Vue声明式编程

摘要

- 声明式与图灵完备性
- 声明式与UI编程的演进
- 声明式与UI架构
- 声明式与交互

声明式与图灵完备性

声明式

- 我们对语言有很多划分的维度
- 声明式 vs 命令式
- 声明式语言：HTML XML CSS SQL
- 命令式语言：C++ Java JavaScript

图灵完备性

- 图灵完备：“可计算性”
- 命令式的图灵完备性：if/for， if/goto
- 声明式的图灵完备性：if/递归

Vue的Template的图灵完备性

- 实际上Vue的组件系统具有图灵完备性
- 使用Vue计算阶乘
- 使用Vue计算斐波那契数列

声明式与UI编程

UI编程

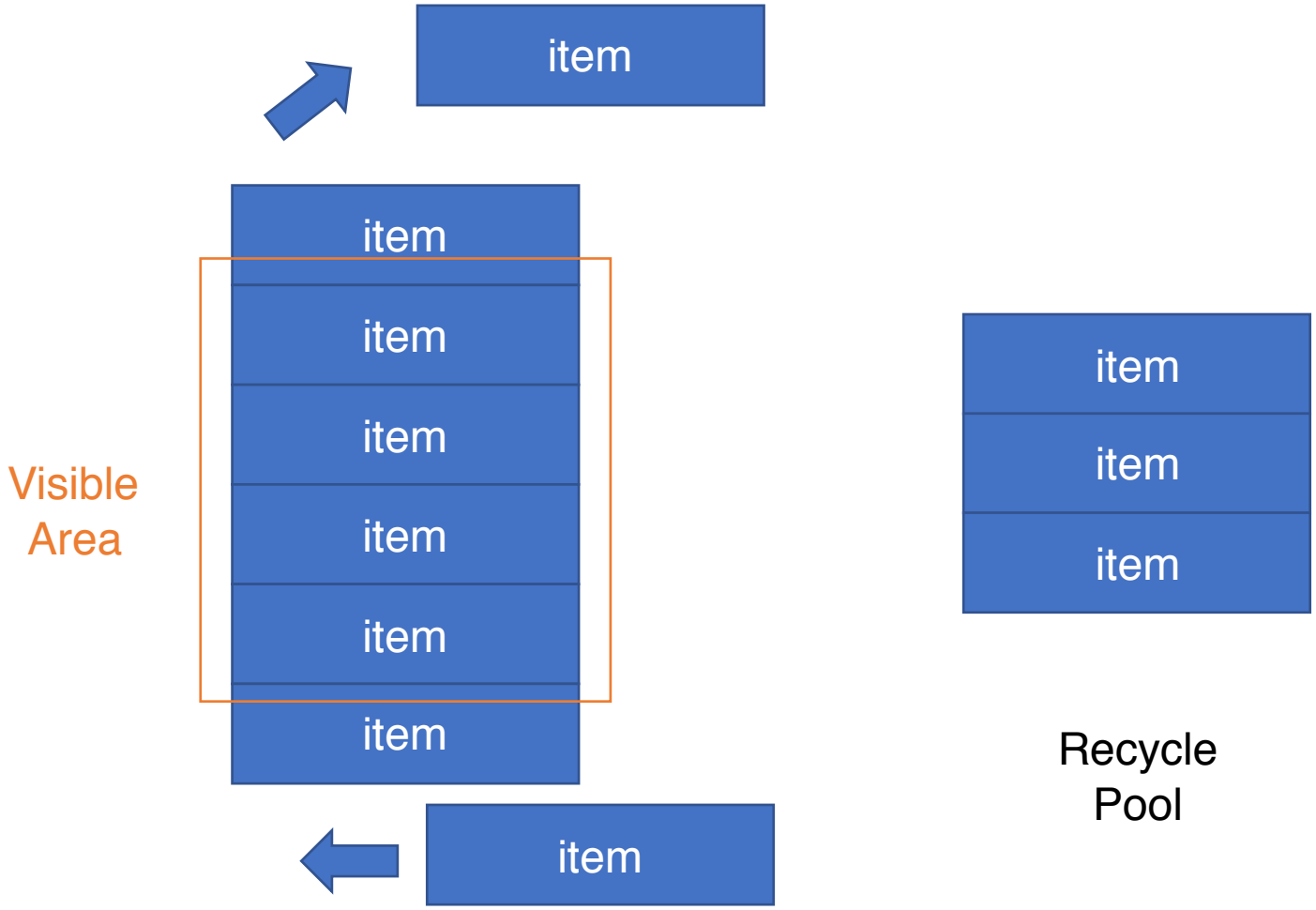
- 70年代 MVC诞生 人们意识到视图应该被独立抽象
- 80年代 标记语言大热 最初更与文本相关
- 90年代 可视化编辑器出现 “独立的UI代码文件”
- 2000年后 markup language+ programming language
- 2009年 “HTML5”

声明式优势

- 可视化的“可逆性”
- 声明式与可再绑定性
- 听说近年某框架把html和css塞回JS了？

可重复绑定性

- recycle-list



声明式与UI架构

MVVM

- MVC => MVP => MVVM 一脉相承的演化
- MVVM是为声明式/多语言量身定做的编程模型

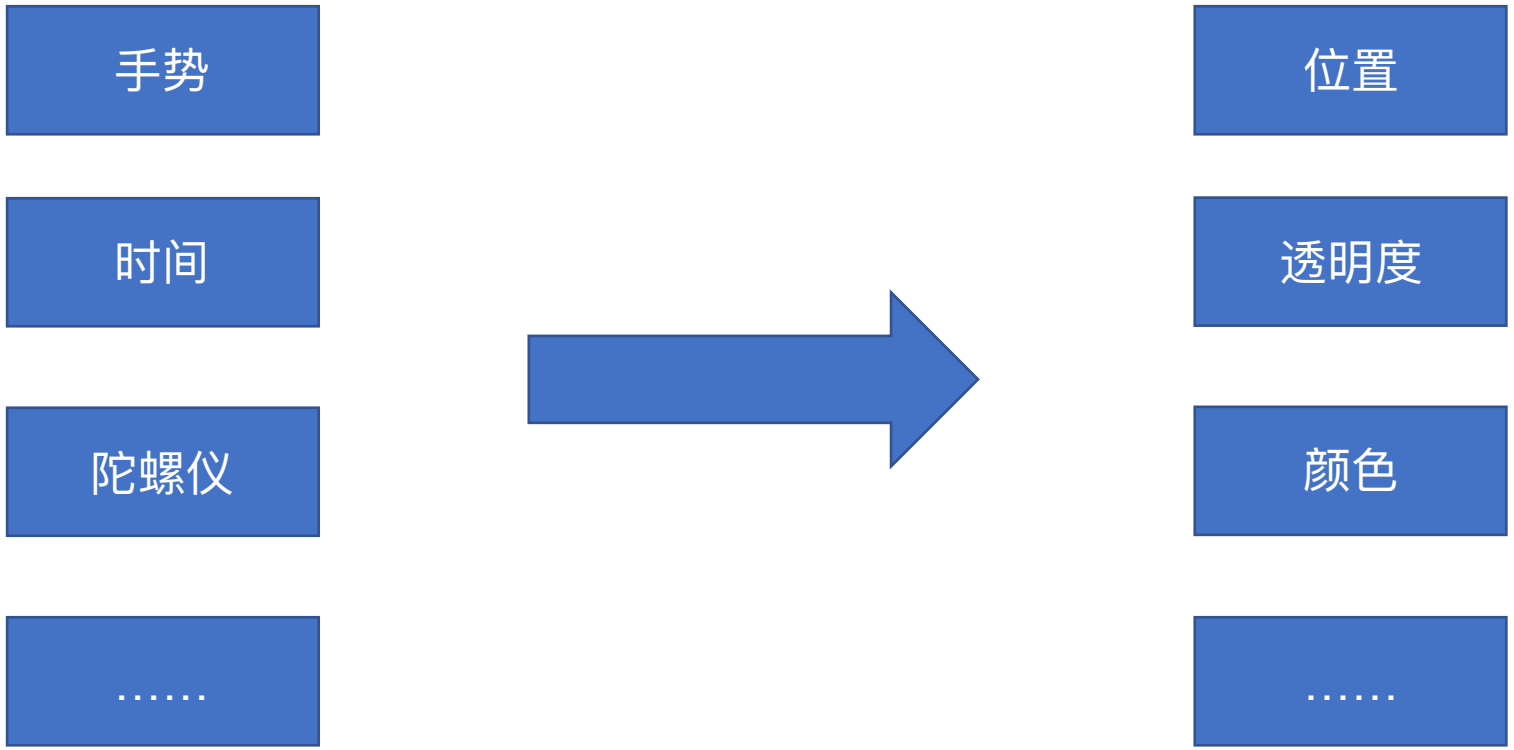
MVVM的数据绑定

- 数据绑定是声明式的数据与UI通讯
- 一些逻辑中数据绑定不需要命令式代码

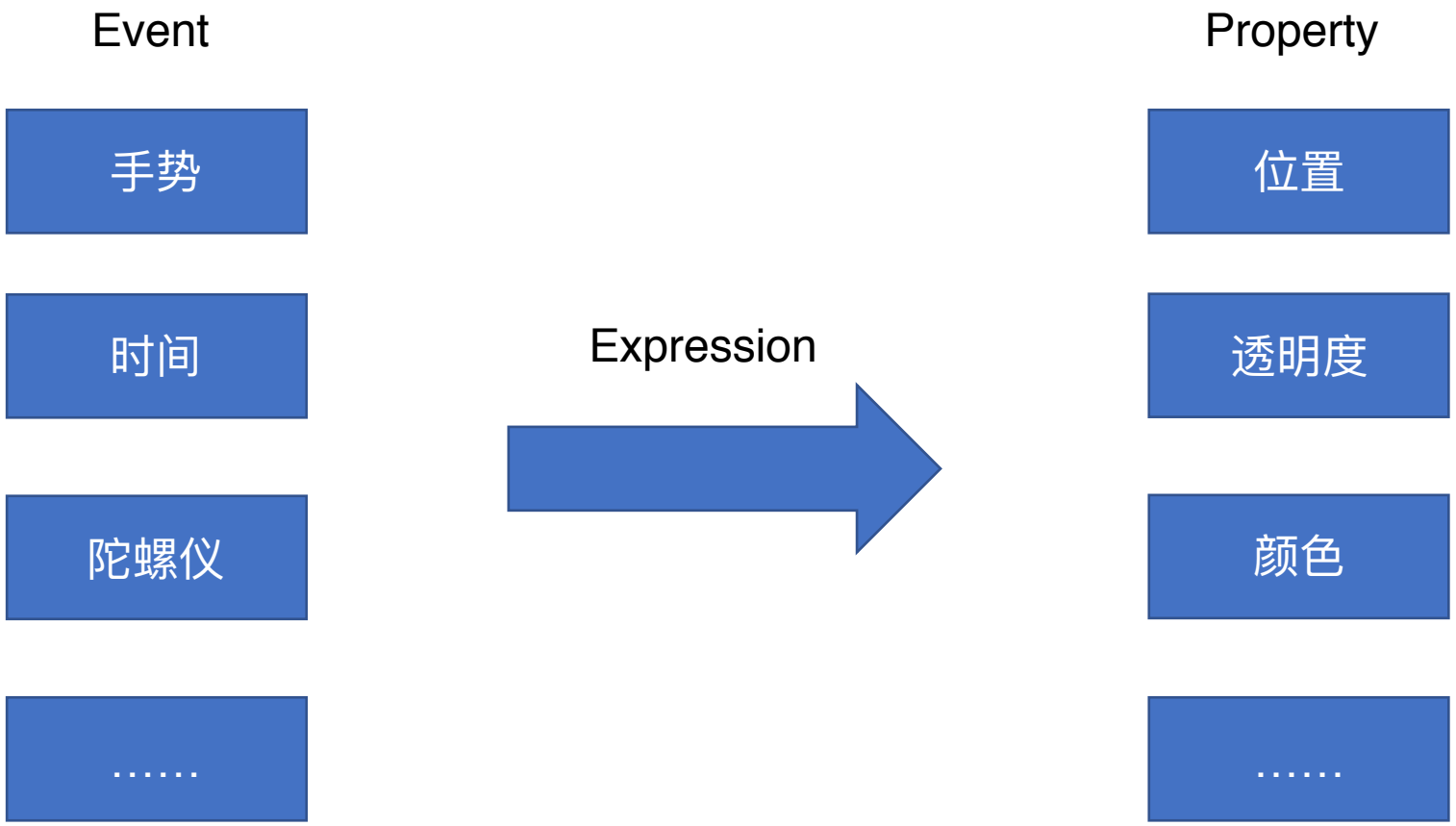
声明式数据交换

```
<template>
  <div id="app">
    <div v-bind:style="{ margin:'auto', width:'100px', height:'100px',
      backgroundColor:`rgb(${r}, ${g}, ${b})` }" ></div>
    <input type=range v-model=r step=1 min=0 max=255 ><br/>
    <input type=range v-model=g step=1 min=0 max=255 ><br/>
    <input type=range v-model=b step=1 min=0 max=255 ><br/>
  </div>
</template>

<script>


export default {
  name: 'App',
  data: () => {
    r: 100,
    g: 100,
    b: 100
  })
}
</script>
```

交互的抽象



交互的抽象



使用Vue Directive给元素添加行为

- Vue directive

```
<div v-time:style="{ left: 'Math.sin(t / 100) * 30 + \'px\'' }"
```

v-<event name>:<property name> = expression

使用Vue Directive给元素添加行为

- Vue directive

```
Vue.directive('time', {
  // 当被绑定的元素插入到 DOM 中时……
  inserted: function (el, info, vnode) {
    var startTime = Date.now();
    requestAnimationFrame(function update(){
        if(info.arg == 'style') {
            var props = new Function( "return " + info.expression)();
            for(var p in props)
                el.style[p] =  (new Function("t", "return " + props[p]))(Date.now() - startTime)
        }

        requestAnimationFrame(update);
    })
  }
})
```

Q & A