极客时间

目 录

CONTENTS

极客时间

扫码试看/订阅
《玩转webpack》

当前构建时的问题

每次构建的时候不会清理目录，造成构建的输出目录 output 文件越来越多

通过 npm scripts 清理构建目录

rm –rf ./dist && webpack

rimraf ./dist && webpack

自动清理构建目录

避免构建前每次都需要手动删除 dist

使用 clean-webpack-plugin

- 默认会删除 output 指定的输出目录

```
module.exports = {
  entry: {
    app: './src/app.js',
    search: './src/search.js'
  },
  output: {
    filename: '[name][chunkhash:8].js',
    path: __dirname + '/dist'
  },
  plugins: [
+   new CleanWebpackPlugin()
};
```

CSS3 的属性为什么需要前缀?

Trident(-ms)

Geko(-moz)

Webkit(-webkit)

Presto(-o)

举个例子

```
.box {
    -moz-border-radius: 10px;
    -webkit-border-radius: 10px;
    -o-border-radius: 10px;
    border-radius: 10px;
}
```

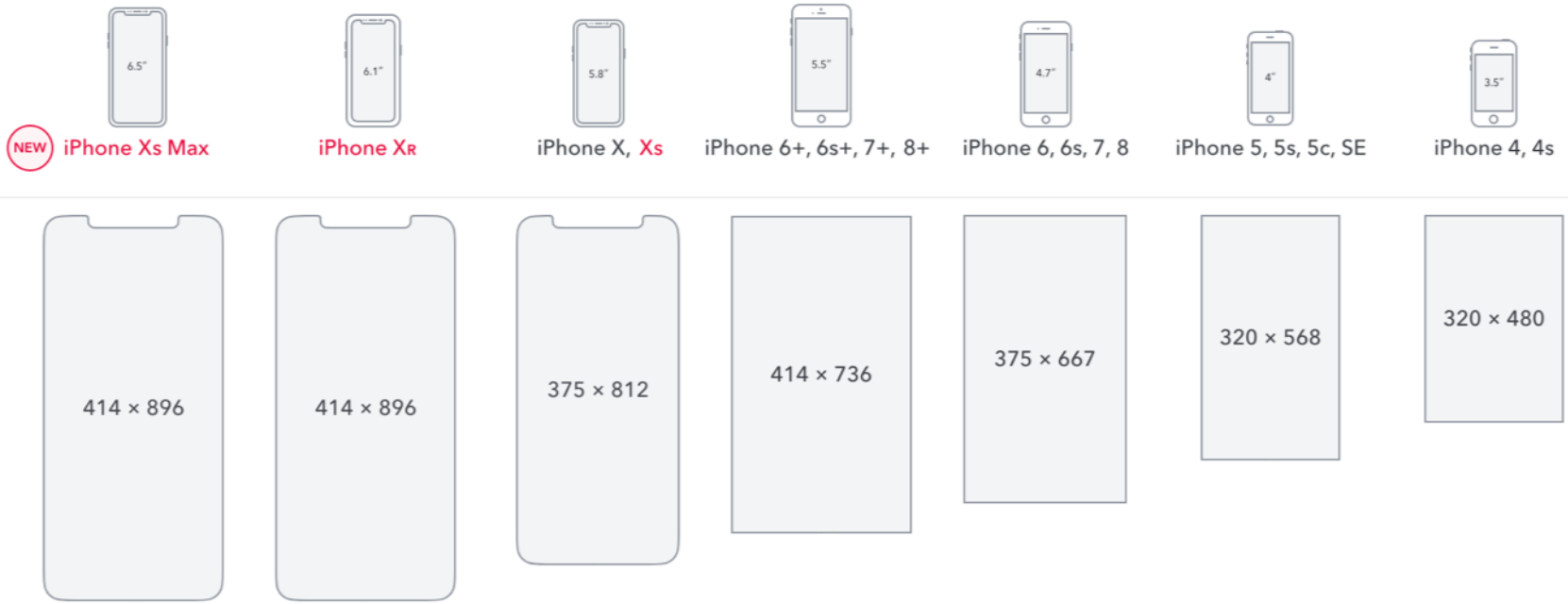
PostCSS 插件 autoprefixer 自动补齐 CSS3 前缀

使用 autoprefixer 插件

根据 Can I Use 规则（ https://caniuse.com/ ）

```
module.exports = {
   module: {
    rules: [
     {
      test: /\.less$/,
      use: [
       'style-loader',
       'css-loader',
       'less-loader',
+       {
+         loader: 'postcss-loader',
+         options: {
+           plugins: () => [
+             require('autoprefixer')({
+               browsers: ["last 2 version", "> 1%",  "iOS 7"]
+             })
+           ]
+         }
+       }
      ]
     }
    ]
   }
};
```

浏览器的分辨率


6.5"
NEW iPhone Xs Max
414 × 896
6.1"
iPhone XR
414 × 896
5.8"
iPhone X, Xs
375 × 812
5.5"
iPhone 6+, 6s+, 7+, 8+
414 × 736
4.7"
iPhone 6, 6s, 7, 8
375 × 667
4"
iPhone 5, 5s, 5c, SE
320 × 568
3.5"
iPhone 4, 4s
320 × 480

CSS 媒体查询实现响应式布局

缺陷：需要写多套适配样式代码

```
@media screen and (max-width: 980px) {
  .header {
    width: 900px;
  }
}
@media screen and (max-width: 480px) {
  .header {
    width: 400px;
  }
}
@media screen and (max-width: 350px) {
  .header {
    width: 300px;
  }
}
```

rem 是什么?

W3C 对 rem 的定义： font-size of the root element

rem 和 px 的对比：

- rem 是相对单位
- px 是绝对单位

移动端 CSS px 自动转换成 rem

使用 px2rem-loader

页面渲染时计算根元素的 font-size 值

- 可以使用手淘的lib-flexible库
- https://github.com/amfe/lib-flexible

```
module.exports = {
  module: {
    rules: [
      {
        test: /\.less$/,
        use: [
          'style-loader',
          'css-loader',
          'less-loader',
+         {
+           loader: "px2rem-loader",
+           options: {
+             remUnit: 75,
+             remPrecision: 8
+           }
+         }
        ]
      }
    ]
  }
};
```

资源内联的意义

代码层面：

- 页面框架的初始化脚本
- 上报相关打点
- css 内联避免页面闪动

请求层面：减少 HTTP 网络请求数

- 小图片或者字体内联 (url-loader)

HTML 和 JS 内联

raw-loader 内联 html

```
<script>${require(' raw-loader!babel-loader!. /meta.html')}</script>
```

raw-loader 内联 JS

```
<script>${require('raw-loader!babel-loader!../node_modules/lib-flexible')}</script>
```

CSS 内联

方案一：借助 style-loader

方案二：html-inline-css-webpack-plugin

```
module.exports = {
  module: {
    rules: [
      {
        test: /\.scss$/,
        use: [
          {
            loader: 'style-loader',
            options: {
              insertAt: 'top', // 样式插入到 <head>
              singleton: true, //将所有的style标签合并成一个
            }
          },
          "css-loader",
          "sass-loader"
        ],
      },
    ]
  },
};
```

多页面应用(MPA)概念

每一次页面跳转的时候，后台服务器都会给返回一个新的 html 文档，这种类型的网站也就是多页网站，也叫做多页应用。

多页面打包基本思路

每个页面对应一个 entry，一个 html-webpack-plugin

缺点：每次新增或删除页面需要改 webpack 配置

```
module.exports = {
  entry: {
    index: './src/index.js',
    search: './src/search.js
  }
};
```

多页面打包通用方案

动态获取 entry 和设置 html-webpack-plugin 数量

利用 glob.sync

- entry: glob.sync(path.join(__dirname, './src/*/index.js')),

```
module.exports = {
  entry: {
    index: './src/index/index.js',
    search: './src/search/index.js
  }
};
```

使用 source map

作用：通过 source map 定位到源代码

- source map科普文：http://www.ruanyifeng.com/blog/2013/01/javascript_source_map.html

开发环境开启，线上环境关闭

- 线上排查问题的时候可以将 sourcemap 上传到错误监控系统

source map 关键字

eval: 使用eval包裹模块代码

source map: 产生.map文件

cheap: 不包含列信息

inline: 将.map作为DataURI嵌入，不单独生成.map文件

module:包含loader的sourcemap

source map 类型

devtool	首次构建	二次构建	是否适合生产环境	可以定位的代码
(none)	+++	+++	yes	最终输出的代码
eval	+++	+++	no	webpack生成的代码(一个个的模块)
cheap-eval-source-map	+	++	no	经过loader转换后的代码(只能看到行)
cheap-module-eval-source-map	o	++	no	源代码(只能看到行)
eval-source-map	--	+	no	源代码
cheap-source-map	+	o	yes	经过loader转换后的代码(只能看到行)
cheap-module-source-map	o	-	yes	源代码(只能看到行)
inline-cheap-source-map	+	o	no	经过loader转换后的代码(只能看到行)
inline-cheap-module-source-map	o	-	no	源代码(只能看到行)
source-map	--	--	yes	源代码
inline-source-map	--	--	no	源代码
hidden-source-map	--	--	yes	源代码

基础库分离

·思路：将 react、react-dom 基础包通过 cdn 引入，不打入 bundle 中

·方法：使用 html-webpack-externals-plugin

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

plugins: [
    new HtmlWebpackExternalsPlugin({
        externals: [
            {
                module: 'react',
                entry: '//11.url.cn/now/lib/15.1.0/react-with-addons.min.js?_bid=3123',
                global: 'React'
            }, {
                module: 'react-dom',
                entry: '//11.url.cn/now/lib/15.1.0/react-dom.min.js?_bid=3123',
                global: 'ReactDOM'
            }
        ]
    });
];
```

```
<!doctype html>
<html lang="zh_CN" style="font-size: 146.5px;">
 ▶<head>…</head>
 ▼<body style="font-size: 18px;">
   ▶<script>…</script>
   ▶<div id="container">…</div>
    <script type="text/javascript" src="//11.url.cn/now/lib/16.2.0/react.min.js?_bid=3123"></script>
    <script type="text/javascript" src="//11.url.cn/now/lib/16.2.0/react-dom.min.js?_bid=3123"></script>
    <script type="text/javascript" src="//s.url.cn/qqun/qun/qunpay/qq/withdraw/income_455d05c8.js?_bid=152"
    TWIaa8ZD/rQZptX8Urp5O2Ef3IT48JbtHS07nW2U= sha384-6E2BbRVmJ2ZLQCQyWHyOYRftEVktwLsaWnC+8h1oyip/0F+6Xa3Lo+
    "anonymous"></script>
  </body>
</html>
```

利用 SplitChunksPlugin 进行公共脚本分离

Webpack4 内置的，替代CommonsChunkPlugin插件

chunks 参数说明：

- async 异步引入的库进行分离(默认)
- initial 同步引入的库进行分离
- all 所有引入的库进行分离(推荐)

```
module.exports = {
  optimization: {
    splitChunks: {
      chunks: 'async',
      minSize: 30000,
      maxSize: 0,
      minChunks: 1,
      maxAsyncRequests: 5,
      maxInitialRequests: 3,
      automaticNameDelimiter: '~',
      name: true,
      cacheGroups: {
        vendors: {
          test: /[\\/]node_modules[\\/]/,
          priority: -10
        }
      }
    }
  }
};
```

利用 SplitChunksPlugin 分离基础包

test: 匹配出需要分离的包

```
module.exports = {
  optimization: {
    splitChunks: {
      cacheGroups: {
        commons: {
          test: /(react|react-dom)/,
          name: 'vendors',
          chunks: 'all'
        }
      }
    }
  }
};
```

利用 SplitChunksPlugin 分离页面公共文件

minChunks: 设置最小引用次数为2次

minuSize: 分离的包体积的大小

```
module.exports = {
  optimization: {
    splitChunks: {
      minSize: 0,
      cacheGroups: {
        commons: {
          name: 'commons',
          chunks: 'all',
          minChunks: 2
        }
      }
    }
  }
};
```

tree shaking(摇树优化)

概念：1 个模块可能有多个方法，只要其中的某个方法使用到了，则整个文件都会被打到 bundle 里面去，tree shaking 就是只把用到的方法打入 bundle ，没用到的方法会在 uglify 阶段被擦除掉。

使用：webpack 默认支持，在 .babelrc 里设置 modules: false 即可

- production mode的情况下默认开启

要求：必须是 ES6 的语法，CJS 的方式不支持

DCE (Elimination)

代码不会被执行，不可到达

代码执行的结果不会被用到

代码只会影响死变量（只写不读）

```
if (false) {
    console.log('这段代码永远不会执行'  );
}
```

Tree-shaking 原理

利用 ES6 模块的特点:

- 只能作为模块顶层的语句出现
- import 的模块名只能是字符串常量
- import binding 是 immutable的

代码擦除： uglify 阶段删除无用代码

极客时间

扫码试看/订阅
《玩转webpack》