前端打包的那些事

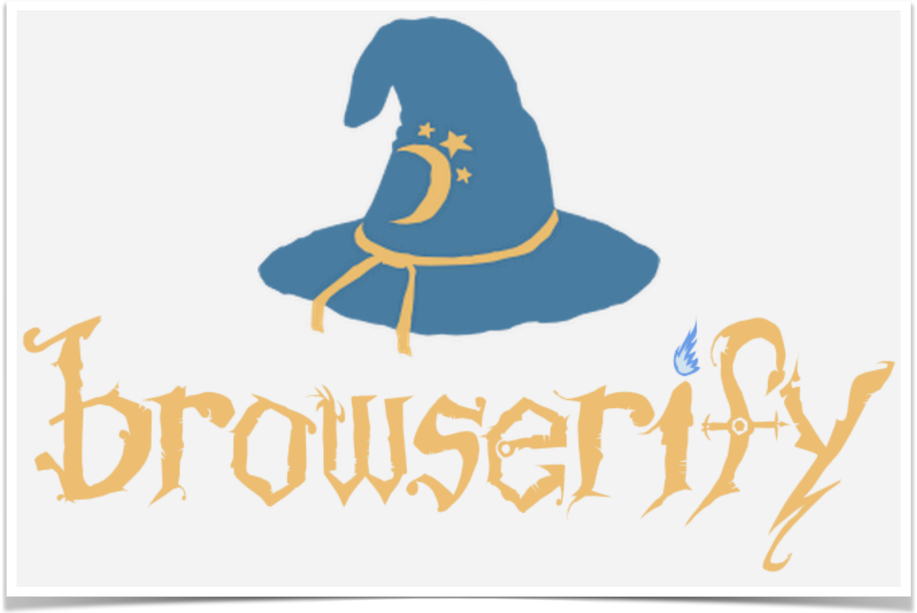
webpack & RollUp & vitejs

Browserify

Browserify 是早期的模块打包工具，让使用 CommonJS 规范的格式组织代码成为可能

```
// add.js
module.exports = function(a, b) {
  return a + b
};

// test.js
var add = require('./add.js');
console.log(add(1, 2));  // 3
```



browserify test.js > bundle.js

Grunt

Grunt 的出现早于 Gulp，Gulp 是后起之秀。他们本质都是通过 JavaScript 语法实现了 shell script 命令的一些功能。

```
// Gruntfile.js
module.exports = function(grunt) {
  grunt.initConfig({
    // js格式检查任务
    jshint: {
      src: 'src/test.js'
    }
    //  代码压缩打包任务
    uglify: {}
  });
  // 导入任务插件
  grunt.loadnpmTasks('grunt-contrib-uglify');
  // 注册自定义任务, 如果有多个任务可以添加到数组中
  grunt.regusterTask('default', ['jshint'])
}
```

Gulp

Gulp 吸取了 Grunt 的优点，拥有更简便的写法，通过流（stream）的概念来简化多任务之间的配置和输出，让任务更加简洁和容易上手。

```
var gulp = require('gulp');
var jshint = require('gulp-jshint');
var uglify = require('gulp-uglify');

// 代码检查任务 gulp 采取了pipe 方法，用流的方法直接往下传递
gulp.task('lint', function() {
  return gulp.src('src/test.js')
    .pipe(jshint())
    .pipe(jshint.reporter('default'));
});

// 压缩代码任务
gulp.task('compress',  function() {
  return gulp.src('src/test.js')
    .pipe(uglify())
    .pipe(gulp.dest('build'));
});

// 将代码检查和压缩组合，新建一个任务
gulp.task('default', ['lint', 'compress']);
```

Webpack

Webpack 可以设置入口文件，然后自动找寻依赖进行编译。webpack插件较多，可以完成各种编译，文件优化等功能

Webpack

入口(entry)

输出(output)

loader

Loader可以将文件从不同的语言（如TypeScript）转换为JavaScript，或者将内联图像转换为data URL。

插件(plugin)

在于解决loader无法实现的其他事，从打包优化和压缩，到重新定义环境变量，功能强大到可以用来处理各种各样的任务

模式(mode)

Webpack

```
{
  entry: {
    driedFishLottery: '/views/driedFishLottery/index.js',
    impactChallenge: 'views/impactChallenge/index.js',
    luckyKick: '/views/luckyKick/index.js',
  },
  mode: 'production',
  devServer: {
    host: '0.0.0.0',
    port: 3000,
    historyApiFallback: true,
    open: true,
    useLocalIp: true,
    openPage: 'driedFishLottery.html'
  },
  resolve: { alias: { '@': '/Users/maomao/code/activities/src' } },
  devtool: 'inline-source-map',
  output: {
    filename: 'js/[name]-bundle.[chunkhash:8].js',
    path: '/Users/maomao/code/activities/dist',
    publicPath: '/gameActivity/'
  },
  module: {
    rules: [
        loaders
    ]
  },
  plugins: [
    ...
  ],
  optimization: {
    splitChunks: { ... },
    minimize: true,
    minimizer: [ [CssMinimizerPlugin], [UglifyJsPlugin] ]
  },
  performance: { maxEntrypointSize: 500000, maxAssetSize: 400000 }
}
```

常用的Loader

style-loader

把 CSS 插入到 DOM 中。

css-loader

会对 @import 和 url() 进行处理，就像 js 解析 import/require() 一样。

html-loader

将 HTML 导出为字符串。当编译器需要时，将压缩 HTML 字符串。

babel-loader

将ES6语法进行转换

常用的Plugin

CopyWebpackPlugin

对文件或者目录进行复制

CleanWebpackPlugin

每次Build前删除dist目录

MiniCssExtractPlugin

将 CSS 提取到单独的文件中，为每个包含 CSS 的 JS 文件创建一个 CSS 文件，并且支持 CSS 和 SourceMaps 的按需加载。

BundleAnalyzerPlugin

使用交互式可缩放图，显示可视化 webpack 输出文件的大小

如何开发Webpack的Plugin

webpack 的插件包括：

- 命名的 JavaScript 函数或 JavaScript 类。
- `apply`在其原型中定义方法。
- 指定要利用的事件挂钩。
- 操作 webpack 内部实例特定数据。
- 在功能完成后调用 webpack 提供的回调。

直接看例子

```
class HelloWorldPlugin {
  apply(compiler) {
    compiler.hooks.done.tap(
      'Hello World Plugin',
      (
        stats
      ) => {
        console.log('Hello World!');
      }
    );
  }
}

module.exports = HelloWorldPlugin;
```

https://gitee.com/-/ide/project/delicious28/upload-plugin/edit/master/-/index.js

RollUp

Rollup 所有资源放同一个地方，一次性加载,利用tree-shake特性来 剔除未使用的代码，减少冗余

RollUp

优点

- Tree Shaking: 自动移除未使用的代码, 输出更小的文件
- Scope Hoisting: 所有模块构建在一个函数内, 执行效率更高
- 可以一次输出多种格式:IIFE, AMD, CJS, UMD, ESM

缺点

- 插件比较少（相对于webpack）

RollUp的配置文件

```
export default {
  // 核心选项
  input,     // 必须
  plugins,

  output: {  // 必须 (如果要输出多个，可以是一个数组)
    // 核心选项
    file,    // 必须
    format,  // 必须
    name,
    globals,
  }
};
```

Vite

可能是启动最快的构建工具

Vite

源代码中的ES Import语法直接提供给浏览器，浏览器通过原生的<script module>支持解析这些语法，并为每次导入发起HTTP请求。
dev服务器拦截请求，并在必要时执行代码转换。

Vite

优点

- 超快速的服务启动（因为启动时啥也不编译）
- 开箱即用 (基于rollUp 开发了less，jsx, ts 的 plugin)

缺点

- 插通过服务器拦截请求根据后缀来进行编译，所以在引用时一定要写清后缀（比如.vue .jsx）
- 因为编译是基于rollUp的，所以插件也还是少

Vite的配置文件

```
export default defineConfig({
  build: {
    rollupOptions: {
      input: {
        main: 'main/index.html',
        main2: 'main2/index.html',
        luckyKick: 'luckyKick/index.html'
      },
    },
    sourcemap:true
  },
  resolve:{
    alias:{
      "@": '/src'
    }
  },
  clearScreen:false,
  plugins: [
    plugin(),
    reactRefresh()
  ]
})
```

点击查看转换成Rollup的配置文件>>

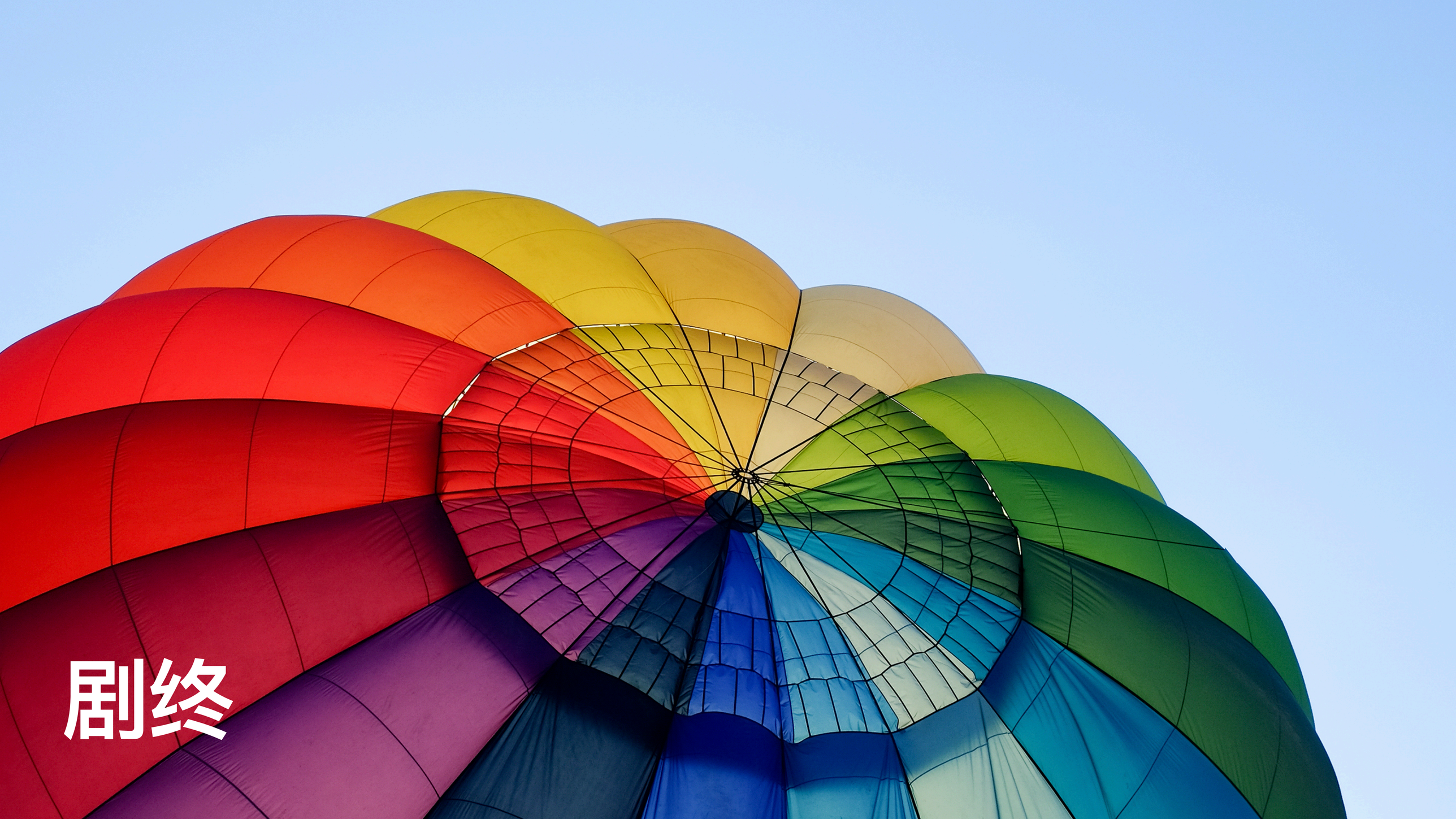
Vite&RollUp plugin 开发

```
export default function uglifyImgPlugin () {
    return {
      name: 'uglifyImg-plugin', // this name will show up in warnings and errors
      configResolved(resolvedConfig){
        //获取全部的rollUp配置信息
        config = resolvedConfig;
      },
      async closeBundle(){
        uglify({
            path:path.resolve(__dirname, './'+config.build.outDir)
        });
      }
    };
  }
```

ps: 由于vite在开发环境有极速启动的功能，所以插件会多一些特殊的钩子

Vite的插件

其实vite的常用插件基本也都有了


剧终