache[moduleId]) {
      return cache[moduleId].exports;
    }
    var module = (cache[moduleId] = {
      exports: {},
    });
    modules[moduleId](module, module.exports, require);
    return module.exports;
  }
  (() => {
    var hotEmitter = require("./webpack/hot/emitter.js");
    var socket = io();
    var currentHash = "";
    var initial = true;
    socket.on("hash", (hash) => {
      currentHash = hash;
    });
    socket.on("ok", () => {
      console.log("ok");
      if (initial) {
        return (initial = false);
      }
      reloadApp();
    });
    function reloadApp() {
      hotEmitter.emit("webpackHotUpdate", currentHash);
    }
  })();
  (() => {
    var hotEmitter = require("./webpack/hot/emitter.js");
    hotEmitter.on("webpackHotUpdate", (currentHash) => {
      console.log("dev-server", currentHash);
    });
  })();
  (() => {
    let render = () => {
      let title = require("./src/title.js");
      root.innerText = title;
    };
    render();
  })();
})();
```

2.创建模块父子关系

2.1 hmr.js

```
(() => {
  var modules = {
+    "./src/index.js": (module,exports,require) => {
+      let render = () => {
+        let title = require("./src/title.js");
+        root.innerText = title;
+      };
+      render();
+    },
    "./src/title.js": (module) => {
      module.exports = "title3";
    },
    "./webpack/hot/emitter.js": (module) => {
      class EventEmitter {
        constructor() {
          this.events = {};
        }
        on(eventName, fn) {
          this.events[eventName] = fn;
        }
        emit(eventName, ...args) {
          this.events[eventName](...args);
        }
      }
      module.exports = new EventEmitter();
    },
  };
  var cache = {};
+  function hotCreateModule() {
+    var hot = {
+      _acceptedDependencies: {},
+      accept: function (deps, callback) {
+        for (var i = 0; i < deps.length; i++)
+          hot._acceptedDependencies[deps[i]] = callback;
+      },
+      check: hotCheck,
+    };
+    return hot;
+  }
+ function hotCreateRequire(parentModuleId) {
+   var parentModule = cache[parentModuleId];
+   if (!parentModule) return require;
+   var fn = function (childModuleId) {
+     parentModule.children.push(childModuleId);
+     require(childModuleId);
+     let childModule = cache[childModuleId];
+     childModule.parents.push(parentModule);
+     return childModule.exports;
+   };
+   return fn;
+ }
  function require(moduleId) {
    if (cache[moduleId]) {
      return cache[moduleId].exports;
    }
    var module = (cache[moduleId] = {
      exports: {},
+      hot: hotCreateModule(moduleId),
+      parents: [],
+      children: [],
    });
+    modules[moduleId](module, module.exports, hotCreateRequire(moduleId));
    return module.exports;
  }
  (() => {
    var hotEmitter = require("./webpack/hot/emitter.js");
    var socket = io();
    var currentHash = "";
    var initial = true;
    socket.on("hash", (hash) => {
      currentHash = hash;
    });
    socket.on("ok", () => {
      console.log("ok");
      reloadApp();
    });
    function reloadApp() {
      hotEmitter.emit("webpackHotUpdate", currentHash);
    }
  })();
  (() => {
    var hotEmitter = require("./webpack/hot/emitter.js");
    hotEmitter.on("webpackHotUpdate", (currentHash) => {
      console.log("dev-server", currentHash);
    });
  })();
+  return hotCreateRequire("./src/index.js")("./src/index.js");
})();
```

3.热更新

3.1 webpack.config.js

webpack.config.js

```
module.exports = {
    output: {
        filename: "[name].js",
        path: path.resolve(__dirname, "dist"),
+        hotUpdateGlobal:'webpackHotUpdate'
    }
}
```

3.2 hmr.js

static\hmr.js

```
(() => {
  var cache = {};
+ var currentHash;
+ var lastHash;
+ let hotCheck = () => {
+   hotDownloadManifest()
+     .then((update) => {
+       update.c.forEach((chunkID) => {
+         hotDownloadUpdateChunk(chunkID);
+       });
+       lastHash = currentHash;
+     })
+     .catch((err) => {
+       window.location.reload();
+     });
+ };
+ let hotDownloadManifest = () => {
+   return new Promise((resolve, reject) => {
+     let xhr = new XMLHttpRequest();
+     let hotUpdatePath = `main.${lastHash}.hot-update.json`;
+     xhr.open("get", hotUpdatePath);
+     xhr.onload = () => {
+       let hotUpdate = JSON.parse(xhr.responseText);
+       resolve(hotUpdate);
+     };
+     xhr.onerror = (error) => {
+       reject(error);
+     };
+     xhr.send();
+   });
+ };
+ let hotDownloadUpdateChunk = (chunkID) => {
+   let script = document.createElement("script");
+   script.src = `${chunkID}.${lastHash}.hot-update.js`;
+   document.head.appendChild(script);
+ };
+ self['webpackHotUpdate'] = (chunkId, moreModules) => {
+   hotAddUpdateChunk(chunkId, moreModules);
+ };
+ let hotUpdate = {};
+ function hotAddUpdateChunk(chunkId, moreModules) {
+   for (var moduleId in moreModules) {
+     hotUpdate[moduleId] = modules[moduleId] = moreModules[moduleId];
+   }
+   hotApply();
+ }
+ function hotApply() {
+   for (let moduleId in hotUpdate) {
+     let oldModule = cache[moduleId];
+     delete cache[moduleId];
+     oldModule.parents &&
+       oldModule.parents.forEach((parentModule) => {
+         parentModule.hot._acceptedDependencies[moduleId] &&
+           parentModule.hot._acceptedDependencies[moduleId]();
+       });
+   }
+ }
  var modules = {
    "./src/index.js": (module, exports, require) => {
      let render = () => {
        let title = require("./src/title.js");
        root.innerText = title;
      };
      render();
      if (module.hot) {
        module.hot.accept(["./src/title.js"], render);
      }
    },
    "./src/title.js": (module) => {
      module.exports = "title3";
    },
    "./webpack/hot/emitter.js": (module) => {
      class EventEmitter {
        constructor() {
          this.events = {};
        }
        on(eventName, fn) {
          this.events[eventName] = fn;
        }
        emit(eventName, ...args) {
          this.events[eventName](...args);
        }
      }
      module.exports = new EventEmitter();
    },
  };

  function hotCreateModule() {
    var hot = {
      _acceptedDependencies: {},
      accept: function (deps, callback) {
        for (var i = 0; i < deps.length; i++)
          hot._acceptedDependencies[deps[i]] = callback;
      },
      check: hotCheck,
    };
    return hot;
  }
  function hotCreateRequire(parentModuleId) {
    var parentModule = cache[parentModuleId];
```

```
    if (!parentModule) return require;
    var fn = function (childModuleId) {
      parentModule.children.push(childModuleId);
      require(childModuleId);
      let childModule = cache[childModuleId];
      childModule.parents.push(parentModule);
      return childModule.exports;
    };
    return fn;
  }
  function require(moduleId) {
    if (cache[moduleId]) {
      return cache[moduleId].exports;
    }
    var module = (cache[moduleId] = {
      exports: {},
      hot: hotCreateModule(moduleId),
      parents: [],
      children: [],
    });
    modules[moduleId](module, module.exports, hotCreateRequire(moduleId));
    return module.exports;
  }
  (() => {
    var hotEmitter = require("./webpack/hot/emitter.js");
    var socket = io();
    var initial = true;
    socket.on("hash", (hash) => {
      currentHash = hash;
    });
    socket.on("ok", () => {
      console.log("ok");
      reloadApp();
    });
    function reloadApp() {
      hotEmitter.emit("webpackHotUpdate", currentHash);
    }
  })();
  (() => {
    var hotEmitter = require("./webpack/hot/emitter.js");
    hotEmitter.on("webpackHotUpdate", (currentHash) => {
+     if (!lastHash) {
+       lastHash = currentHash;
+       console.log("lastHash=", lastHash, "currentHash=", currentHash);
+       return;
+     }
+     console.log("lastHash=", lastHash, "currentHash=", currentHash);
+     console.log("webpackHotUpdate hotCheck");
+     hotCheck();
    });
  })();
  return hotCreateRequire("./src/index.js")("./src/index.js");
})();
```

4.注释版

4.1 hmr.js

```
(() => {
  var cache = {};
  var currentHash;
  var lastHash;
  let hotUpdate = {};

  let hotCheck = () => {

    hotDownloadManifest()
      .then((update) => {
        update.c.forEach((chunkID) => {

          hotDownloadUpdateChunk(chunkID);
        });
        lastHash = currentHash;
      })
      .catch((err) => {
        window.location.reload();
      });
  };
  let hotDownloadManifest = () => {
    return new Promise((resolve, reject) => {
      let xhr = new XMLHttpRequest();
      let hotUpdatePath = `main.${lastHash}.hot-update.json`;
      xhr.open("get", hotUpdatePath);
      xhr.onload = () => {
        let hotUpdate = JSON.parse(xhr.responseText);
        resolve(hotUpdate);
      };
      xhr.onerror = (error) => {
        reject(error);
      };
      xhr.send();
    });
  };
  let hotDownloadUpdateChunk = (chunkID) => {
    let script = document.createElement("script");
    script.src = `${chunkID}.${lastHash}.hot-update.js`;
    document.head.appendChild(script);
  };

  self['webpackHotUpdate'] = (chunkId, moreModules) => {

      hotAddUpdateChunk(chunkId, moreModules);
```

```
};

function hotAddUpdateChunk(chunkId, moreModules) {
  for (var moduleId in moreModules) {
    hotUpdate[moduleId] = modules[moduleId] = moreModules[moduleId];
  }

  hotApply();
}
function hotApply() {
  for (let moduleId in hotUpdate) {
    let oldModule = cache[moduleId];
    delete cache[moduleId];
    oldModule.parents &&
      oldModule.parents.forEach((parentModule) => {
        parentModule.hot._acceptedDependencies[moduleId] &&
          parentModule.hot._acceptedDependencies[moduleId]();
      });
  }
}
var modules = {
  "./src/index.js": (module, exports, require) => {
    let render = () => {
      let title = require("./src/title.js");
      root.innerText = title;
    };
    render();
    if (module.hot) {
      module.hot.accept(["./src/title.js"], render);
    }
  },
  "./src/title.js": (module) => {
    module.exports = "title3";
  },
  "./webpack/hot/emitter.js": (module) => {
    class EventEmitter {
      constructor() {
        this.events = {};
      }
      on(eventName, fn) {
        this.events[eventName] = fn;
      }
      emit(eventName, ...args) {
        this.events[eventName](...args);
      }
    }
    module.exports = new EventEmitter();
  },
};

function hotCreateModule() {
  var hot = {
    _acceptedDependencies: {},
    accept: function (deps, callback) {
      for (var i = 0; i < deps.length; i++)
        hot._acceptedDependencies[deps[i]] = callback;
    },
    check: hotCheck,
  };
  return hot;
}
function hotCreateRequire(parentModuleId) {
  var parentModule = cache[parentModuleId];
  if (!parentModule) return require;
  var fn = function (childModuleId) {
    parentModule.children.push(childModuleId);
    require(childModuleId);
    let childModule = cache[childModuleId];
    childModule.parents.push(parentModule);
    return childModule.exports;
  };
  return fn;
}
function require(moduleId) {
  if (cache[moduleId]) {
    return cache[moduleId].exports;
  }
  var module = (cache[moduleId] = {
    exports: {},
    hot: hotCreateModule(moduleId),
    parents: [],
    children: [],
  });
  modules[moduleId](module, module.exports, hotCreateRequire(moduleId));
  return module.exports;
}
(() => {
  var hotEmitter = require("./webpack/hot/emitter.js");
  var socket = io();

  socket.on("hash", (hash) => {
    currentHash = hash;
  });
  socket.on("ok", () => {
    console.log("ok");

    reloadApp();
  });
  function reloadApp() {

    hotEmitter.emit("webpackHotUpdate", currentHash);
  }
})();
(() => {
```

```
    var hotEmitter = require("./webpack/hot/emitter.js");

    hotEmitter.on("webpackHotUpdate", (currentHash) => {
      if (!lastHash) {
        lastHash = currentHash;
        console.log("lastHash=", lastHash, "currentHash=", currentHash);
        return;
      }
      console.log("lastHash=", lastHash, "currentHash=", currentHash);
      console.log("webpackHotUpdate hotCheck");

    });
  })();
  return hotCreateRequire("./src/index.js")("./src/index.js");
})();
```

4.2 startDevServer.js

startDevServer.js

```
const webpack = require("webpack")
const Server = require('./webpack-dev-server/lib/Server');
const config = require("./webpack.config")
function startDevServer(compiler,options) {
    const devServerOptions = options.devServer||{};

    const server = new Server(compiler, devServerOptions);
    const {host='localhost',port=8080}=devServerOptions;
    server.listen(port, host, (err) => {
        console.log(`Project is running at http://${host}:${port}`);
    });
}

const compiler = webpack(config);
startDevServer(compiler,config);
```

4.3 webpack-dev-server\lib\Server.js

webpack-dev-server\lib\Server.js

```
const express = require("express");
const updateCompiler = require('./utils/updateCompiler');
const webpackDevMiddleware = require('../../webpack-dev-middleware');
const http = require("http");
const WebsocketServer = require("socket.io");
class Server {
    constructor(compiler,devServerOptions) {
        this.compiler = compiler;
        this.devServerOptions = devServerOptions;
        updateCompiler(compiler);
        this.sockets = [];

        this.setupHooks();

        this.setupApp();

        this.setupDevMiddleware();

        this.createServer();

        this.createSocketServer();
    }
    setupDevMiddleware() {
        if(this.devServerOptions.contentBase)
            this.app.use(express.static(this.devServerOptions.contentBase));
        this.middleware = webpackDevMiddleware(this.compiler);
        this.app.use(this.middleware);
    }
    setupHooks() {
        this.compiler.hooks.done.tap('webpack-dev-server', (stats) => {
            console.log("stats.hash", stats.hash);
            this.sockets.forEach((socket) => {
                socket.emit("hash", stats.hash);
                socket.emit("ok");
            });
            this._stats = stats;
        });
    }
    setupApp() {
        this.app = new express();
    }
    createServer() {
        this.server = http.createServer(this.app);
    }
    createSocketServer() {
        const io = WebsocketServer(this.server);
        io.on("connection", (socket) => {
            console.log("client connected");
            this.sockets.push(socket);
            socket.on("disconnect", () => {
                let index = this.sockets.indexOf(socket);
                this.sockets = this.sockets.splice(index, 1);
            });
            if(this._stats){
                socket.emit('hash', this._stats.hash);
                socket.emit("ok");
            }
        });
    }
    listen(port, host = "localhost", callback = ()=>{}) {
         this.server.listen(port, host, callback);
    }
}
module.exports = Server;
```

4.4 webpack-dev-middleware\index.js

webpack-dev-middleware\index.js

```
const middleware = require("./middleware");
const MemoryFileSystem = require("memory-fs");
let memoryFileSystem = new MemoryFileSystem();
function webpackDevMiddleware(compiler) {
    compiler.watch({}, () => {
        console.log("start watching!");
    });

    let fs = compiler.outputFileSystem = memoryFileSystem;
    return middleware({
        fs,
        outputPath:compiler.options.output.path
    });
}

module.exports = webpackDevMiddleware;
```