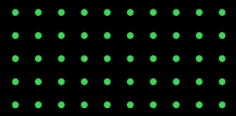
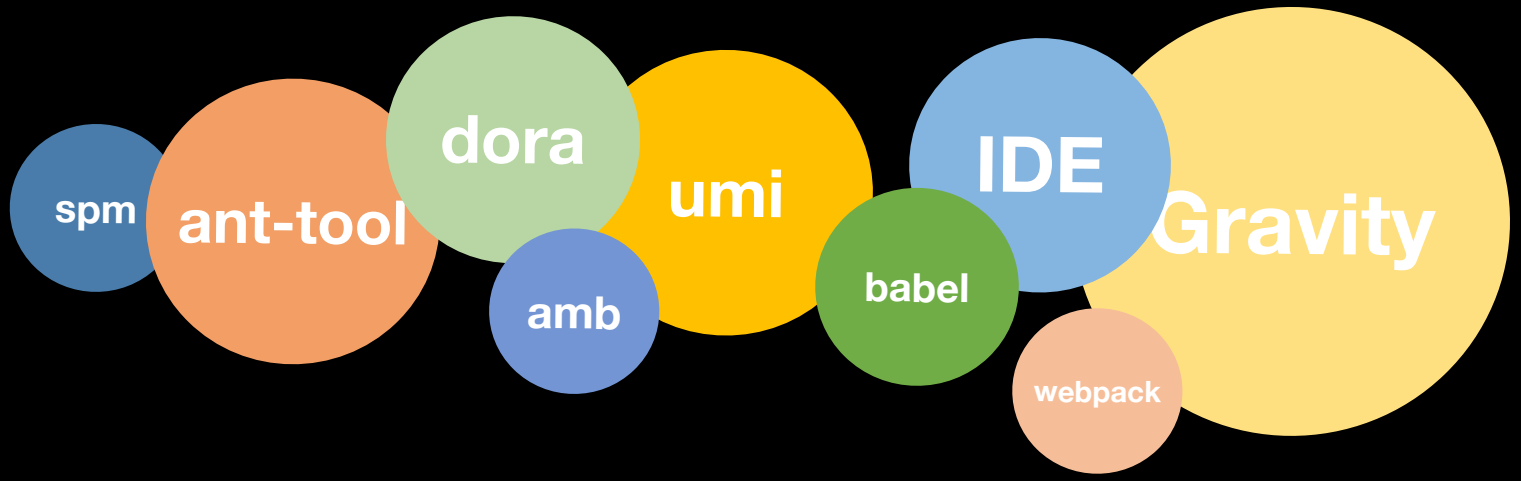

创心
第十四届
D2前端技术论坛
2019 - 12 - 14 杭州和达希尔顿逸林酒店

基于浏览器的实时构建探索之路

RichLab 花呗借呗前端团队 – 姜维（玄寂）



自我介绍



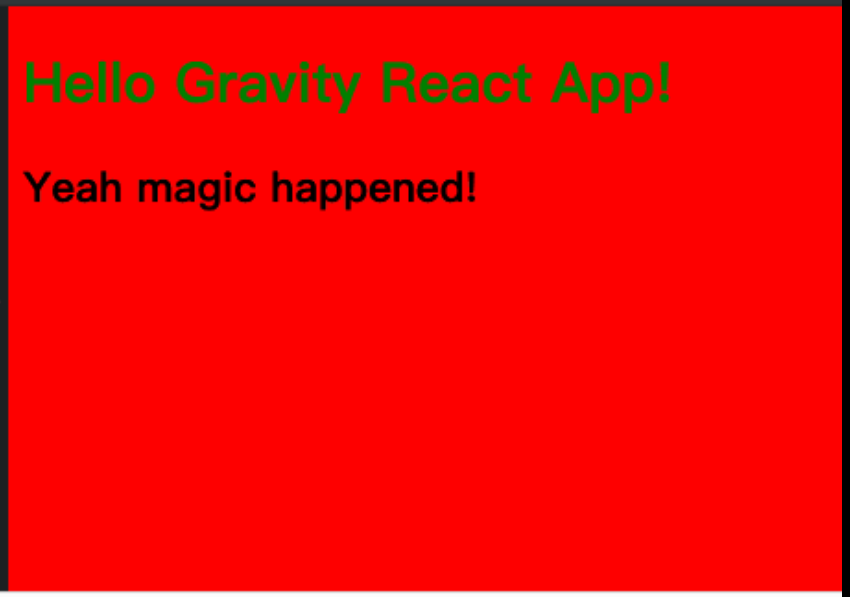
pigcan

猪罐头

基于浏览器实时构建的案例

index.js
c.js
x.css
style.css

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import style from "./style.css";
4 function App() {
5   return (
6     <div>
7       <h1 className={style.green}>Hello Gravity React App!</h1>
8       <h2 className={style.blue}>Yeah magic happened!</h2>
9     </div>
10  );
11 }
12 const rootElement = document.getElementById("root");
13 ReactDOM.render(<App />, rootElement);
```



Elements Console Sources Network Performance Memory Application Security Audits 4

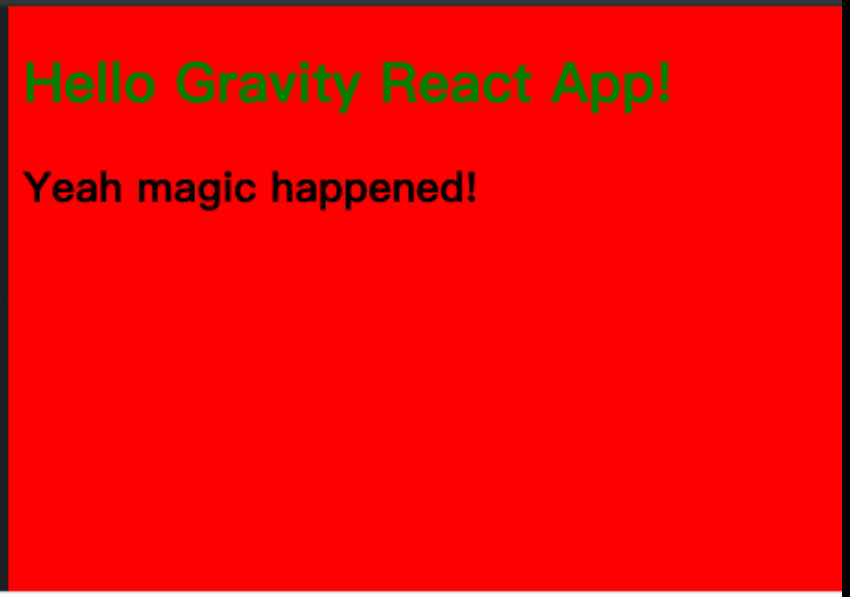
top Filter Default levels

```
receiveMessageFromIndex ▶MessageEvent {isTrusted: true, data: "zero-timeout-message", origin: "http://localhost:3000", lastEventId: "", source: Window, …}  frame.html:13
receiveMessageFromIndex ▶MessageEvent {isTrusted: true, data: "zero-timeout-message", origin: "http://localhost:3000", lastEventId: "", source: Window, …}  frame.html:13
receiveMessageFromIndex ▶MessageEvent {isTrusted: true, data: "zero-timeout-message", origin: "http://localhost:3000", lastEventId: "", source: Window, …}  frame.html:13
```

基于浏览器实时构建的案例

index.js
c.js
x.css
style.css

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import style from "./style.css";
4 function App() {
5   return (
6     <div>
7       <h1 className={style.green}>Hello Gravity React App!</h1>
8       <h2 className={style.blue}>Yeah magic happened!</h2>
9     </div>
10  );
11 }
12 const rootElement = document.getElementById("root");
13 ReactDOM.render(<App />, rootElement);
```



Elements Console Sources Network Performance Memory Application Security Audits 4

top Filter Default levels

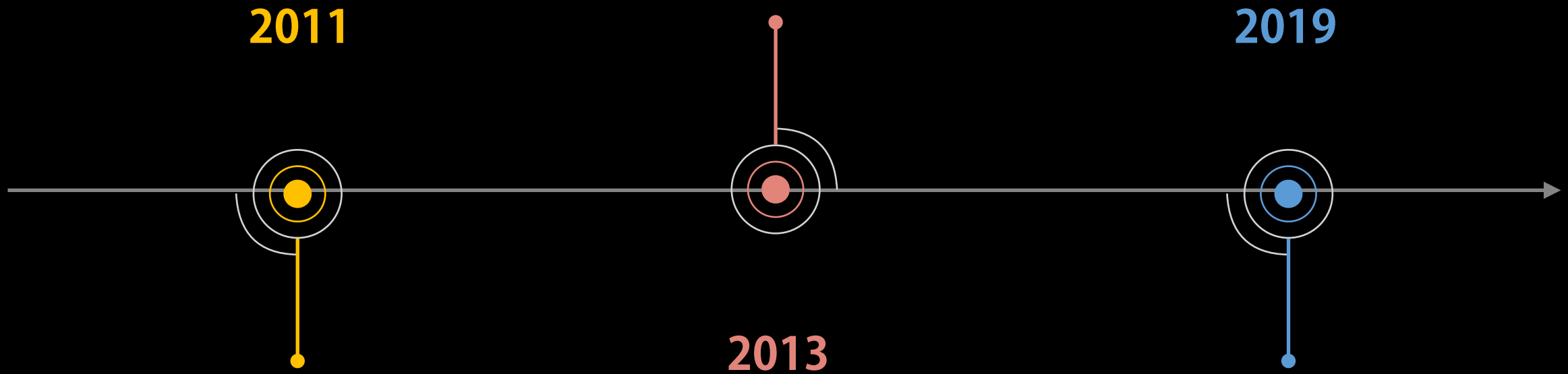
```
receiveMessageFromIndex ▶MessageEvent {isTrusted: true, data: "zero-timeout-message", origin: "http://localhost:3000", lastEventId: "", source: Window, …}  frame.html:13
receiveMessageFromIndex ▶MessageEvent {isTrusted: true, data: "zero-timeout-message", origin: "http://localhost:3000", lastEventId: "", source: Window, …}  frame.html:13
receiveMessageFromIndex ▶MessageEvent {isTrusted: true, data: "zero-timeout-message", origin: "http://localhost:3000", lastEventId: "", source: Window, …}  frame.html:13
```

目录

- 背景
- 机遇与挑战
- 架构大图
- 专题深入
- 未来

背景

背景 – 前端构建史



背景 – 前端构建史

2011

2013

2019

复用性: 粒子化

合并、压缩

背景 – 前端构建史

2011

复用性: 粒子化
合并、压缩

GRUNT

Gulp

2013

模块化: 模块定义 源生态
bundler

Browserify

2019

背景 – 前端构建史

模块化: 模块定义 源生态
bundler

2011

browserify

2013

2019

复用性: 粒子化
合并、压缩

GRUNT

Gulp

Don’t Build That App!
Luke Jackson / April 4, 2019

A Future Without Webpack
Fred K. Schott / March 12, 2019

背景 – 声音的背后 – 学习曲线

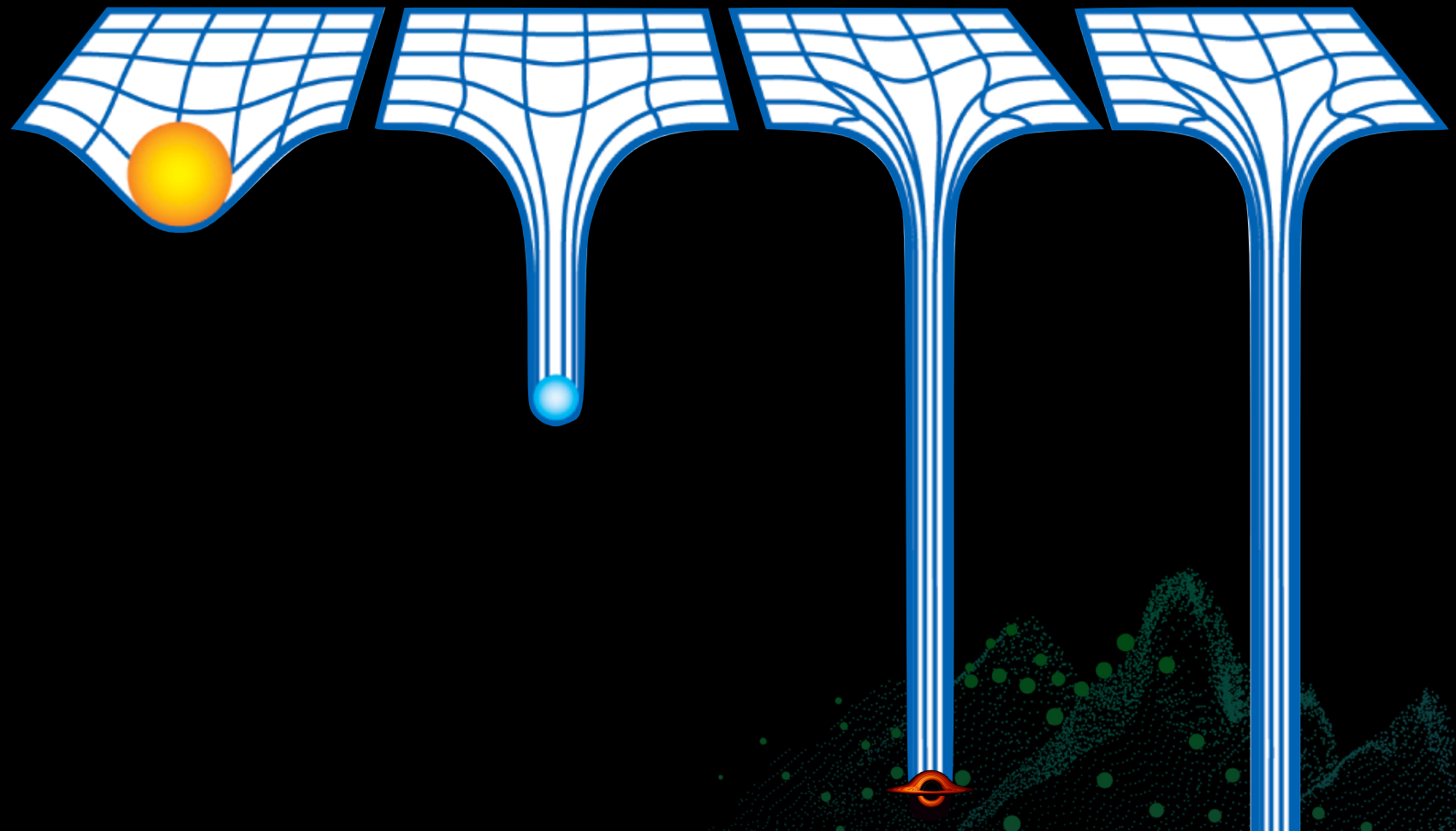
- 前端构建的概念
- 如何在琳琅满目的打包工具中做选择
- 如何安装环境，如何执行构建，如何执行调试
- 如何配置 - webpack、webpack loaders and plugins etc.
- 如何写插件 - babel APIs、webpack APIs etc.
- 如何调试插件
- 如何解决依赖升级 - babel 5 -> 6 -> 7, webpack 1 -> 2 -> 3 -> 4 -> 5

背景 – 声音的背后 – 包管理

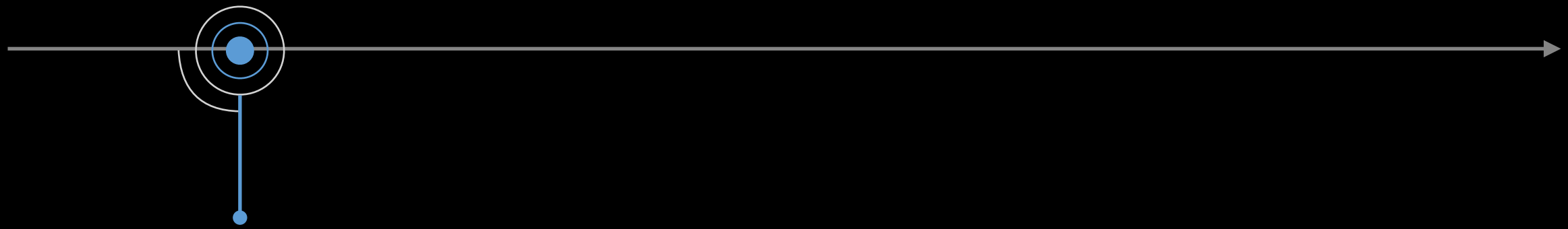

react-scripts
show
package info
graph info
# of nodes
1294
# of links
2728
maintainers
view source code
share to twitter
share to google+
react-scripts@3.0.1
Show 3D

背景 – 声音的背后 – 包管理

太阳　　中子星　　黑洞　　node_modules



2019



云化

可视化编程

背景 – 趋势

2019

云化
可视化编程

?

包管理

Bundler

背景 – 趋势

2019

云化
可视化编程

?

包管理

Bundler

背景 – 趋势

2019

云化
可视化编程

?

包管理

Bundler

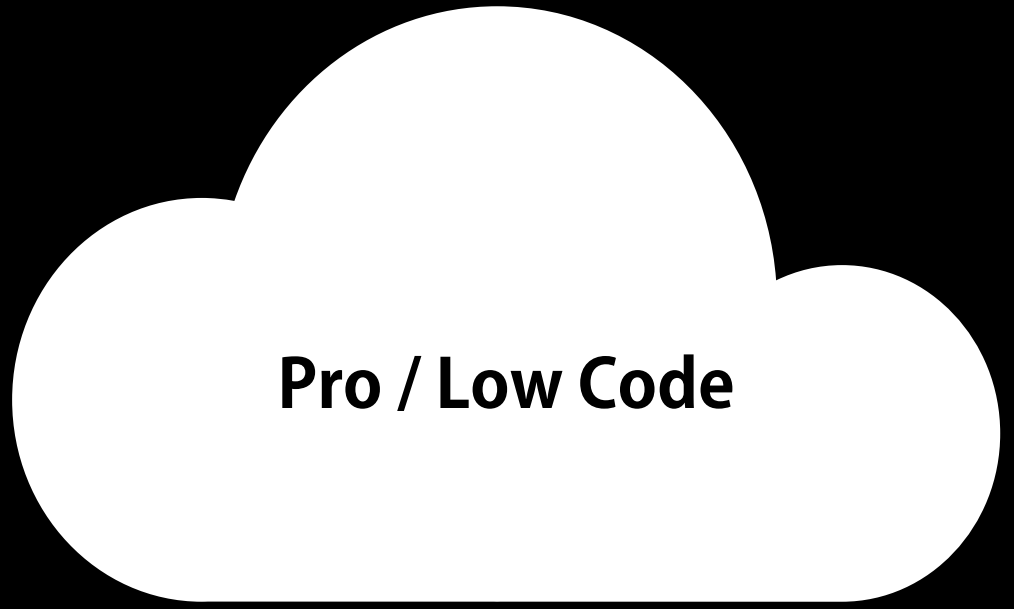
?

资源分发

资源加载

机遇与挑战

机遇 – 2019 年在发生哪些变化



机遇 – 变化背后的构建

专业　Codesandbox、Stackbiltz、Gitlab Web IDE、Ali Cloud IDE

辅助　Outsystems、Mendix、云凤蝶

机遇 – 变化背后的构建

专业　Codesandbox、Stackbiltz、Gitlab Web IDE、Ali Cloud IDE

辅助　Outsystems、Mendix、云凤蝶

机遇 – 变化背后的构建

专业 Codesandbox、Stackbiltz、Gitlab Web IDE、Ali Cloud IDE

辅助 Outsystems、Mendix、云凤蝶

三种态度

- 编辑器 / 画板
- 编辑器 / 画板 + 限制性的研发环境
- 编辑器 / 画板 + 开放性的研发环境

机遇 – 变化背后的构建

专业 Codesandbox、Stackbiltz、Gitlab Web IDE、Ali Cloud IDE

辅助 Outsystems、Mendix、云凤蝶

三种态度

- 编辑器 / 画板
- 编辑器 / 画板 + 限制性的研发环境
- 编辑器 / 画板 + 开放性的研发环境

两种方案

- 容器
- 基于浏览器的加载策略

机遇 – 变化背后的构建

机遇 – 变化背后的构建

两种方案

- 容器
- 基于浏览器的加载策略

机遇 – 变化背后的构建

两种方案
- 容器
- 基于浏览器的加载策略

两种类型
- 服务端能力的输出
- 客户端能力的释放

机遇 – 变化背后的构建

两种方案

- 容器
- 基于浏览器的加载策略

两种类型

- 服务端能力的输出
  - **优势:** 服务端拥有和本地研发环境一致化的环境
  - **缺点:** 即时性较差、效率较差、无法离线、成本高昂
- 客户端能力的释放

机遇 – 变化背后的构建

两种方案

- 容器
- 基于浏览器的加载策略

两种类型

- **服务端能力的输出**
  - **优势:** 服务端拥有和本地研发环境一致化的环境
  - **缺点:** 即时性较差、效率较差、无法离线、成本高昂
- **客户端能力的释放**
  - **优势:** 无服务端依赖、即时性、高效率、可离线运行
  - **缺点:** 无法输出系统级能力，所有能力建设都围绕着浏览器技术

机遇 – 面向未来

机遇 – 声音

Don’t Build That App!
Luke Jackson / April 4, 2019

A Future Without Webpack
Fred K. Schott / March 12, 2019

Bundless

机遇 – 声音背后 – Bundless 的实现

机遇 – 声音背后 – Bundless 的实现

— 模块加载器： 获取依赖 –》编译依赖

systemjs 0.21.x & JSPM 1.x

@stackblitz

@codesandbox

机遇 – 声音背后 – Bundless 的实现

— 模块加载器： 获取依赖 –》编译依赖

systemjs 0.21.x & JSPM 1.x
@stackblitz
@codesandbox

— Native-Module 加载方案

systemjs >= 3.x & JSPM 2.x
@pika/web

云 + Browser Based Bundless + Web NPM

Gravity

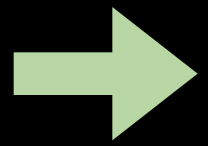
Gravity 的挑战

— 如何基于浏览器实现：

- nodejs 文件系统
- nodejs 文件 resolve 算法
- nodejs 内置模块
- 任意模块格式的加载
- 多媒体文件
- 单一文件多种编译方式
- 缓存策略
- 包管理
- ……

Gravity 的挑战

— 如何基于浏览器实现：



- nodejs 文件系统
- nodejs 文件 resolve 算法
- nodejs 内置模块
- 任意模块格式的加载
- 多媒体文件
- 单一文件多种编译方式
- 缓存策略
- 包管理
- ……

Gravity 的挑战

如何基于浏览器实现：

- nodejs 文件系统
- nodejs 文件 resolve 算法
- nodejs 内置模块
- 任意模块格式的加载
- 多媒体文件
- 单一文件多种编译方式
- 缓存策略
- 包管理
- ……

总结为：

- 如何设计资源文件的加载器
- 如何设计资源文件的编译体系
- 如何设计浏览器端的文件系统
- 如何设计浏览器端的包管理

Gravity 架构大图

我们现在走的路

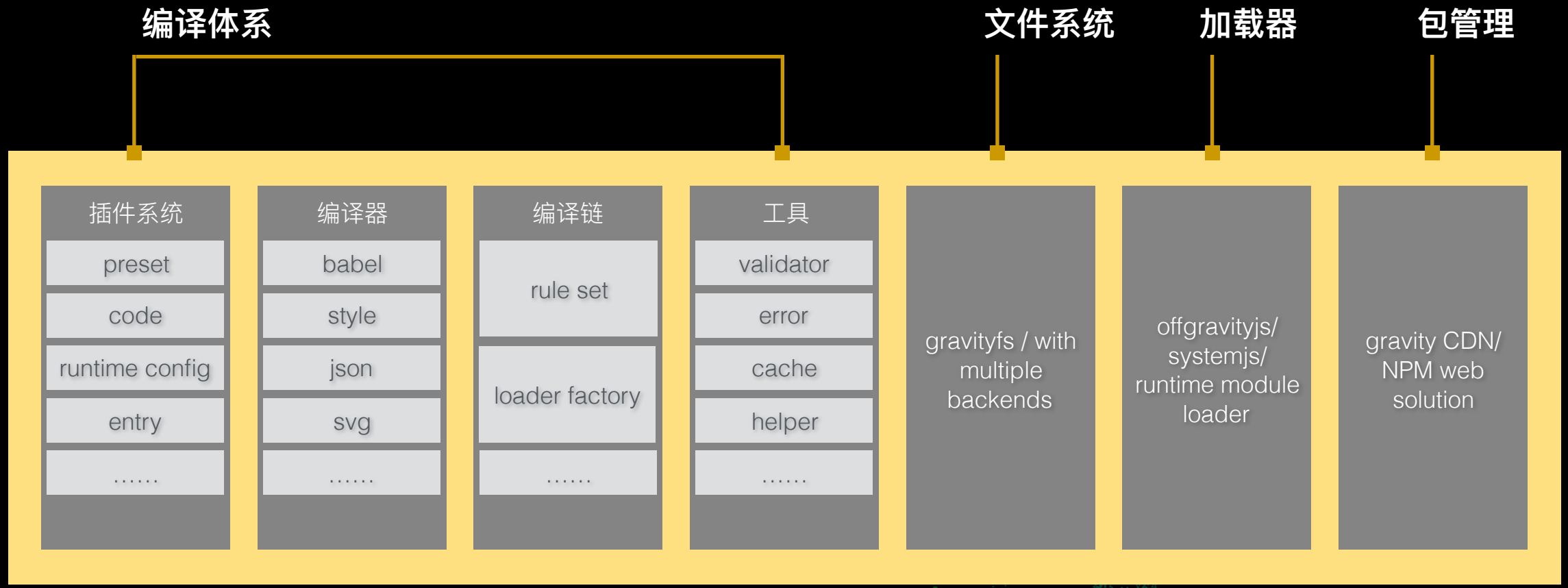
Gravity 核心

插件系统	编译器	编译链	工具类			
preset	babel	rule set	validator	gravityfs / with multiple backends	offgravityjs/ systemjs/ runtime module loader	gravity CDN/ NPM web solution
code	style		error			
runtime config	json	loader factory	cache			
entry	svg		helper			
……	……	……	……			

tapable	offgravityjs	browser-resolve	gravityfs	babel-standalone
browserfs	systemjs	less	…….	

BASE

核心在解决的问题


编译体系
文件系统
加载器
包管理
插件系统
preset
code
runtime config
entry
……
编译器
babel
style
json
svg
……
编译链
rule set
loader factory
……
工具
validator
error
cache
helper
……
gravityfs / with multiple backends
offgravityjs/ systemjs/ runtime module loader
gravity CDN/ NPM web solution

Transpiler

代码 A 转换为代码 B 转换器

Preset

是一份构建描述集合，该集合包含了模块加载器文件加载的描述，
转换器的描述，插件的描述等。

Ruleset

具体一个文件应该被怎么样的 transpilers 来转换。

名词解释

Transpiler

代码 A 转换为代码 B 转换器

```
const { fpath, code } = context;
const result = `module.exports = JSON.parse(${JSON.stringify( value: code || {})});`;

return Promise.resolve({
  result: {
    transpiledCode: result,
    filename: fpath,
  }
});
```

JSONTranspiler

Preset

是一份构建描述集合，该集合包含了模块加载器文件加载的描述，
转换器的描述，插件的描述等。

Ruleset

具体一个文件应该被怎么样的 transpilers 来转换。

名词解释

Transpiler

代码 A 转换为代码 B 转换器

Preset

是一份构建描述集合，该集合包含了模块加载器文件加载的描述，转换器的描述，插件的描述等。

Ruleset

具体一个文件应该被怎么样的 transpilers 来转换。

```
name: 'demo',
mode,
rules: [
  {
    test:  /\.(t|j)sx?$/,
    exclude: /\.d\.ts/,
    use: [
      {
        loader: 'babel-loader',
        transpiler: babelTranspiler,
        options: {
          ...babelOptions,
          config: {
            ...babelOptions.config,
            plugins: [
              ['proposal-decorators', { legacy: true }],
              ...babelOptions.config.plugins,
            ],
          },
        },
      },
    ]
  },
],
plugins: [
  new PresetGravityPlugin(),
  new CodeGravityPlugin(),
  new SystemConfigGravityPlugin(),
  new EntryGravityPlugin(),
],
map: {
  mfsLoader: '/static/loader/index.js',
```

demoPreset

名词解释

index.axml 对应的 Ruleset

```
/pages/index/index.axml 该文件会使用如下 transpiler 进行源码转换
▼(2) [{…}, {…}]
  ▼0:
     enforce: undefined
     type: "use"
   ▶value: {options: {…}, ident: "ref--0-1", loader: "appx-loader", transpiler: AppxTranspiler}
   ▶__proto__: Object
  ▼1:
     enforce: undefined
     type: "use"
   ▶value: {options: {…}, ident: "ref--0-0", loader: "babel-loader", transpiler: BabelTranspiler}
   ▶__proto__: Object
    length: 2
  ▶__proto__: Array(0)
```

Ruleset

具体一个文件应该被怎么样的 transpilers 来转换。

Transpiler

代码 A 转换为代码 B 转换器

Preset

是一份构建描述集合，该集合包含了模块加载器文件加载的描述，
转换器的描述，插件的描述等。

Ruleset

具体一个文件应该被怎么样的 transpilers 来转换。

Gravity 的消费链

载体

小程序 Web IDE

云凤蝶

……

Preset

小程序

appx

……

React

CRA

……

Vue

Vue-cli

……

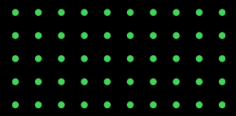
更多的
垂直业务
资源文件
加载策略描述

Gravity 核心

Gravity 到底是什么

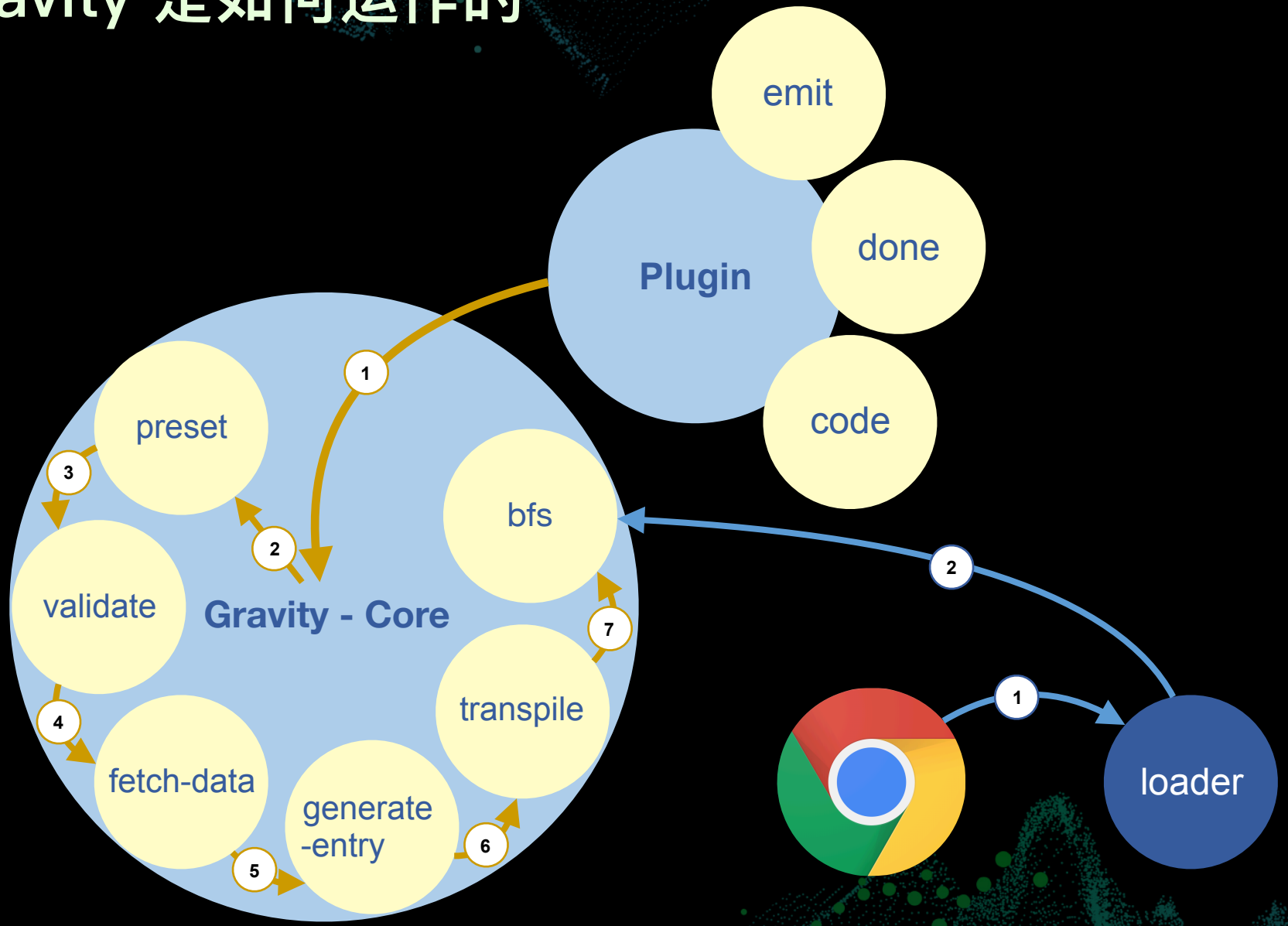
Gravity 是实现浏览器实时构建的事件流集合，它不指代某种通用业务。

专题深入



专题 1：插件机制

Gravity 是如何运作的



Gravity 是如何运作的

Gravity 本质上是事件流机制，它的核心流程就是将插件连接起来。

Gravity 是如何运作的

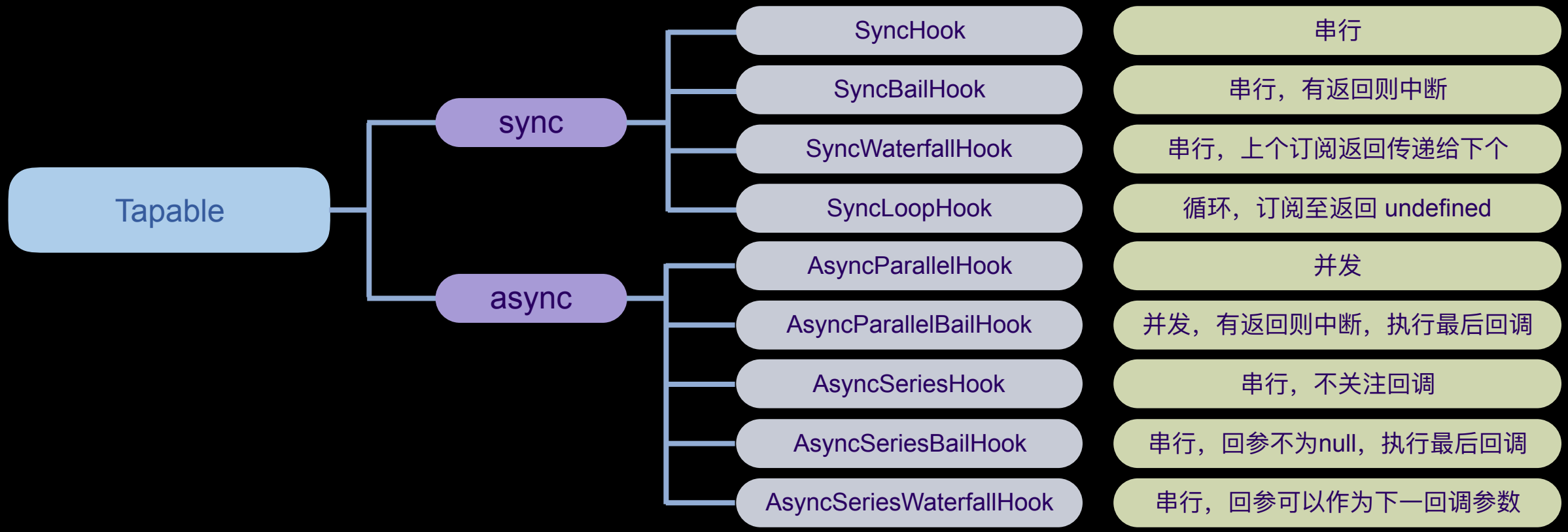
Gravity 本质上是事件流机制，它的核心流程就是将插件连接起来。

- 如何进行事件编排
- 如何保证事件执行的有序性
- 如何进行事件的订阅和消息的分发

似曾相识

Webpack <- Tapable

Tapable


Tapable
sync
async
SyncHook
SyncBailHook
SyncWaterfallHook
SyncLoopHook
AsyncParallelHook
AsyncParallelBailHook
AsyncSeriesHook
AsyncSeriesBailHook
AsyncSeriesWaterfallHook
串行
串行，有返回则中断
串行，上个订阅返回传递给下个
循环，订阅至返回 undefined
并发
并发，有返回则中断，执行最后回调
串行，不关注回调
串行，回参不为null，执行最后回调
串行，回参可以作为下一回调参数

举个例子

举个例子

Core - Plugin

```
1  const {
2    Tapable,
3    SyncHook,
4  } = require('tapable');
5
6  class Plugin extends Tapable {
7    constructor() {
8      super();
9      this.hooks = {
10       info: new SyncHook( args: ['env']),
11     }
12   }
13 }
14
15 export default Plugin;
```

1 申明一个可以被订阅的事件

举个例子

Core - Plugin

```
const {
  Tapable,
  SyncHook,
} = require('tapable');

class Plugin extends Tapable {
  constructor() {
    super();
    this.hooks = {
      info: new SyncHook(args: ['env']),
    }
  }
}

export default Plugin;
```

1 申明一个可以被订阅的事件

自定义插件

```
class InfoGravityPlugin {
  apply(gravity) {
    gravity.hooks.info.tap(options: 'info-gravity-plugin', fn: (env) => {
      console.log(`当前的 node 环境是 ${env.NODE_ENV}`);
    });
  }
}

export default InfoGravityPlugin;
```

2 订阅事件

举个例子

Core - Plugin

```
const {
  Tapable,
  SyncHook,
} = require('tapable');

class Plugin extends Tapable {
  constructor() {
    super();
    this.hooks = {
      info: new SyncHook( args: ['env']),
    }
  }
}

export default Plugin;
```

1 申明一个可以被订阅的事件

自定义插件

```
class InfoGravityPlugin {
  apply(gravity) {
    gravity.hooks.info.tap( options: 'info-gravity-plugin', fn: (env) => {
      console.log(`当前的 node 环境是 ${env.NODE_ENV}`);
    });
  }
}

export default InfoGravityPlugin;
```

2 订阅事件

Core

```
applyPlugins() {
  const plugins = this.buildInPreset && this.buildInPreset.plugins;
  if (plugins && Array.isArray(plugins)) {
    for (const plugin of plugins) {
      plugin.apply(this.plugin);
    }
  }
}

showInfo() {
  this.plugin.hooks.info.call(process.env);
}
```

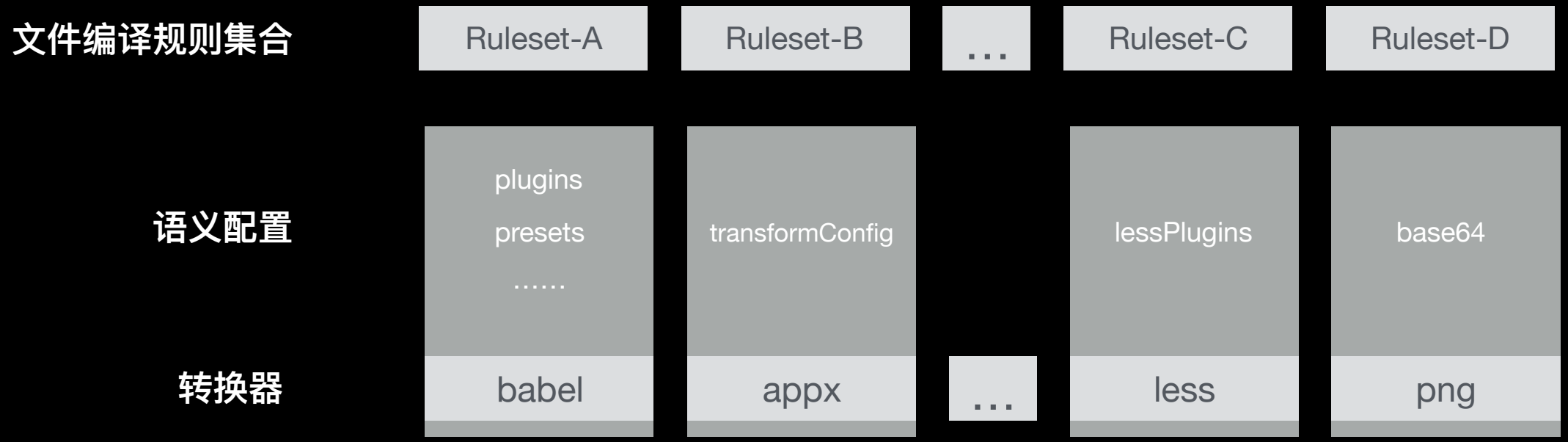
3 绑定订阅事件

4 分发事件

Gravity 的抽象

专题 2：如何实现编译链

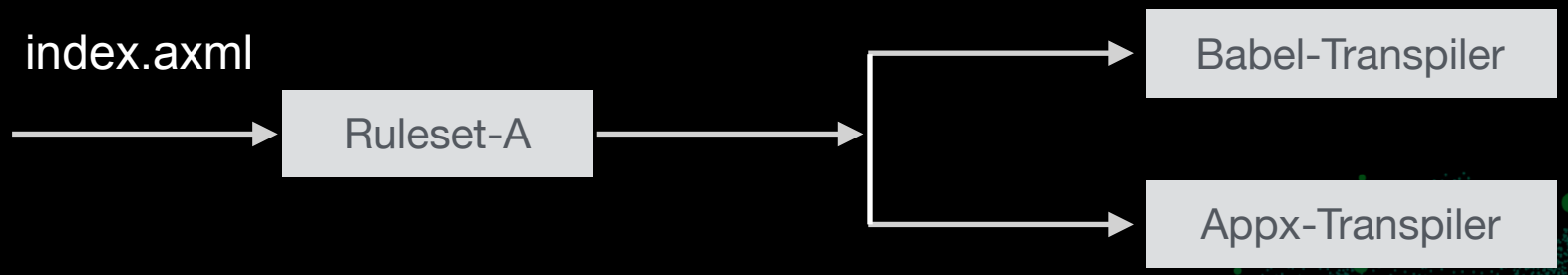
回看文件编译规则 Ruleset



Ruleset 的含义

index.axml 对应的 Ruleset

```
/pages/index/index.axml 该文件会使用如下 transpiler 进行源码转换
▼(2) [{…}, {…}]
  ▼0:
     enforce: undefined
     type: "use"
   ▶value: {options: {…}, ident: "ref--0-1", loader: "appx-loader", transpiler: AppxTranspiler}
   ▶__proto__: Object
  ▼1:
     enforce: undefined
     type: "use"
   ▶value: {options: {…}, ident: "ref--0-0", loader: "babel-loader", transpiler: BabelTranspiler}
   ▶__proto__: Object
    length: 2
  ▶__proto__: Array(0)
```



编译链的含义

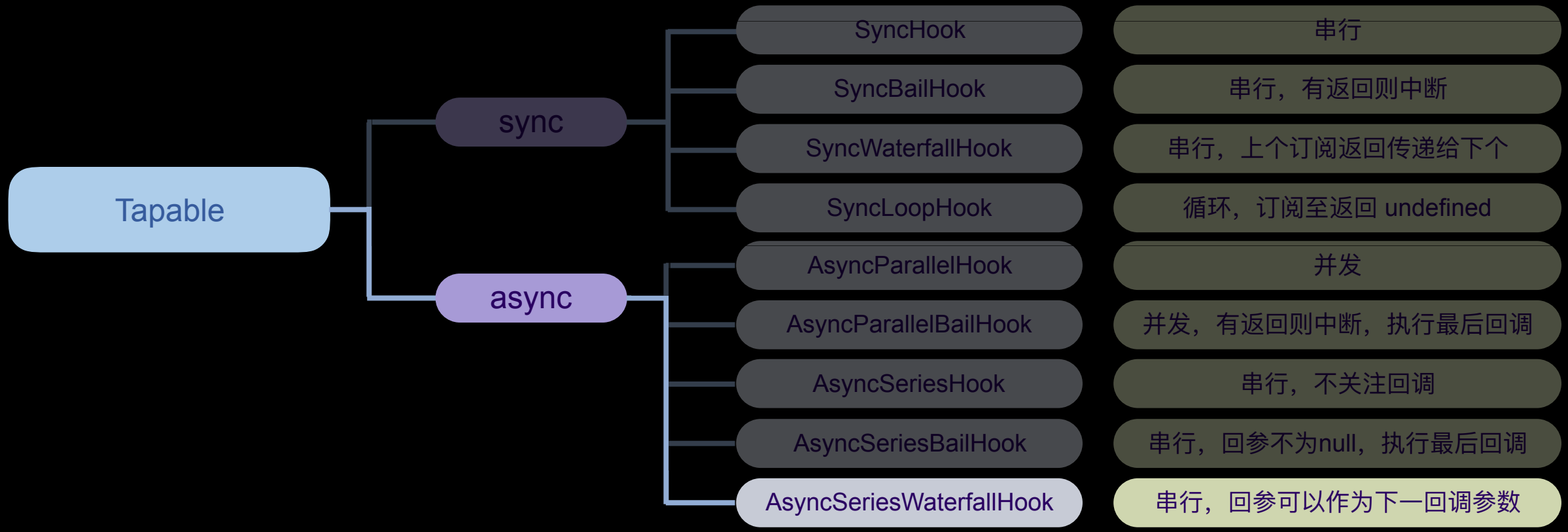

index.axml
result
result
Appx-Transpiler
Babel-Transpiler
下级消费链路

解决事件有序性的问题

似曾相识

Tapable

AsyncSeriesWaterfallHook



问题被简化为

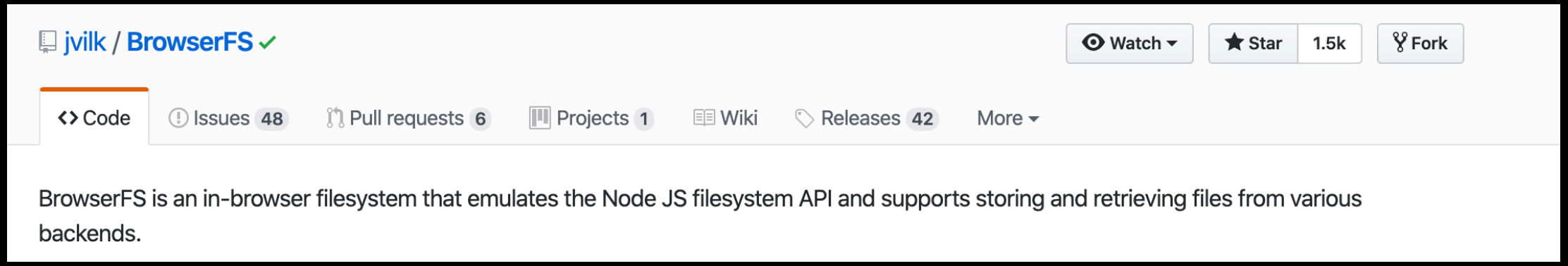
如何动态创建 AsyncSeriesWaterfallHook 事件，以及如何分发

专题 3：文件系统和包管理

文件系统

文件系统

站在巨人的肩膀上



文件系统


```
<script type="text/javascript" src="browserfs.min.js"></script>
<script type="text/javascript">
  // Installs globals onto window:
  // * Buffer
  // * require (monkey-patches if already defined)
  // * process
  // You can pass in an arbitrary object if you do not wish to pollute
  // the global namespace.
  BrowserFS.install(window);
  // Configures BrowserFS to use the LocalStorage file system.
  BrowserFS.configure({
    fs: "LocalStorage"
  }, function(e) {
    if (e) {
      // An error happened!
      throw e;
    }
    // Otherwise, BrowserFS is ready-to-use!
  });
</script>
```

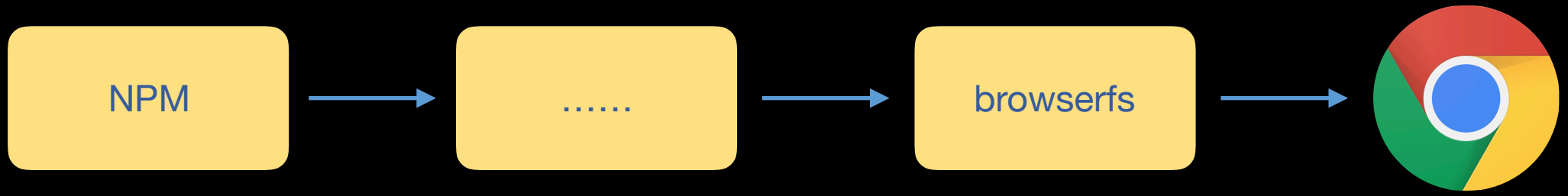
backends

```
var fs = require('fs');
fs.writeFile('/test.txt', 'Cool, I can do this in the browser!', function(err) {
  fs.readFile('/test.txt', function(err, contents) {
    console.log(contents.toString());
  });
});
```

FileSystem API

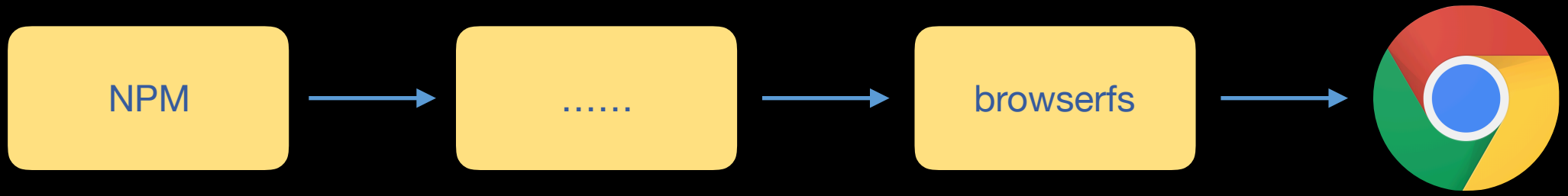
文件系统如何介入包管理

1 思路：浏览器内实现 NPM



文件系统如何介入包管理

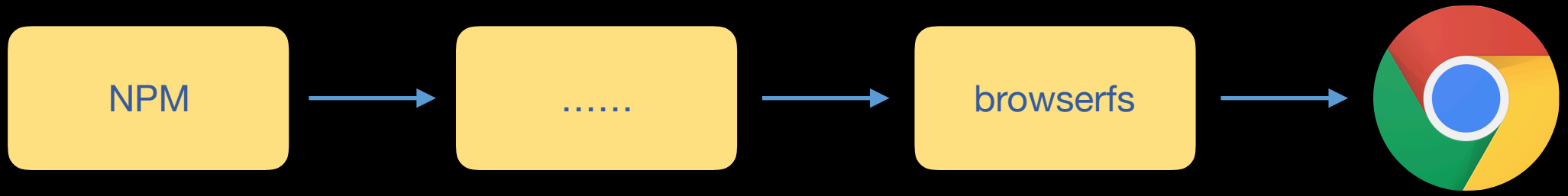
1 思路：浏览器内实现 NPM



- 包信息拉取
- 包依赖处理
- 包安装

文件系统如何介入包管理

1 思路：浏览器内实现 NPM

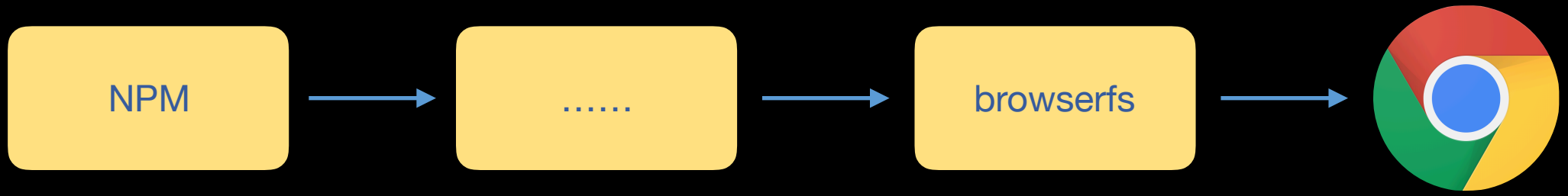


- 包信息拉取
- 包依赖处理
- 包安装

- 依赖拓扑结构
- 文件编译

文件系统如何介入包管理

1 思路：浏览器内实现 NPM



- 包信息拉取
- 包依赖处理
- 包安装

- 依赖拓扑结构
- 文件编译

- 存储

文件系统如何介入包管理

1 思路：浏览器内实现 NPM 的缺点：

文件系统如何介入包管理

1 思路：浏览器内实现 NPM 的缺点：

- 慢
- 存储量大
- 依赖 NPM Scripts 的包如何解决

文件系统如何介入包管理

1 **思路：浏览器内实现 NPM 的缺点：**

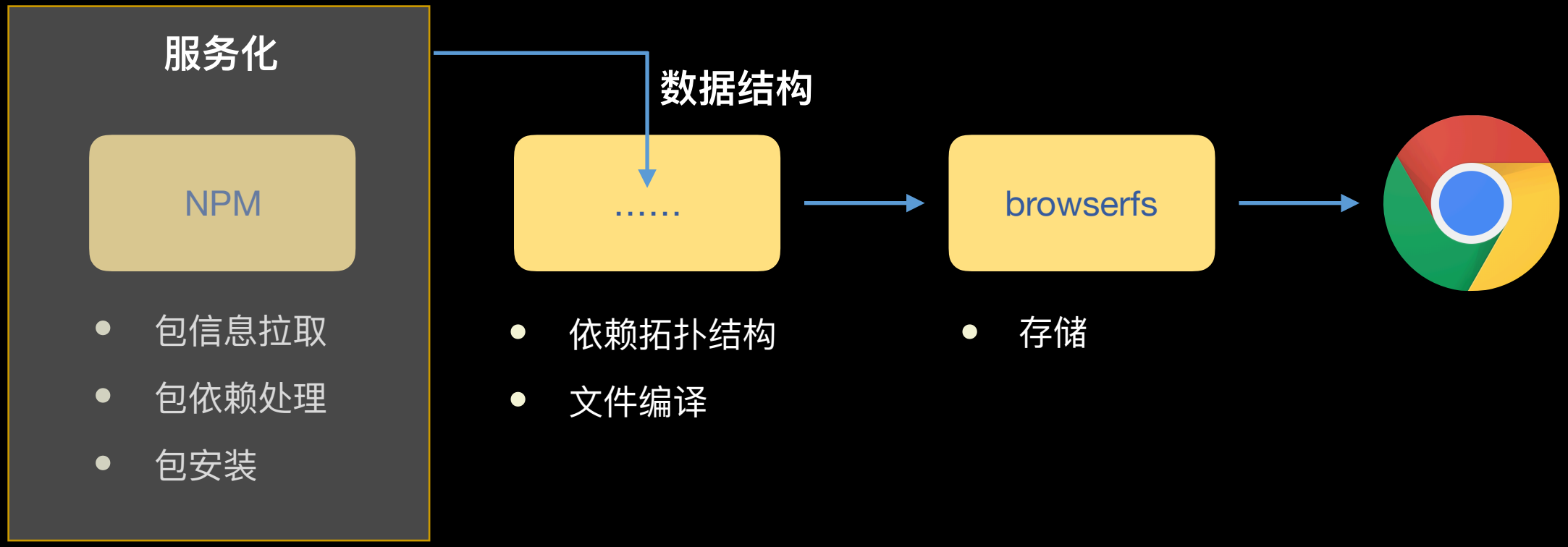
- 慢
- 存储量大
- 依赖 NPM Scripts 的包如何解决

完整的移植 NPM 在面向浏览器时这种包管理的形式是否合理?

Turbo CDN 引起的思考

https://medium.com/stackblitz-blog/introducing-turbo-5x-faster-than-yarn-npm-and-runs-natively-in-browser-cc2c39715403

Turbo CDN 引起的思考



基于网络的文件系统 – unpkg / jsdelivr

基于网络的文件系统 – unpkg / jsdelivr

结构获取

基于网络的文件系统 – unpkg / jsdelivr

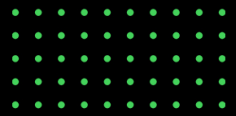
结构获取

文件获取

基于网络的文件系统 – 本质

基于网络的文件系统 – 本质

- 下发策略
- 桥接浏览器文件系统



基于网络的文件系统 – 本质

- **下发策略**
- **桥接浏览器文件系统**

- 依赖的下发策略
  - 基于项目维度
  - 基于入口文件分析的首次下发逻辑
  - 默认策略不满足时，动态下发
  - 数据结构记录依赖关系
- 缓存依赖信息
- 数据结构和浏览器文件系统建立桥接
  - 文件 resolve，读取

未来

- Pipelined 流水线化

  垂直业务场景所对应的 Preset 的产出，可以按着某个流程，用极少的成本自由组合一下就可以使用。

- Visualized 可视化

  所有搭建 Preset 、以及 Preset 内配置都为可视化方式露出

- Clouds 云化

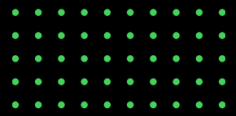
  能力服务化

总结

浏览器的实时构建

- 如何设计资源文件的加载器
- 如何设计资源文件的编译体系
- 如何设计浏览器端的文件系统
- 如何设计浏览器端的包管理

感谢聆听



演讲调查问卷

个人微信号