



VS CODE

百度内部实践

百度

张立理

2019-11

插件开发

01

复杂业务逻辑的串联与调度

02

VS Code命令式构建界面 vs 前端常见声明式界面

03




多样的事件源与日志采集

业务逻辑



 Setting up remote connection: Signing in...

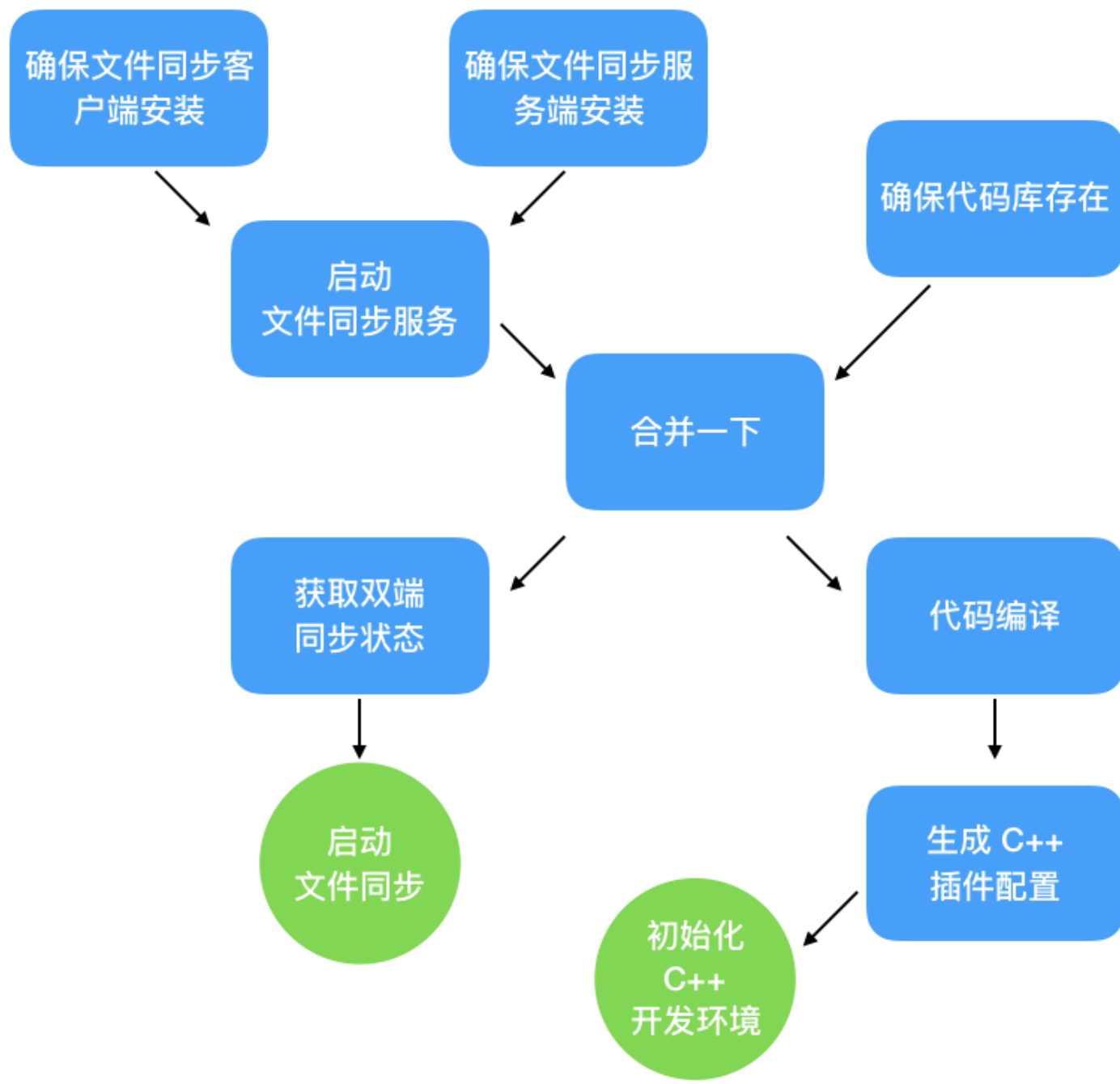
业务的执行器

-  多个业务流程之间可能会有可复用的任务
-  一个业务流程可以被中止，切换到其他流程
-  超时的流程要能被监控并作出响应

状态的监听器

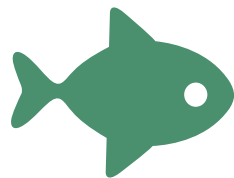
-  日志服务
-  视图更新

业务逻辑



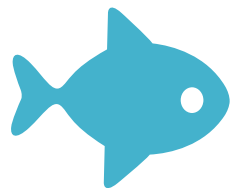
抽象与简化

所有任务和其构成的
业务之间的关系



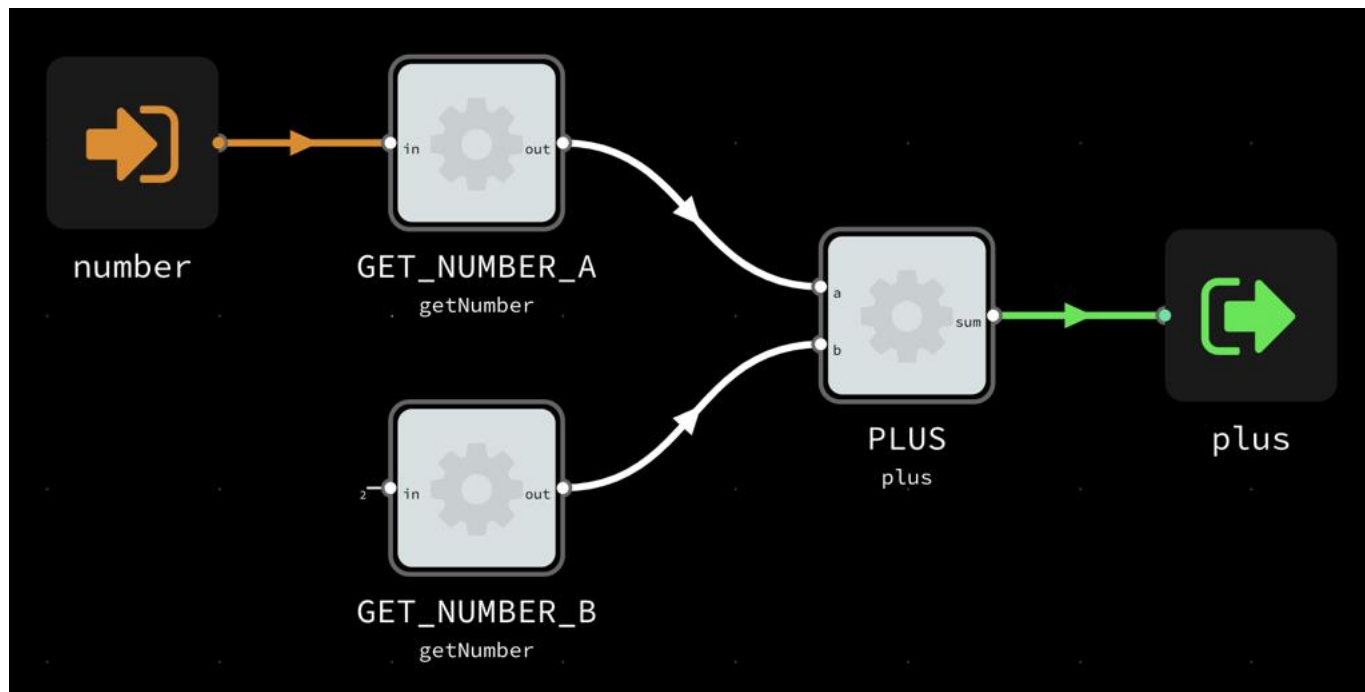
一张图

所有任务



异步任务

怎么用图



```
{  
  // 图中依赖的外部输入，橘黄色点  
  "inports": { ...  
},  
  // 图的输出，绿色点  
  "outports": { ...  
},  
  // 图中的任务块，灰色点  
  "processes": { ...  
},  
  // 图中各个点之间的连线  
  "connections": [ ...  
]  
}
```

```
1  {  
2  .. "inports": {  
3    .... "number": {  
4      .... .. "process": "GET_NUMBER_A",  
5      .... .. "port": "in"  
6    .... }  
7  },  
8  .. "outports": {  
9    .... "plus": {  
10     .... .. "process": "PLUS",  
11     .... .. "port": "sum"  
12    .... }  
13  },
```


flowbased / fbp

Watch ▾

16

★ Star

80

🔗 Fork

18

Code

Issues 7

Pull requests 0

Actions

Projects 0

Wiki

Security

Insights

FBP flow definition language parser

parser

dsl

242 commits

7 branches

0 packages

17 releases

15 contributors

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾



bergie Merge pull request #93 from LuisKen/master ...

Latest commit 5784fd3 on 15 Oct

bin	Implement fixes from @jonnor	3 years ago
browser	Webpack build setup	3 years ago
grammar	fix: fix the bug that use inports data in outports	2 months ago
lib	fix: fix the bug that use inports data in outports	2 months ago

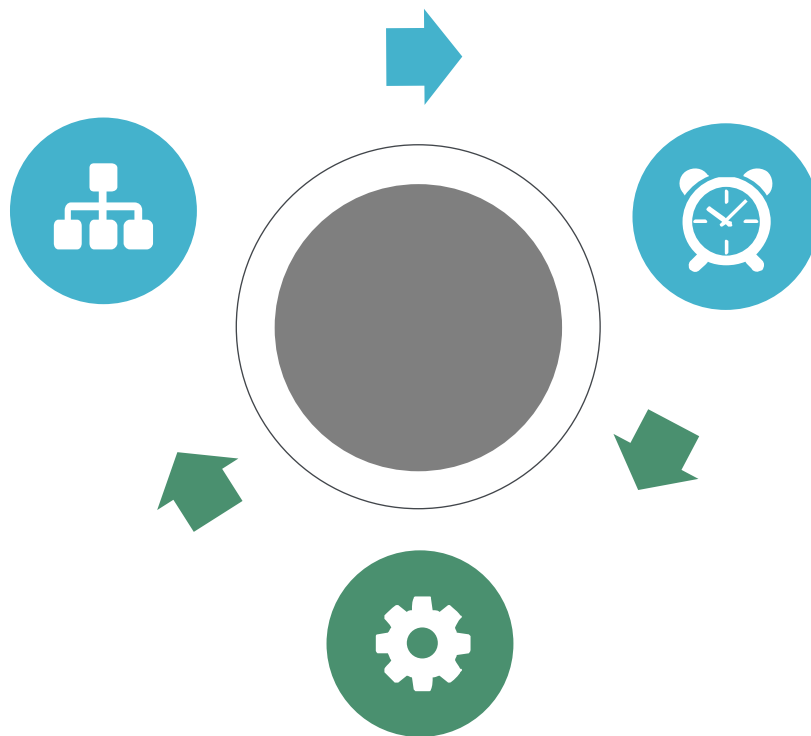
```
1  INPORT=GET_NUMBER_A.IN:number
2  OUTPORT=PLUS(plus).SUM:plus
3  GET_NUMBER_A(getNumber) OUT -> A PLUS
4  2 -> GET_NUMBER_B(getNumber) OUT -> B PLUS
```

基于 DSL fbp 的图表述

可一键生成可交互的图，
相对于 JSON 表述降低了阅读和维护成本

基于事件机制

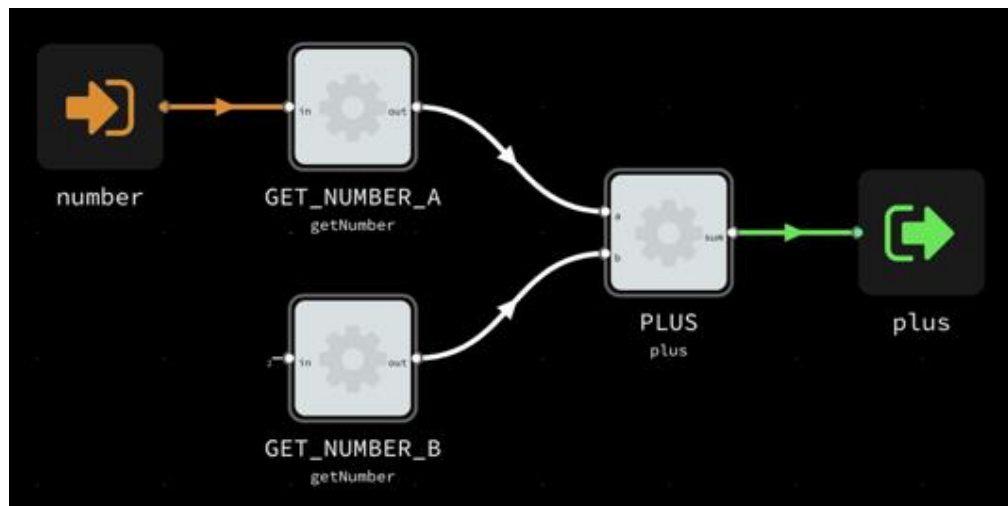
工作流内部逻辑流转外部可感知，
做到了业务逻辑与UI、统计逻辑的解耦



作为运行时

提供了超时报警、错误报警等功能
提供了完整的错误栈以降低定位调试成本

Workflow

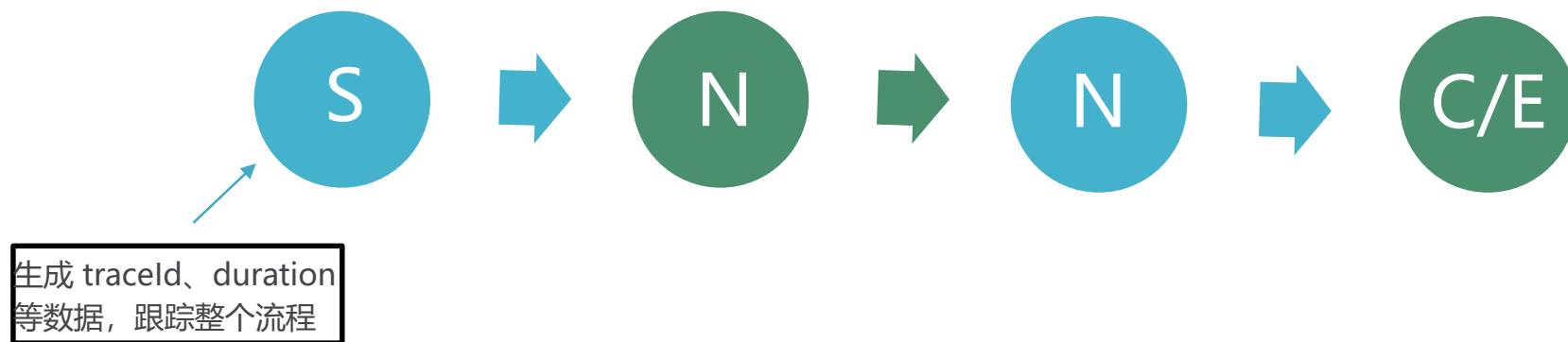


```
1  const {plus} = createWorkflowFromGraph(  
2    ...graph,  
3    ...components  
4  );  
5  
6  plus.run((e) => {  
7    ...console.log(e);  
8  }, {  
9    ...number: 1  
10   });  
13  ...};  
14  }  
15  
16  const components = {getNumber, plus};
```

Notification 对象

```
interface WorkflowNotification {  
    ... // 当前执行的 FbpComponent.component  
    ... id: string;  
    ... kind: NOTIFICATION_TYPE;  
    ... context: Array<[string, WorkflowContextFields]>;  
    ... payload: WorkflowPayload;  
    ... value?: any;  
}  
  
const enum NOTIFICATION_TYPE {  
    ... start = 'S',  
    ... next = 'N',  
    ... error = 'E',  
    ... complete = 'C'  
}
```

4 种 Notification



符合预期的任务比例

C / S

失踪的任务

$S - (C + E)$



CancellationToken

<https://github.com/tc39/proposal-cancellation> Stage-1



Polyfill

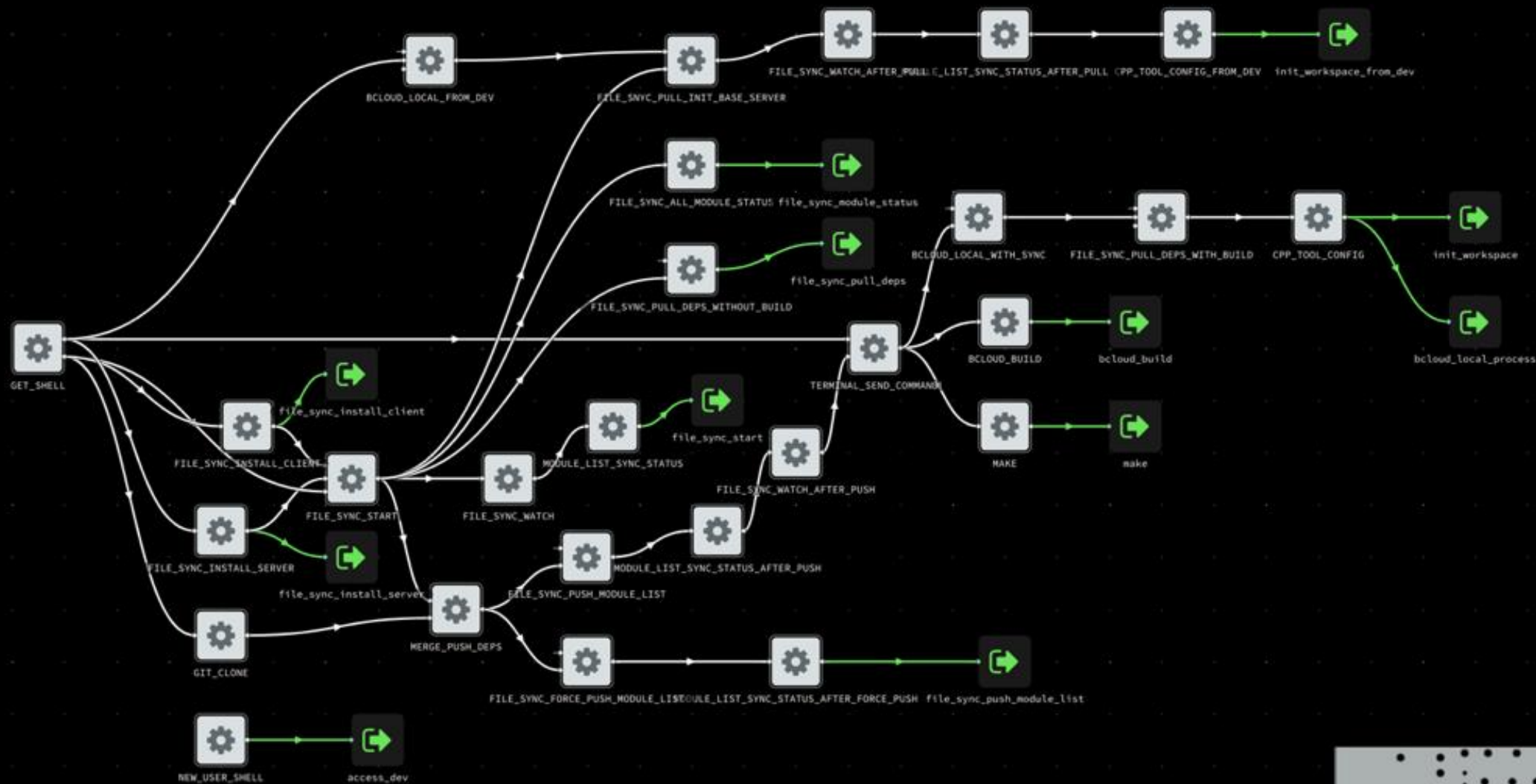
<https://github.com/conradreuter/cancellationtoken#readme>

// 异步任务内部的使用

```
function runAsyncTask(deps, token) {  
  ... const {timeout, value} = deps;  
  ... return new Promise(resolve => {  
    ...   const timer = setTimeout(() => {  
    ...     resolve(value);  
    ...   }, timeout);  
    ...   token && token.onCancelled(() => {  
    ...     clearTimeout(timer);  
    ...   });  
    ... });  
  }  
}
```

联动VS Code Progress

```
vscode.window.withProgress(option, progress => new Promise((resolve) => {  
    ...progress.report(getDisplayName(eventKey, RUNNER_START_ID));  
    ...iCodingRunnerEventEmitter.on(eventKey, (e: WorkflowNotification) => {  
        ...if (e.kind === NOTIFICATION_TYPE.error) {  
            ...// 处理错误展示  
            ...return;  
        }  
        ...// 更新进度条内容  
        ...const progressInfo = getDisplayName(eventKey, e.id);  
        ...if (!progressInfo) {  
            ...return;  
        }  
        ...progress.report(getDisplayName(eventKey, e.id));  
  
        ...if (e.kind === NOTIFICATION_TYPE.complete) {  
            ...vscode.window.showInformationMessage(`${eventKey} 已完成 (*^o^*)'`);  
            ...resolve();  
        }  
    ...});  
}));
```

插件开发

01

复杂业务逻辑的串联与调度

02

VS Code命令式构建界面 vs 前端常见声明式界面

03

多样的事件源与日志采集

TreeView

REPOSITORIES

- vscode-plugin master • Last fetched Nov
 - master — origin/master
 - Compare master (working) with <branch>
 - Branches
 - Contributors
 - Incoming Activity experimental
 - Remotes
 - Stashes

```
class TreeDataProvider
... implements vscode.TreeDataProvider<any>, vscode.Disposable {
... private disposables: vscode.Disposable[] = [];
... private _onDidChangeTreeData = new vscode.EventEmitter<any>();
... readonly onDidChangeTreeData? = this._onDidChangeTreeData.event;

... constructor() { ...
... }

... getChildren(element: any) { ...
... }

... getTreeItem(element: any) { ...
... }

... dispose() { ...
... }
};
```

TreeView

百度 ICODING: 命令面板

✓ 百度 C++

bcloud init workspace

bcloud build

bcloud local

bcloud make

bcloud config json

bcloud make clean

bcloud target BCLLOUD

```
<root>
... <tree-item label="百度 C++">
...   <tree-item
...     label="bcloud init workspace"
...     command="bcloud.initws"
...   />
...   <tree-item
...     label="bcloud build"
...     command="bcloud.build"
...   />
...   <tree-item
...     label="bcloud local"
...     command="bcloud.local"
...   />
...   <tree-item
...     label="bcloud make"
...     command="bcloud.make"
...   />
... />
```

插件开发

01



复杂业务逻辑的串联与调度

02

VS Code命令式构建界面 vs 前端常见声明式界面

03

多样的事件源与日志采集

-  编码是研发流程中非常核心的环节，IDE 作为承载这一场景的产品，其数据对于分析工程师研发行为非常重要
-  收集用户在 IDE 内部的行为并进行分析，尝试对研发过程中的各个环节进行优化



VS Code提供众多交互事件，用户行为得到了细致的分解



事件以subscribe的形式提供，模式统一利于抽象



事件之间存在上下游关系，与用户专注程度紧密联系



在时间轴上持续发生的事件可串联起来，形成编程现场

vscode 交互事件

- 调试(debug)
 - 修改了调试会话(onDidChangeActiveDebugSession)
 - 修改了断点(onDidChangeBreakpoints)
 - 调试插件发出的基于“调试适配器协议”的事件(onDidReceiveDebugSessionCustomEvent)
 - 调试启动(onDidStartDebugSession)
 - 调试结束(onDidTerminateDebugSession)
- 语言(languages)
 - 代码检查工具检查结果更新(onDidChangeDiagnostics)
- 任务(tasks)
 - 任务结束(onDidEndTask)
 - 任务进程结束(onDidEndTaskProcess)
 - 任务启动(onDidStartTask)
 - 任务进程启动(onDidStartTaskProcess)
- 窗口(window)
 - 当前使用的 Terminal 变化 (onDidChangeActiveTerminal)
 - 当前编辑的 TextEditor 变化 (onDidChangeActiveTextEditor)
 - 当前 TextEditor 的配置项变化 (onDidChangeTextEditorOptions)
 - 当前 TextEditor 的文本选区变化 (onDidChangeTextEditorSelection)
 - 当前 TextEditor 选中的列变化 (onDidChangeTextEditorViewColumn)
 - 当前 TextEditor 的可视范围变化 (onDidChangeTextEditorVisibleRanges)
 - 当前可视状态的 TextEditor 变化 (onDidChangeVisibleTextEditors)
 - 当前 window 的 focus 状态变化 (onDidChangeWindowState)
 - 关了一个 Terminal (onDidCloseTerminal)
 - 开了一个 Terminal (onDidOpenTerminal)
- 工作区(workspace)
 - 工作区配置修改 (onDidChangeConfiguration)
 - 文档文本（通常指源码文件）修改 (onDidChangeTextDocument)
 - 修改了工作区的文件夹 (onDidChangeWorkspaceFolders)
 - 关闭了一个文件或文件的语言选项修改（纯文本 -> markdown） (onDidCloseTextDocument)
 - 打开了一个文件或文件的语言选项修改 (onDidOpenTextDocument)
 - 保存了一个文件的内容到硬盘 (onDidSaveTextDocument)
 - 一个文件的内容将要保存到硬盘 (onWillSaveTextDocument)

VSC Event + Rxjs


```

/**
 * 创建符合 vscode 事件 api 的 observable
 *
 * @param addListener 注册监听函数, 如 vscode.window.onDidChangeWindowState
 * @param eventIdentifier 事件 id
 * @return 事件对应的 observable
 */
export const createVSCEventObservable = <T>({
  ...addListener: vscode.Event<T>,
  ...eventIdentifier: string
}): rxjs.Observable<CodingVSCEvent<T>> => {
  ...return rxjs.Observable.create((observer: rxjs.Observer<CodingVSCEvent<T>>) => {
    ...const registration = addListener((event: T) => {
      ...const payload = {
        ...event,
        ...eventIdentifier,
        ...timestamp: (new Date()).getTime()
      };
      ...observer.next(payload);
    });
    ...return () => registration.dispose();
  });
};

const changeBreakpoints = createVSCEventObservable(
  ...vscode.debug.onDidChangeBreakpoints,
  ...'debug.onDidChangeBreakpoints'
);

```

```
rxjs.merge(  
  ...Object.values(observables)  
)  
)  
.pipe(  
  ...filter((event: any) => event.eventIdentifier !== 'window.onDidChangeTextEditorVisibleRanges'),  
  ...bufferWhen(() => rxjs.interval(minutesGap * 60 * 1000)),  
  ...map((events: any) => {  
    ...const detail = events.reduce((map: {[key: string]: number}, eventItem: any) => {  
      ...const eventCount = map[eventItem.eventIdentifier] || 0;  
      ...map[eventItem.eventIdentifier] = eventCount + 1;  
      ...return map;  
    }, {});  
    ...return {  
      ...name: VSCODE_SLEEP_STATE,  
      ...content: {  
        ...detail,  
        ...state: events.length === 0 ? 'on' : 'off',  
        ...count: events.length  
      }  
    };  
  }  
)  
  ...filter((event: any) => event.content.count > 0)  
)  
.subscribe(codeServerTrackOperation);
```

Red bars of varying lengths and positions.

Blue bars of varying lengths and positions.

Blue bars of varying lengths and positions.

Red bars of varying lengths and positions.

Blue bars of varying lengths and positions.



2019

THANKS

百度