

拖拽那些事儿



# 拖拽那些事儿

The Events of Drag



# 自我介绍

- 赵雨森
- 杭州研究院 - 前端技术部
- 2014年入职
- 闷骚男，爱摄影，组件控
- [rainfore@github](https://github.com/rainfore), [hzzhaoyusen@corp](mailto:hzzhaoyusen@corp)



# 拖拽概述



什么是拖拽



# 什么是拖拽





# 什么是拖拽



**拖拽**是指抓住一个物体并让其跟随手移动的行为



# 拖拽的定义

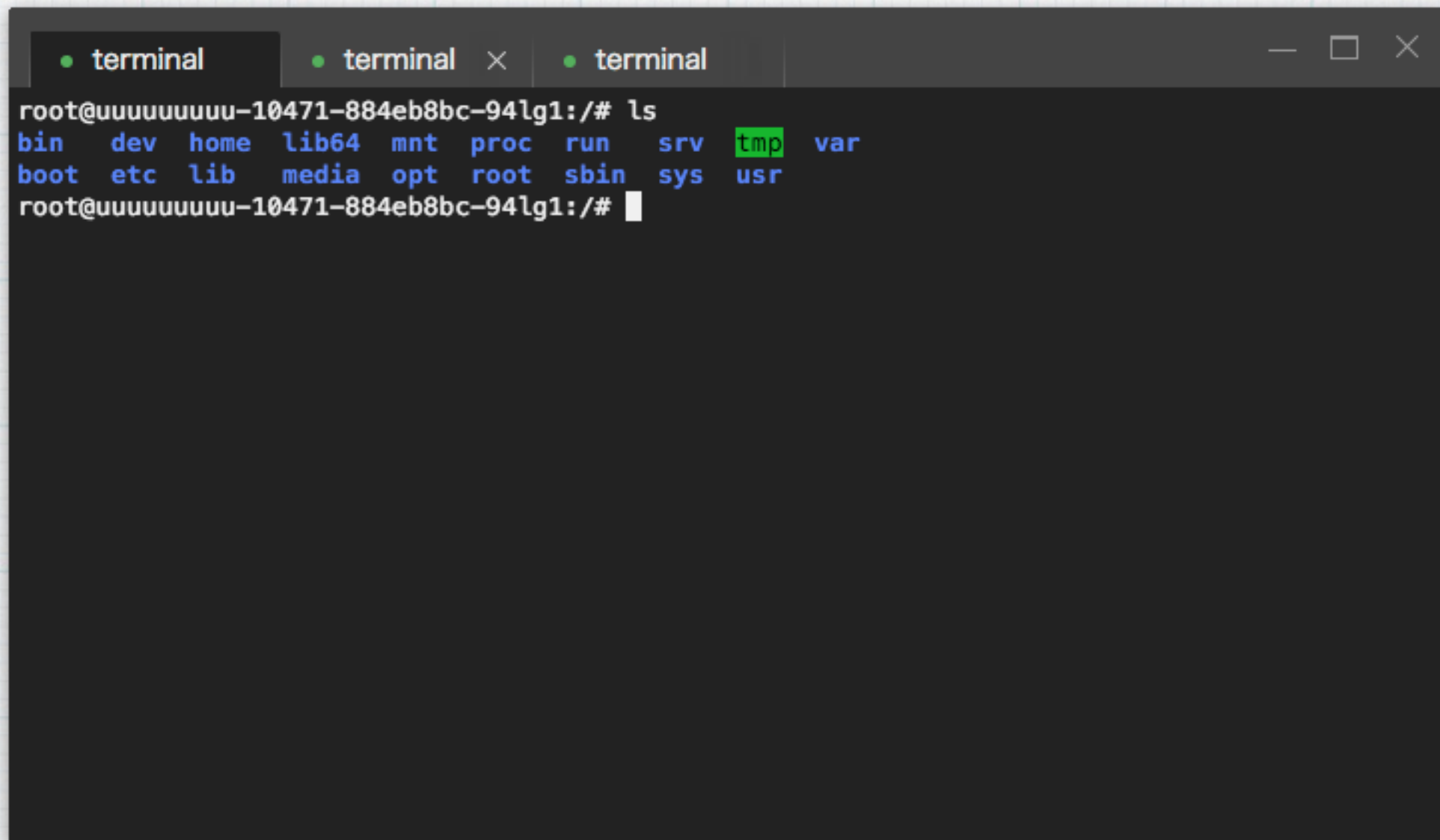


# 拖拽的定义

在人机交互中，**拖拽**是指点击一个元素  
并让其跟随鼠标（手指）移动的操作



## ● 网易蜂巢 - WebTerminal



The image shows a screenshot of a WebTerminal window. The window has a dark gray title bar with three tabs labeled "terminal" and standard window control buttons (minimize, maximize, close) on the right. The terminal content shows a root user at a machine with IP 10471-884eb8bc-94lg1. The user has executed the "ls" command, displaying a list of system directories in two rows. The "tmp" directory is highlighted in green in the original image.

```
root@uuuuuuuuuu-10471-884eb8bc-94lg1:/# ls
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@uuuuuuuuuu-10471-884eb8bc-94lg1:/#
```



# 拖拽的两种类型

- 拖移 (DragAndMove)
- 拖放 (DragAndDrop)



拖移(DragAndMove)



# 拖移(DragAndMove)

Send Trackbacks

Send trackbacks to:

(Separate multiple URLs with spaces)

Custom Fields

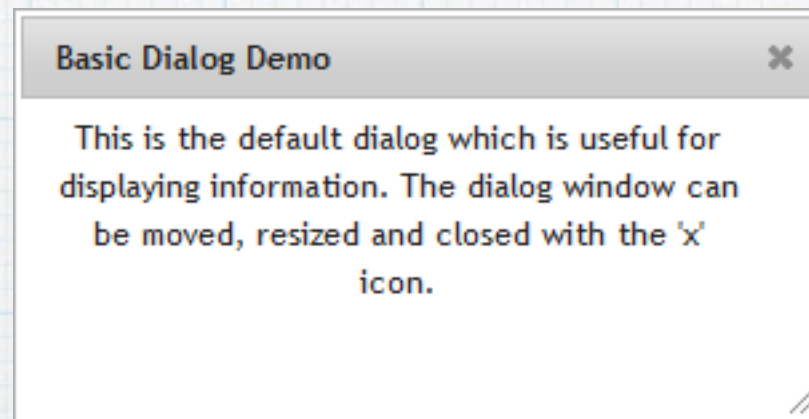
Trackbacks are a way to notify legacy blog systems that you've linked to them. If you link other blogs, [pingbacks](#), no other action necessary.

Add new custom field:

Name
<div></div>

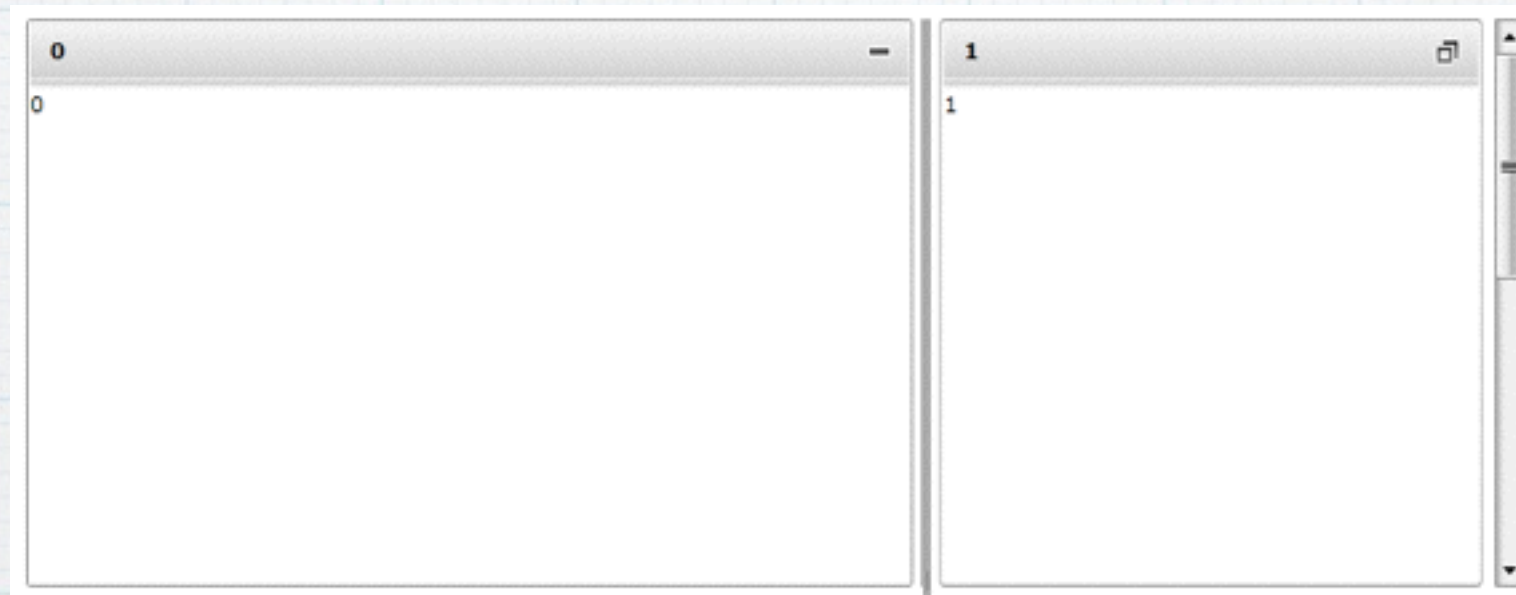


# 拖移(DragAndMove)



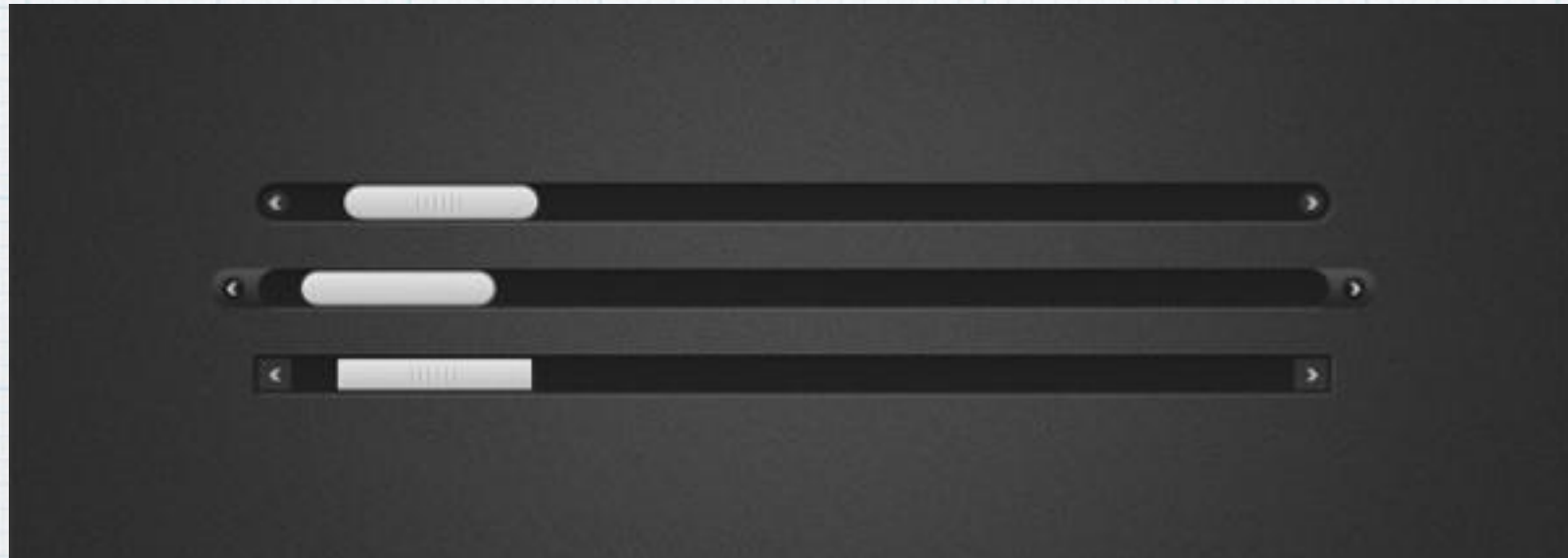


# 拖移(DragAndMove)



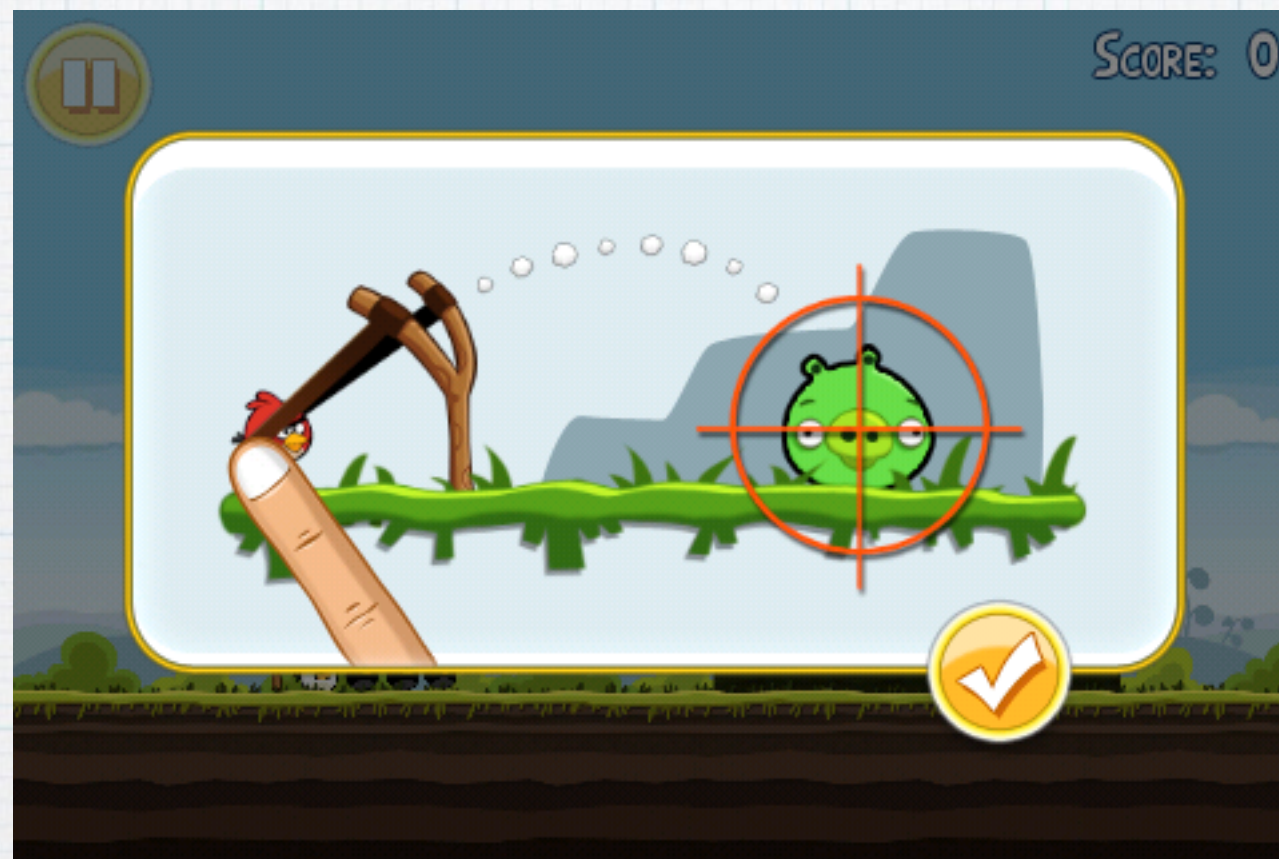


# 拖移(DragAndMove)





# 拖移(DragAndMove)





# 拖放(DragAndDrop)

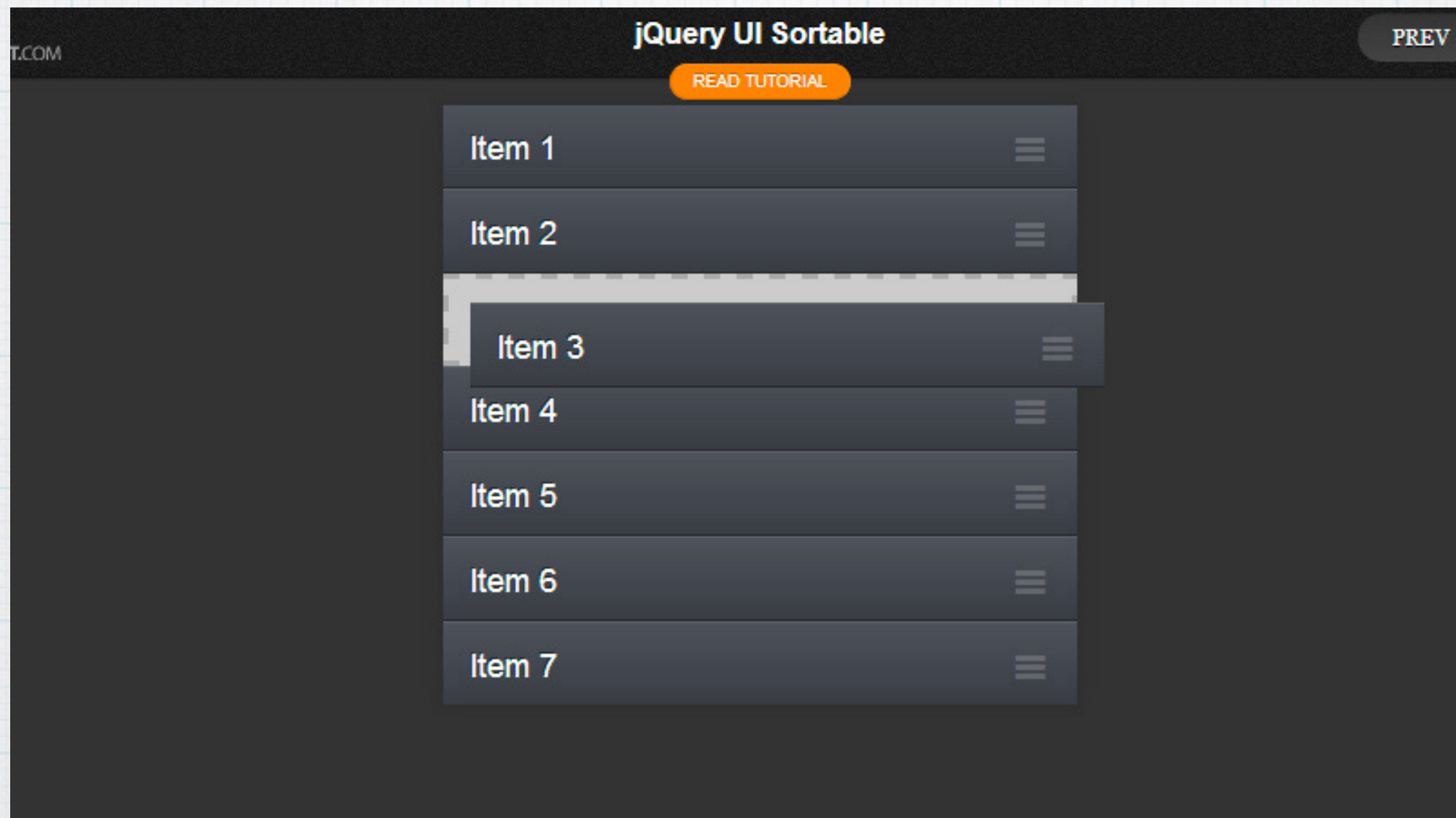


# 拖放(DragAndDrop)



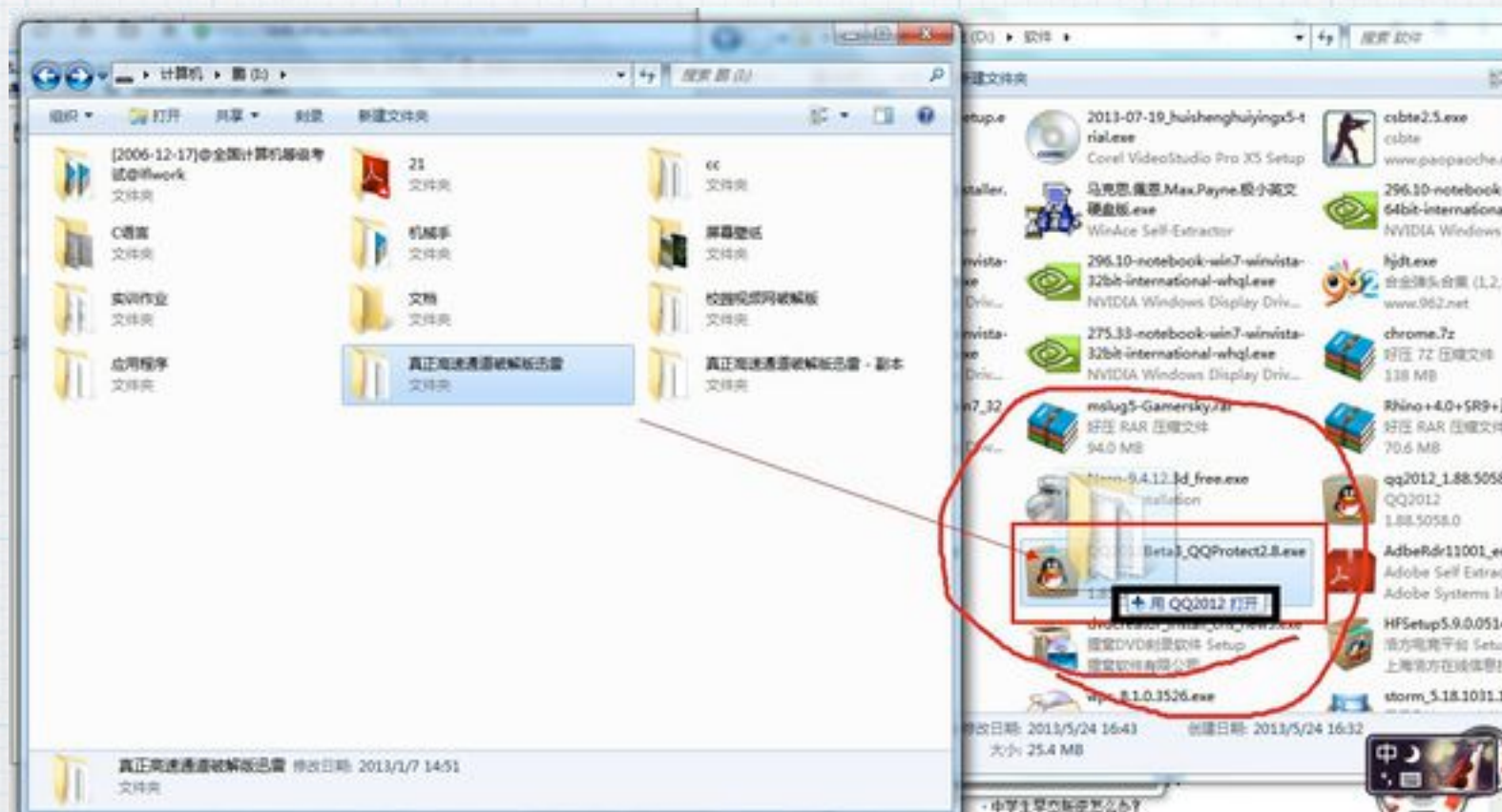


# 拖放 (DragAndDrop)





# 拖放(DragAndDrop)



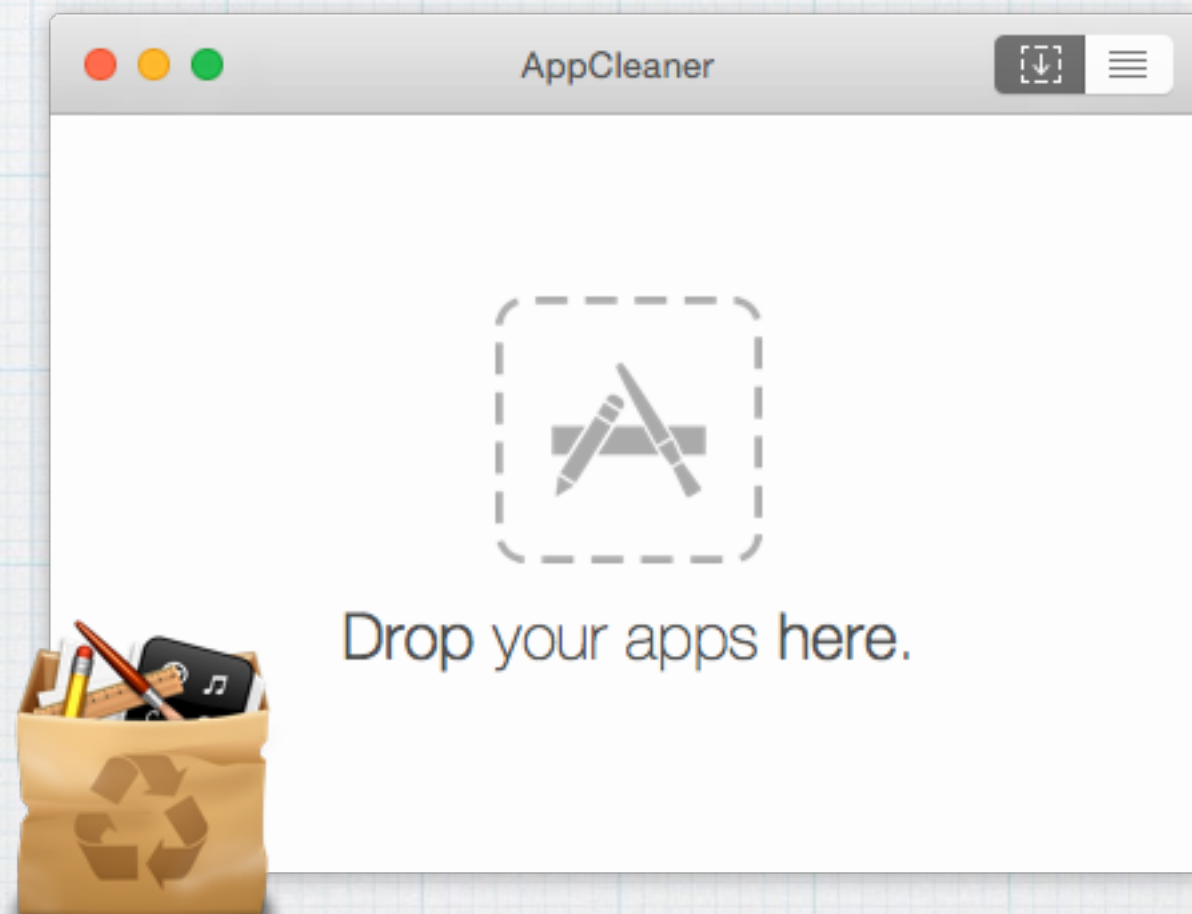


# 拖放(DragAndDrop)

→ **Drop files** to upload  
(or click)



# 拖放(DragAndDrop)





# 常见的拖移库

- ActionScript - startDrag(), stopDrag()
- jQuery UI - Draggable
- jQuery UI - Resizable
- Draggabilly
- interact.js



# 常见的拖放库

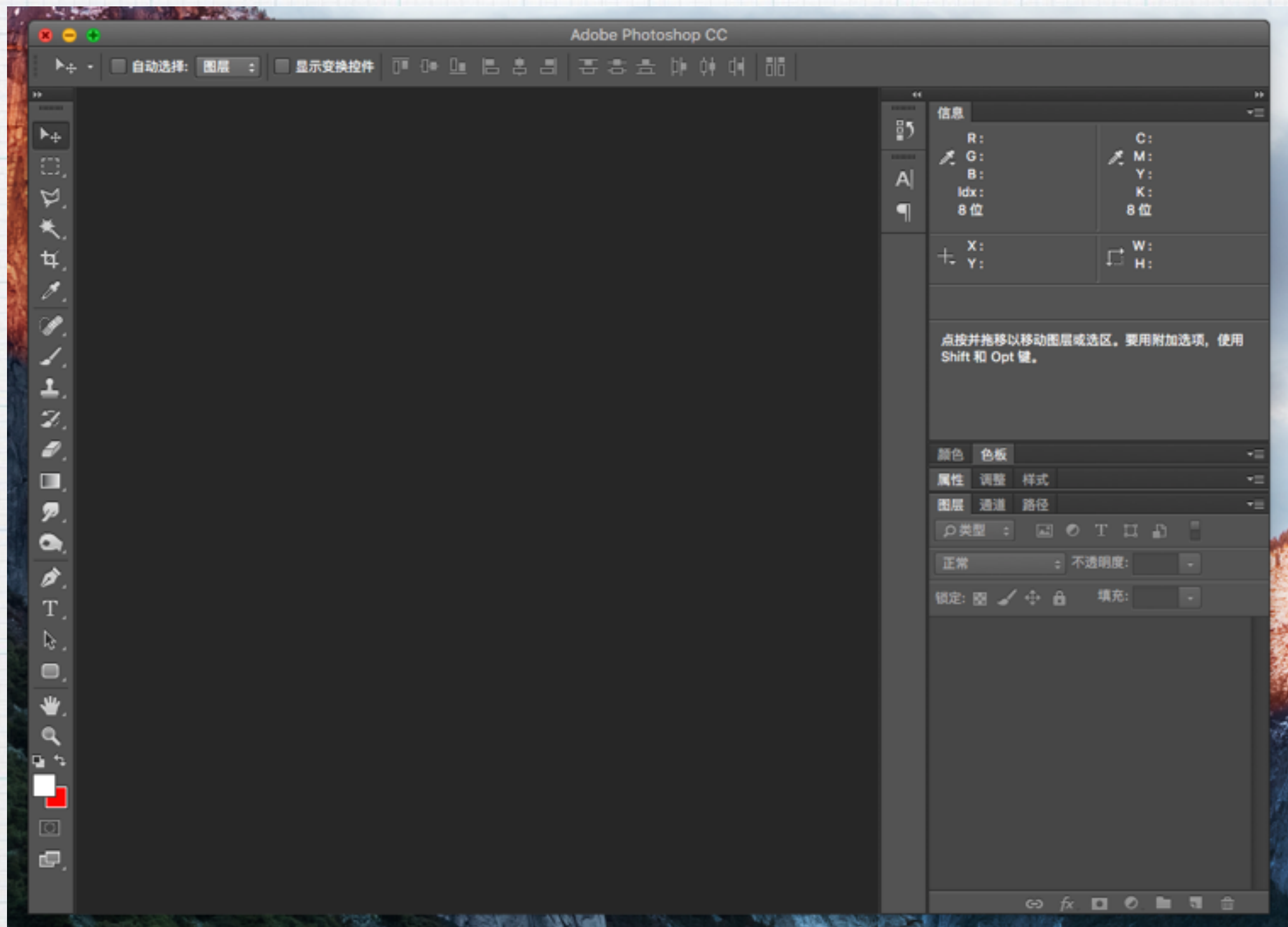
- ◎ HTML5 - DragAndDrop
- ◎ Android - DragAndDrop
- ◎ WinForm - DragAndDrop
- ◎ jQuery UI - Sortable
- ◎ Dragula
- ◎ HTML5Sortable
- ◎ Sortable.js



# 两者的区别

拖移	拖放
专注于位置操作	专注于数据操作
拖拽时约束位置，拖拽后一般不处理结果	拖拽时位置自由，拖拽后处理结果
一般为拖拽自身	一般为拖动副本
野生库较多	官方库较多
用起来不太放心	用起来不太灵活







# 基本流程



# 拖放示例





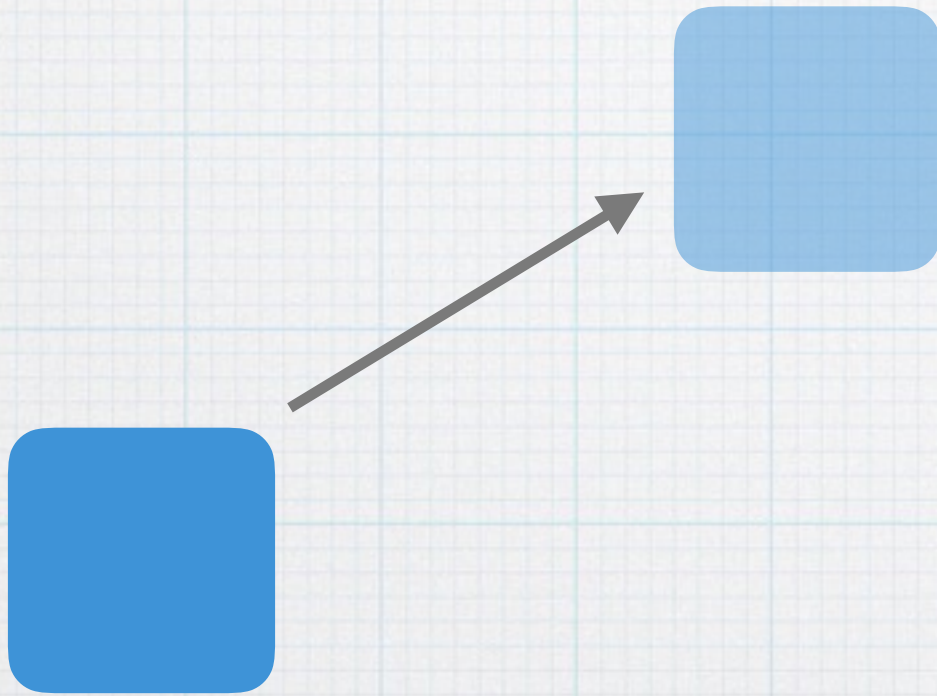
# 拖放示例



起始元素(source)



# 拖放示例

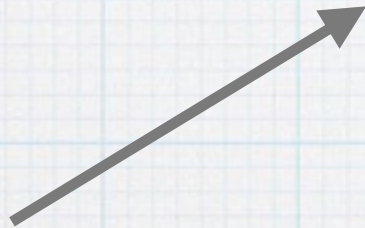
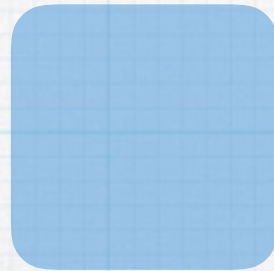


起始元素(source)



# 拖放示例

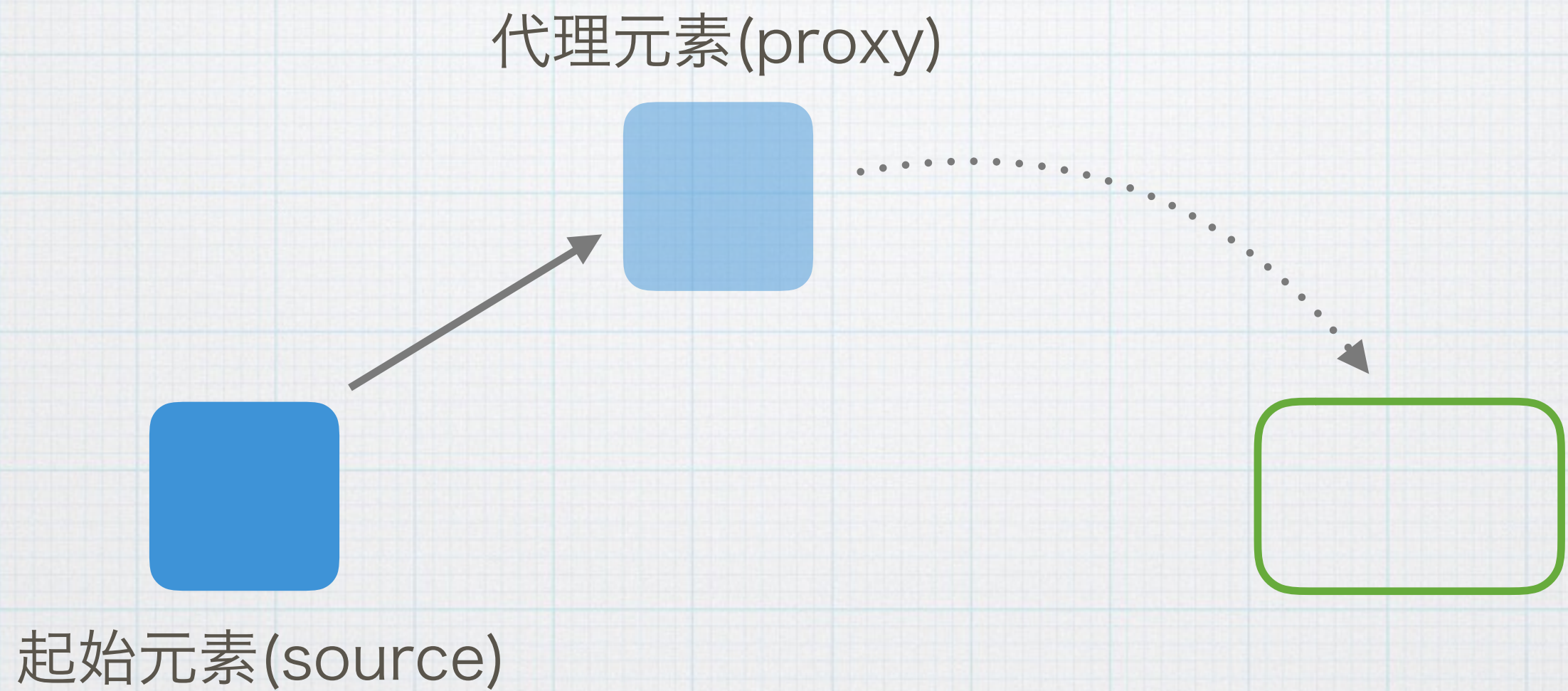
代理元素(proxy)



起始元素(source)

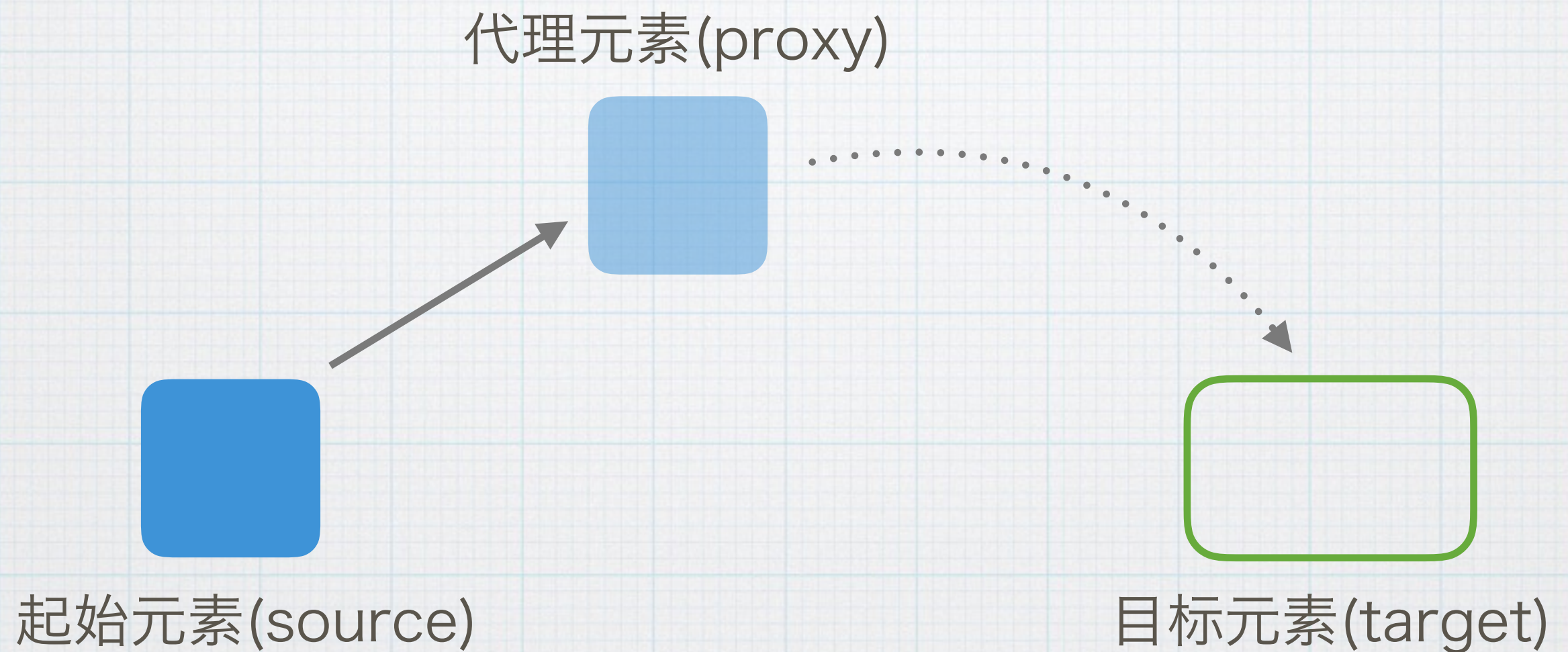


# 拖放示例





# 拖放示例





# 三种元素



# 三种元素

- 起始元素(source): 当点击并开始移动鼠标时, 能够触发拖拽事件的元素, 又叫draggableElement



# 三种元素

- 起始元素(source): 当点击并开始移动鼠标时, 能够触发拖拽事件的元素, 又叫draggableElement
- 代理元素(proxy): 拖拽时跟随鼠标移动的元素, 又叫dragImage、dragShadow、helper



# 三种元素

- 起始元素(source): 当点击并开始移动鼠标时, 能够触发拖拽事件的元素, 又叫draggableElement
- 代理元素(proxy): 拖拽时跟随鼠标移动的元素, 又叫dragImage、dragShadow、helper
- 目标元素(target): 当有元素在上方拖拽时, 能够触发拖放事件的元素, 又叫droppableElement, dropzone



# 拖放事件（起始元素）





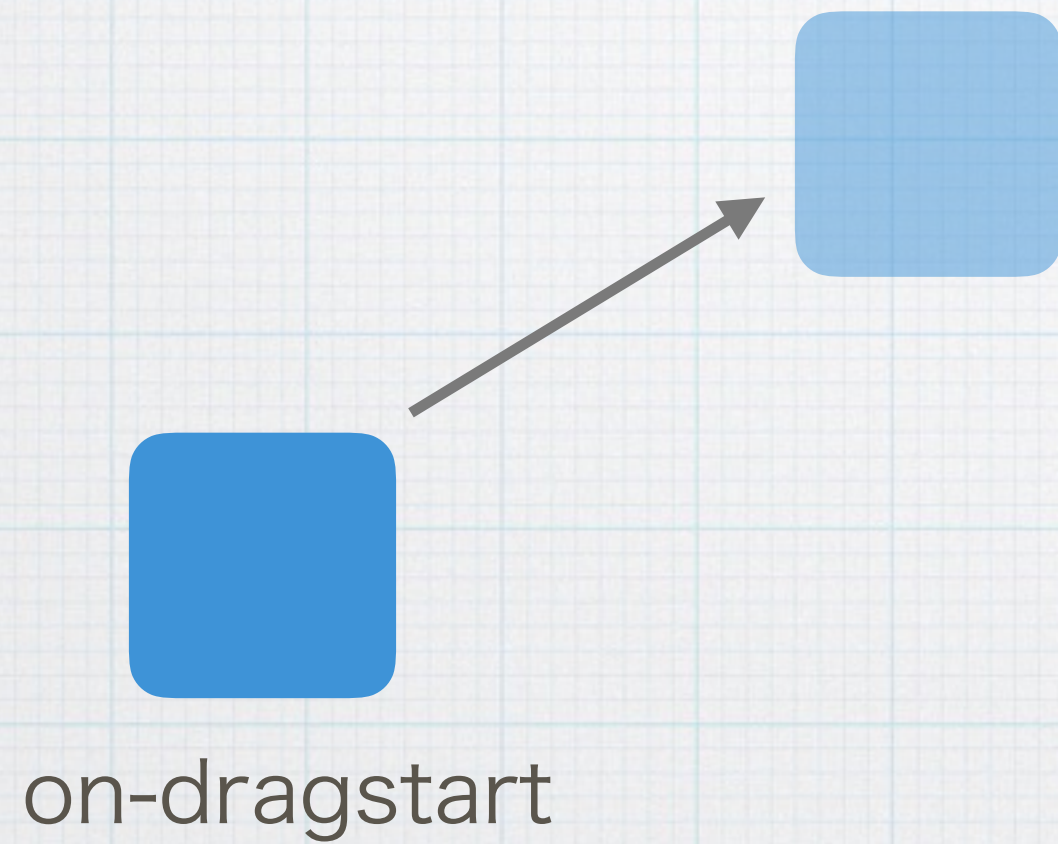
# 拖放事件（起始元素）



on-dragstart

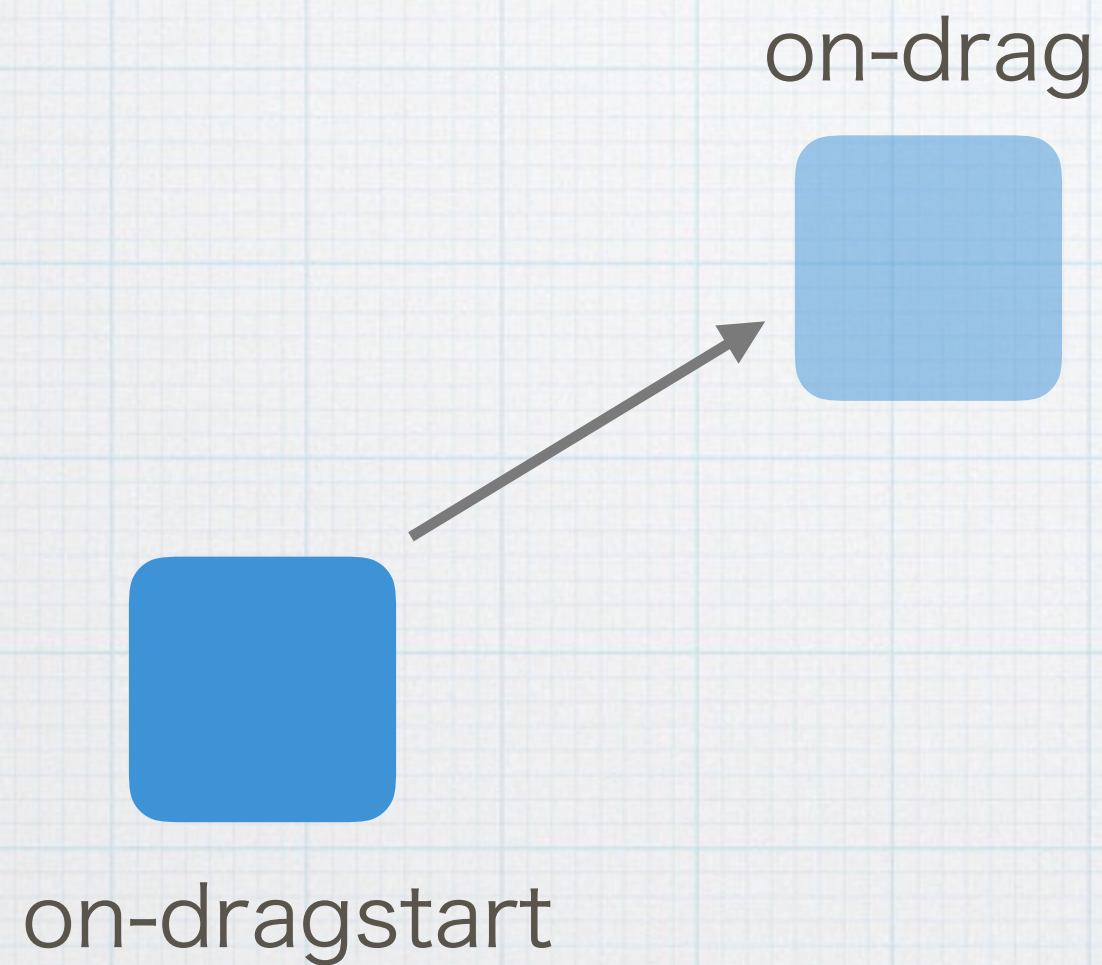


# 拖放事件（起始元素）



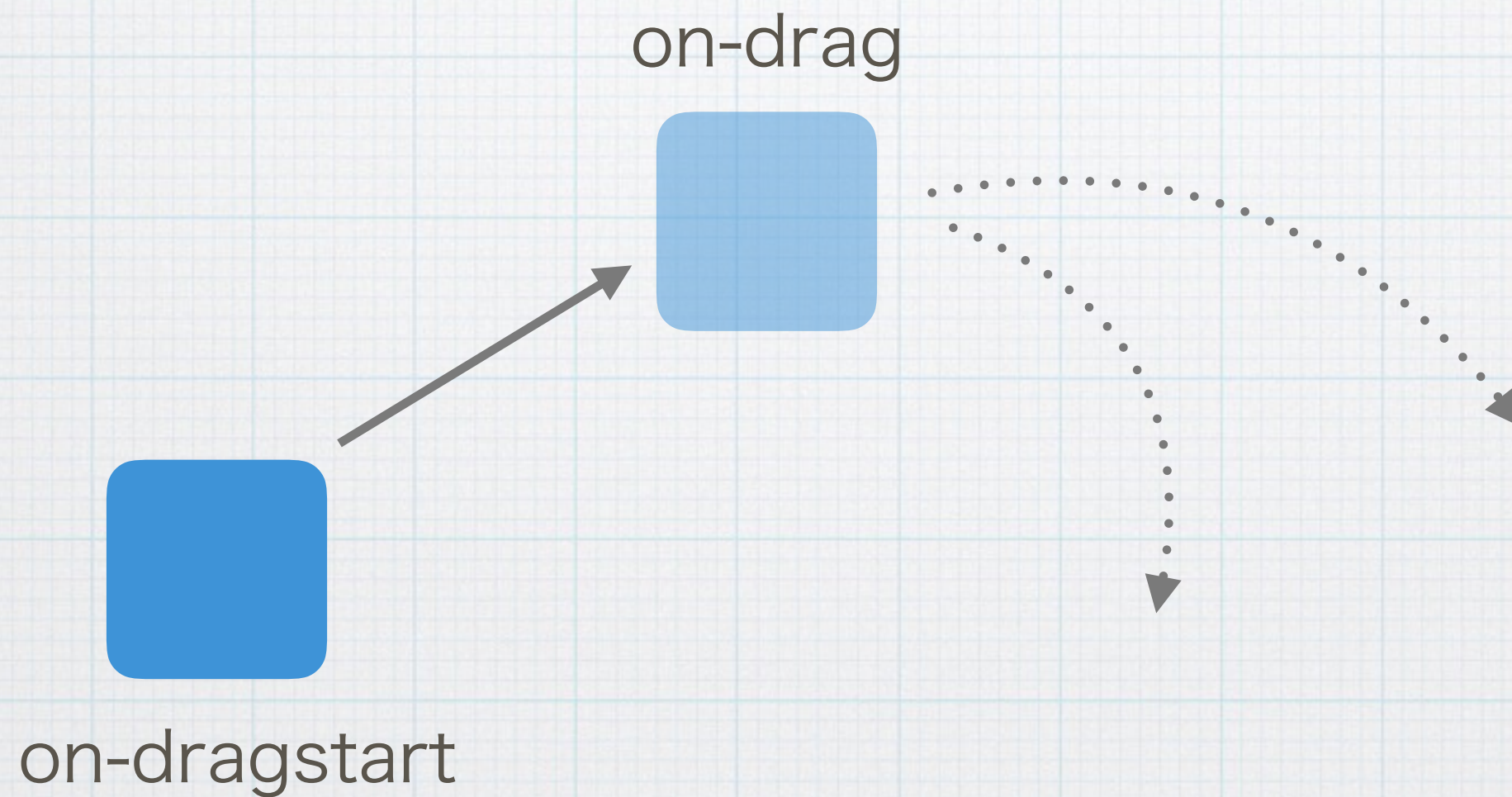


# 拖放事件（起始元素）



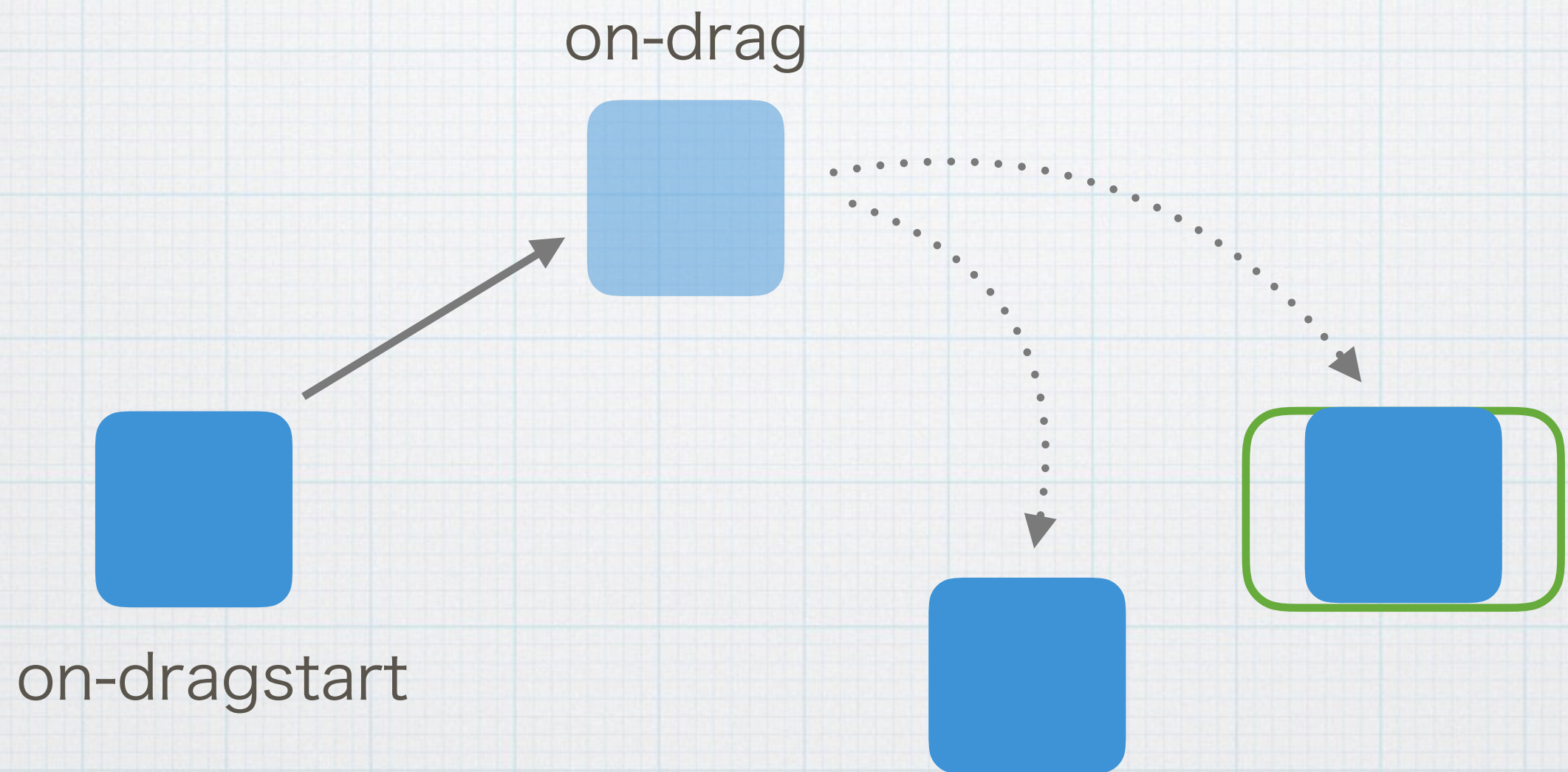


# 拖放事件（起始元素）



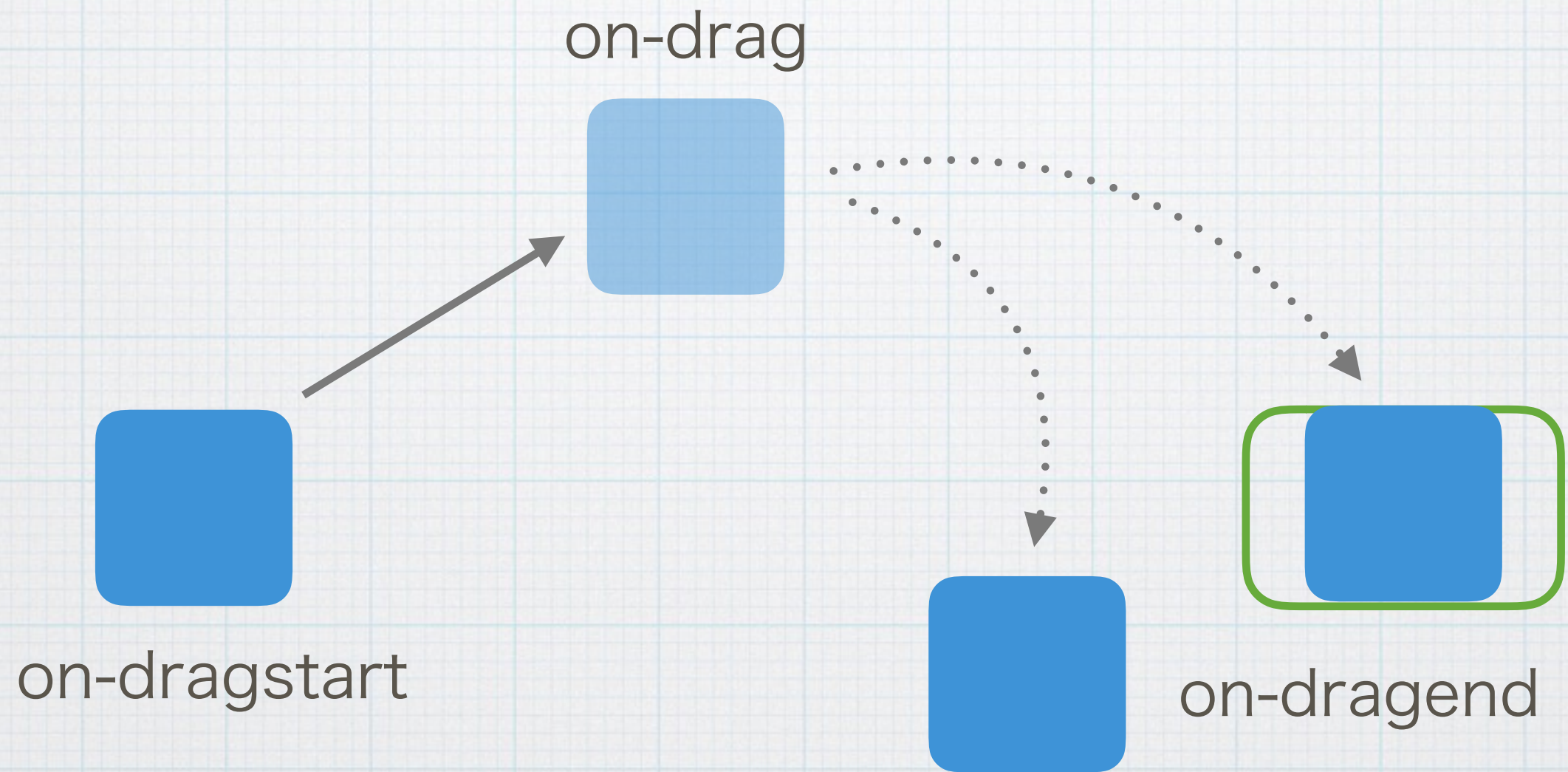


# 拖放事件（起始元素）



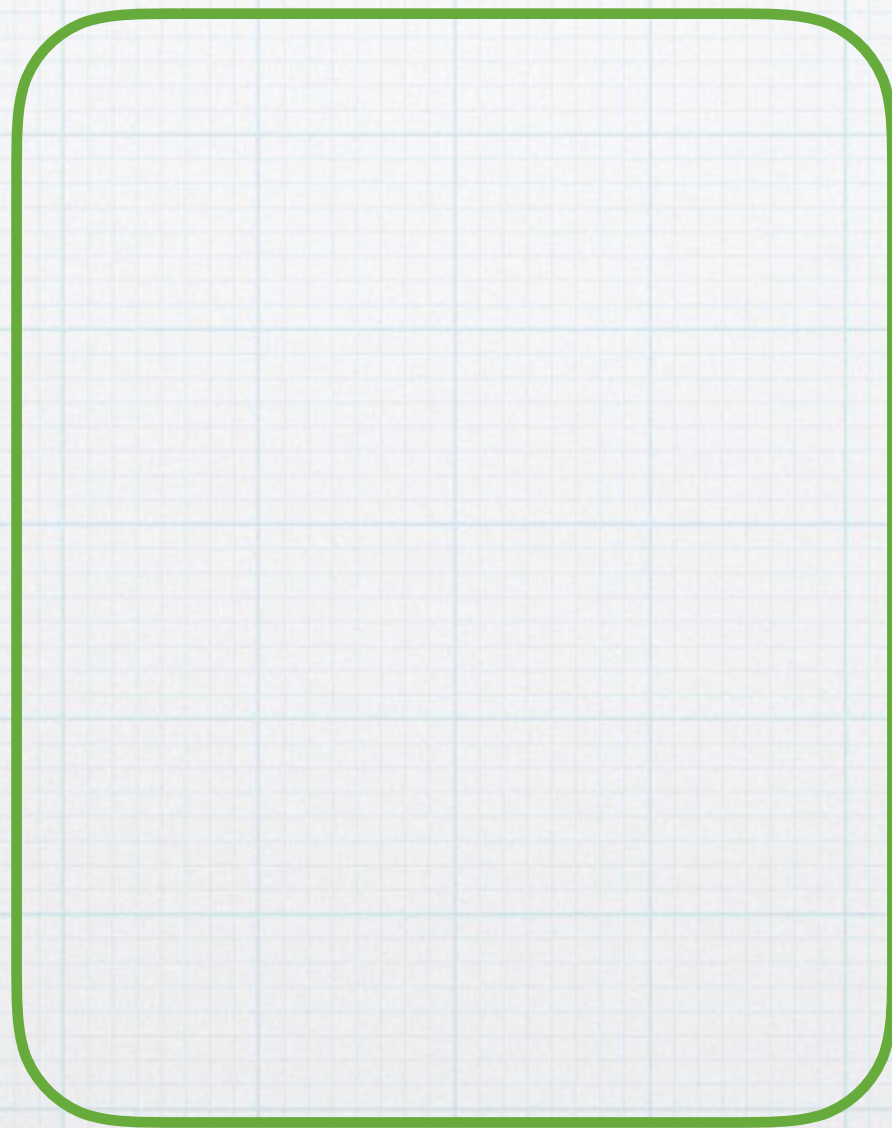


# 拖放事件（起始元素）



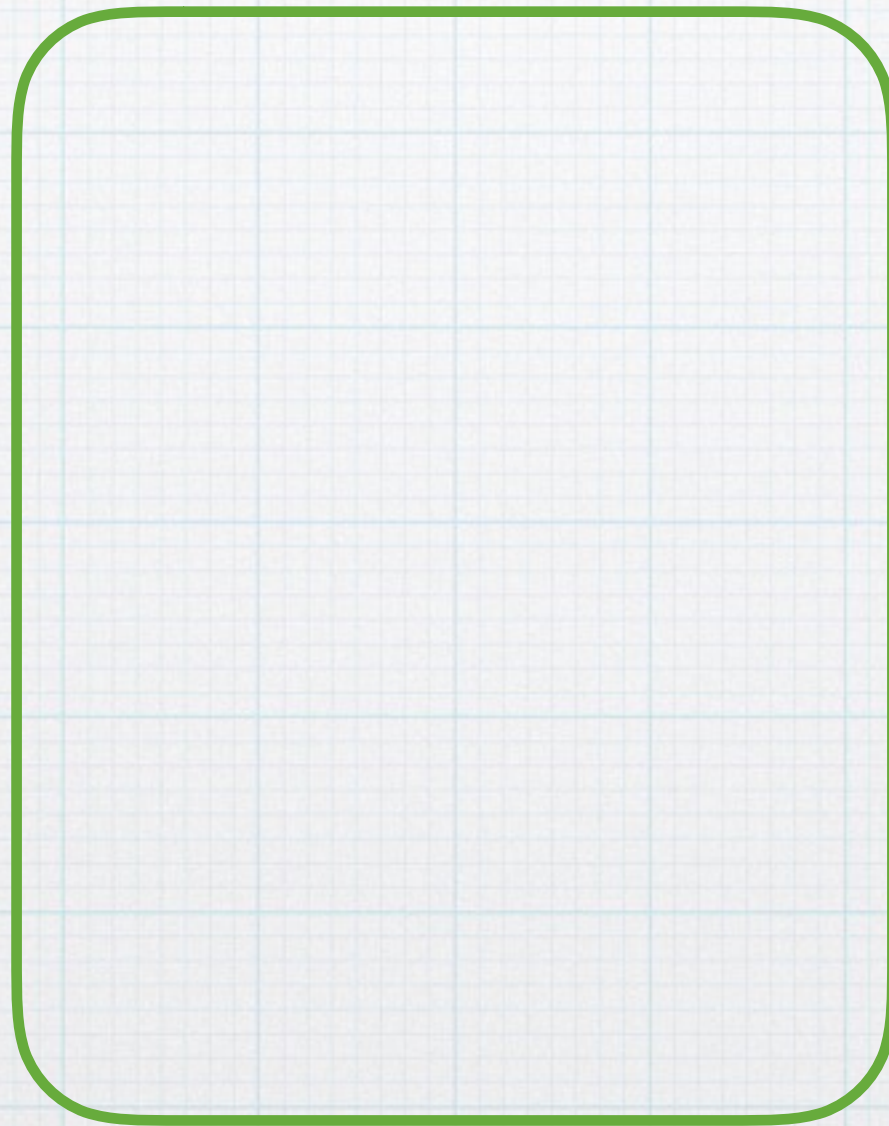


# 拖放事件（目标元素）



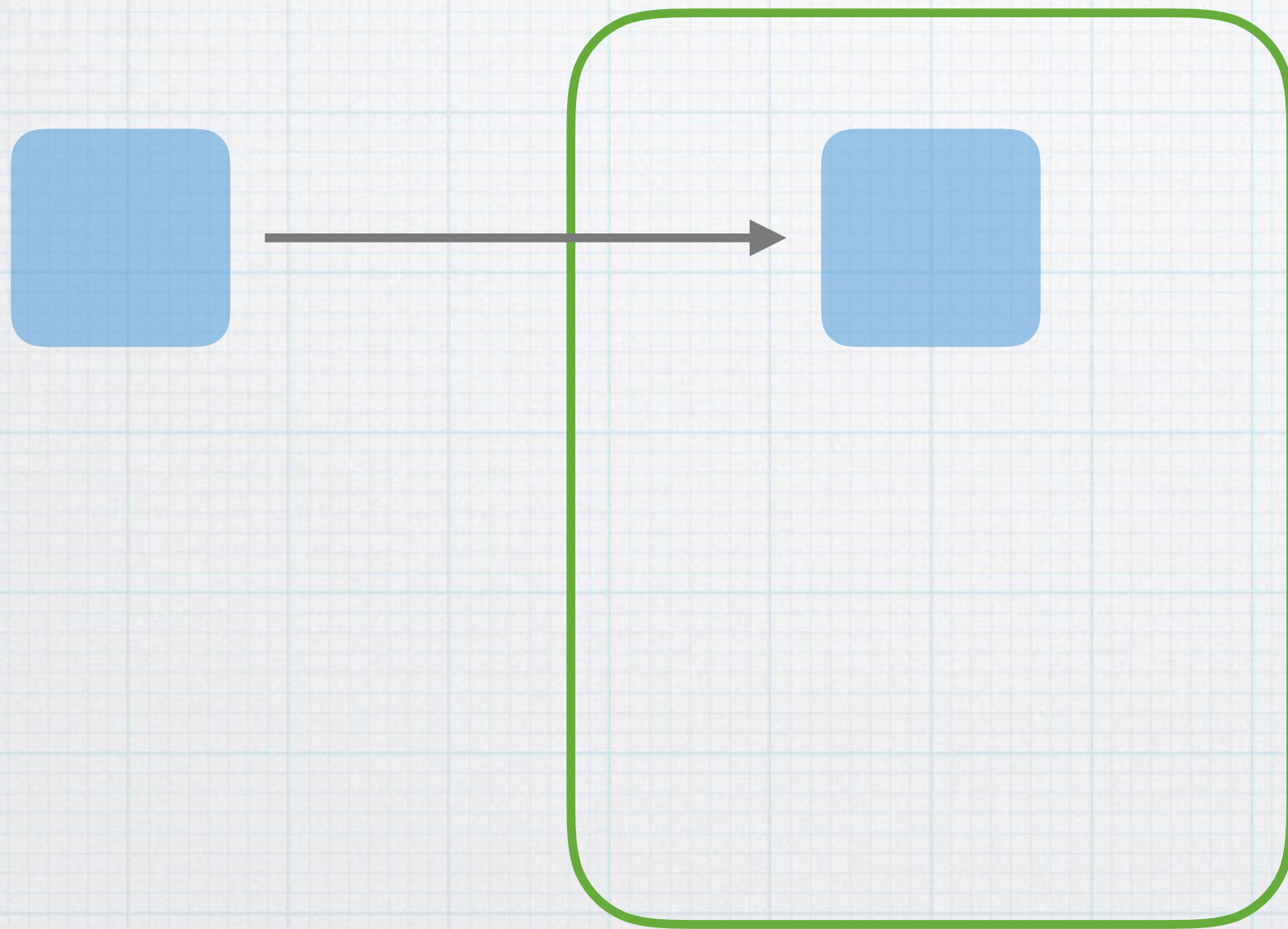


# 拖放事件（目标元素）



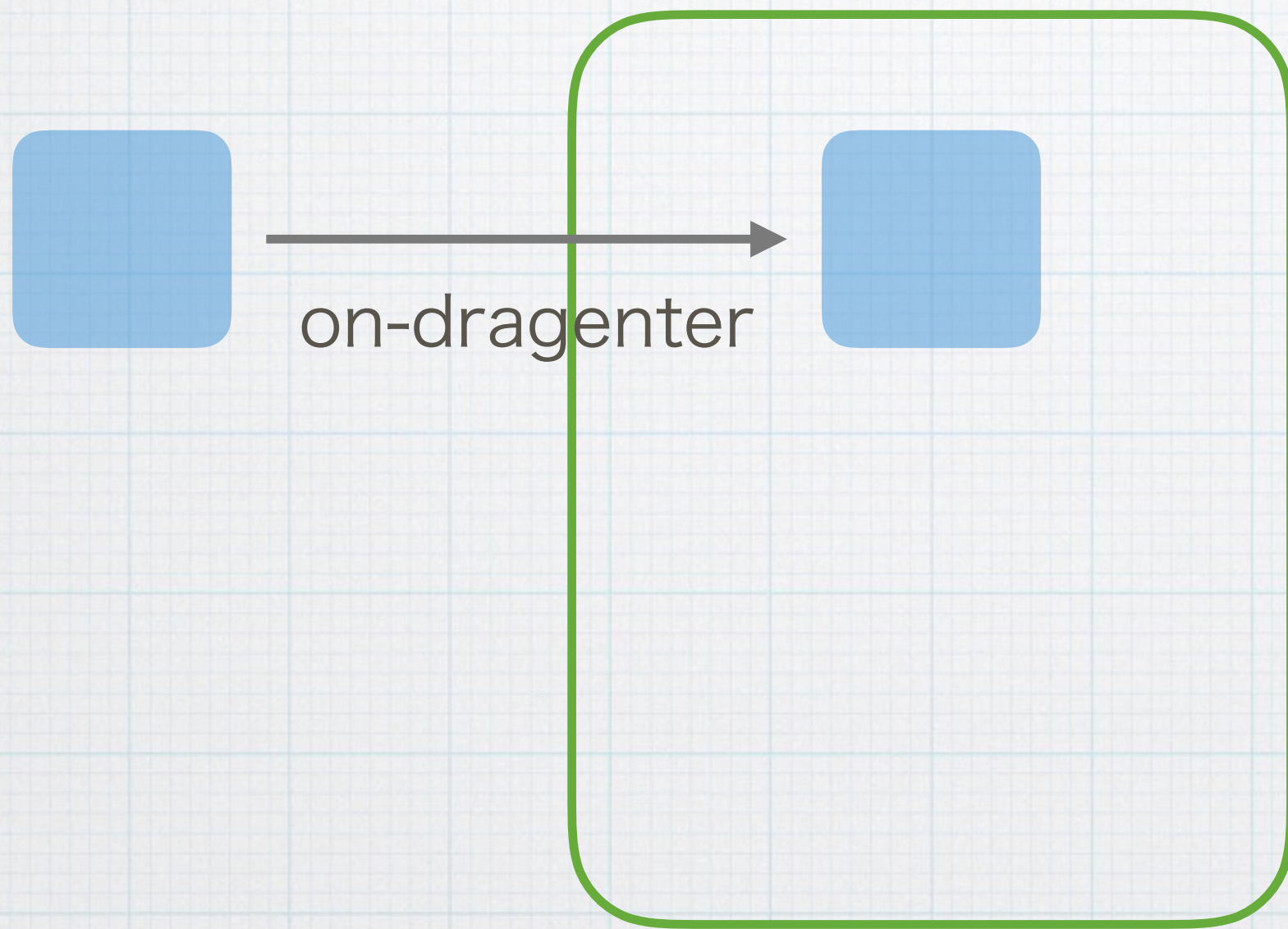


# 拖放事件（目标元素）



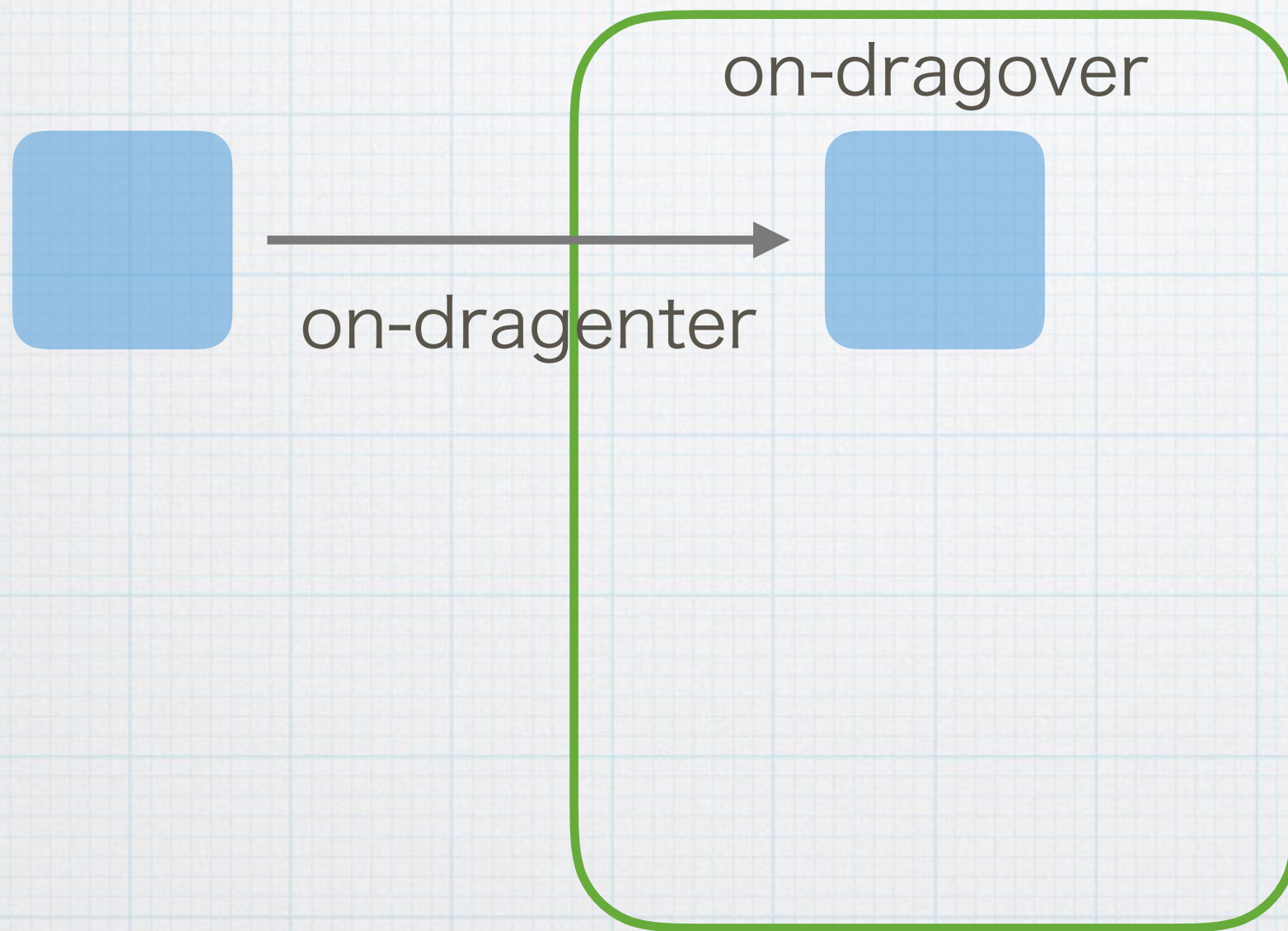


# 拖放事件（目标元素）



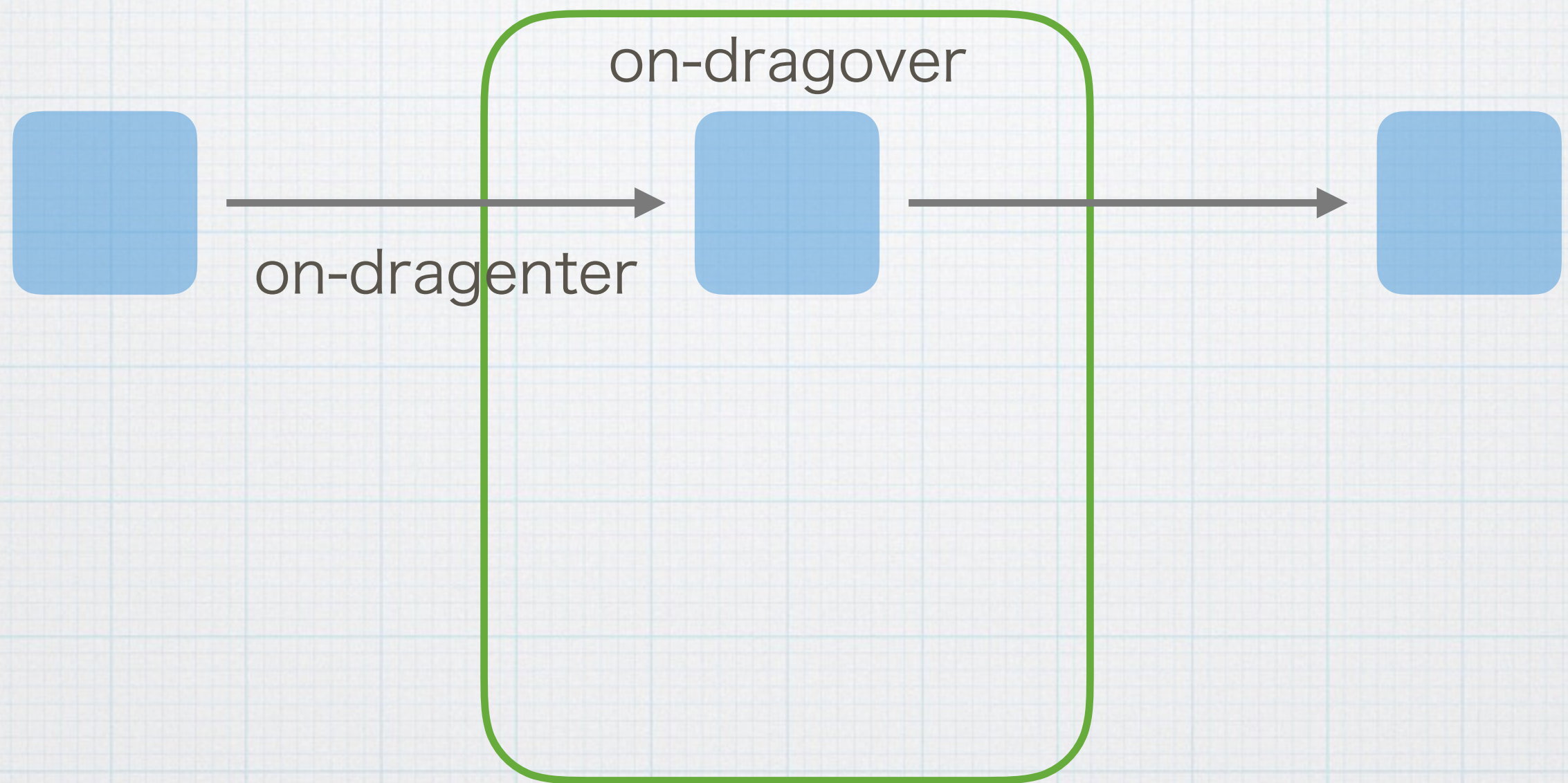


# 拖放事件（目标元素）



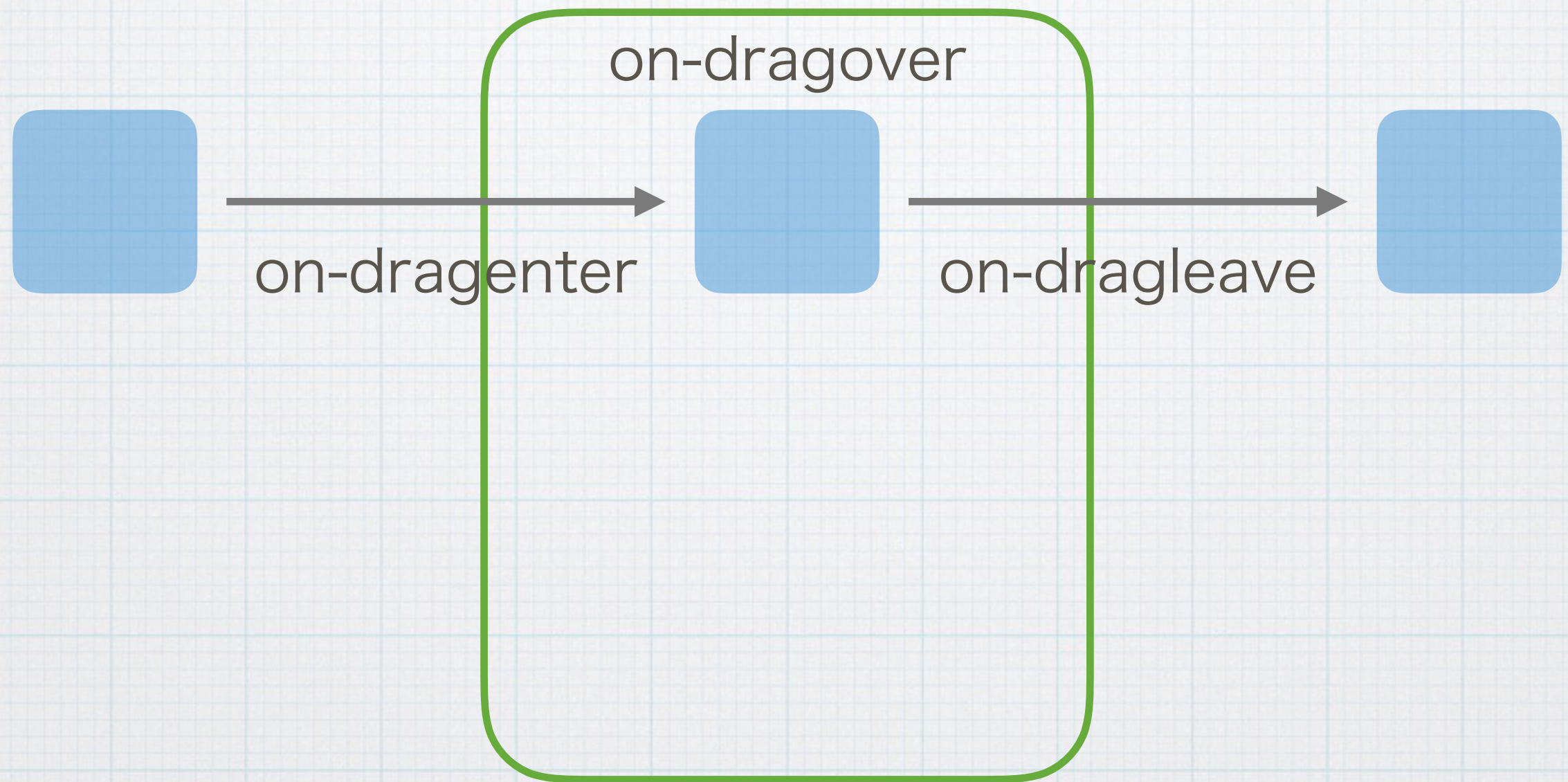


# 拖放事件（目标元素）



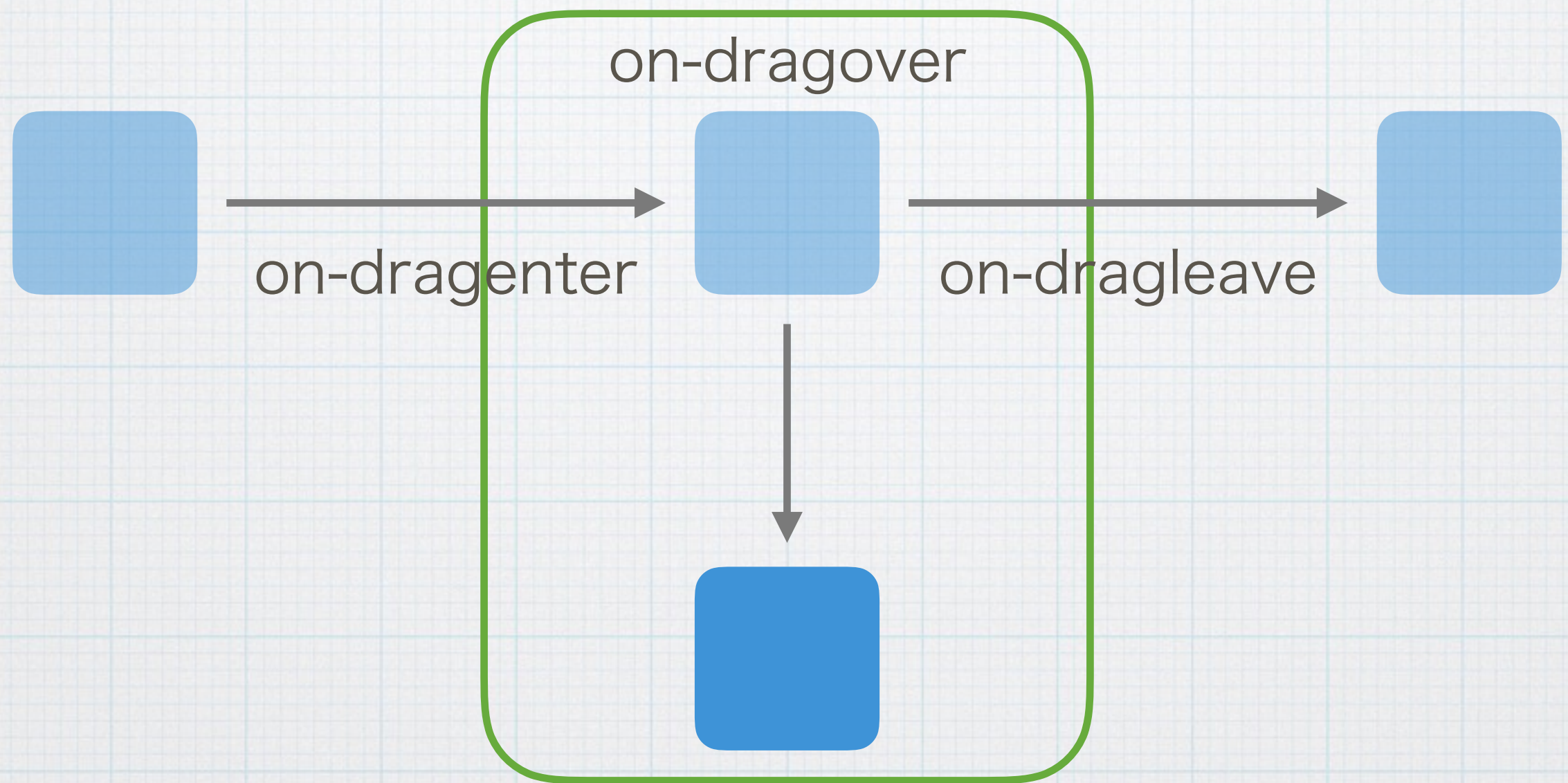


# 拖放事件（目标元素）



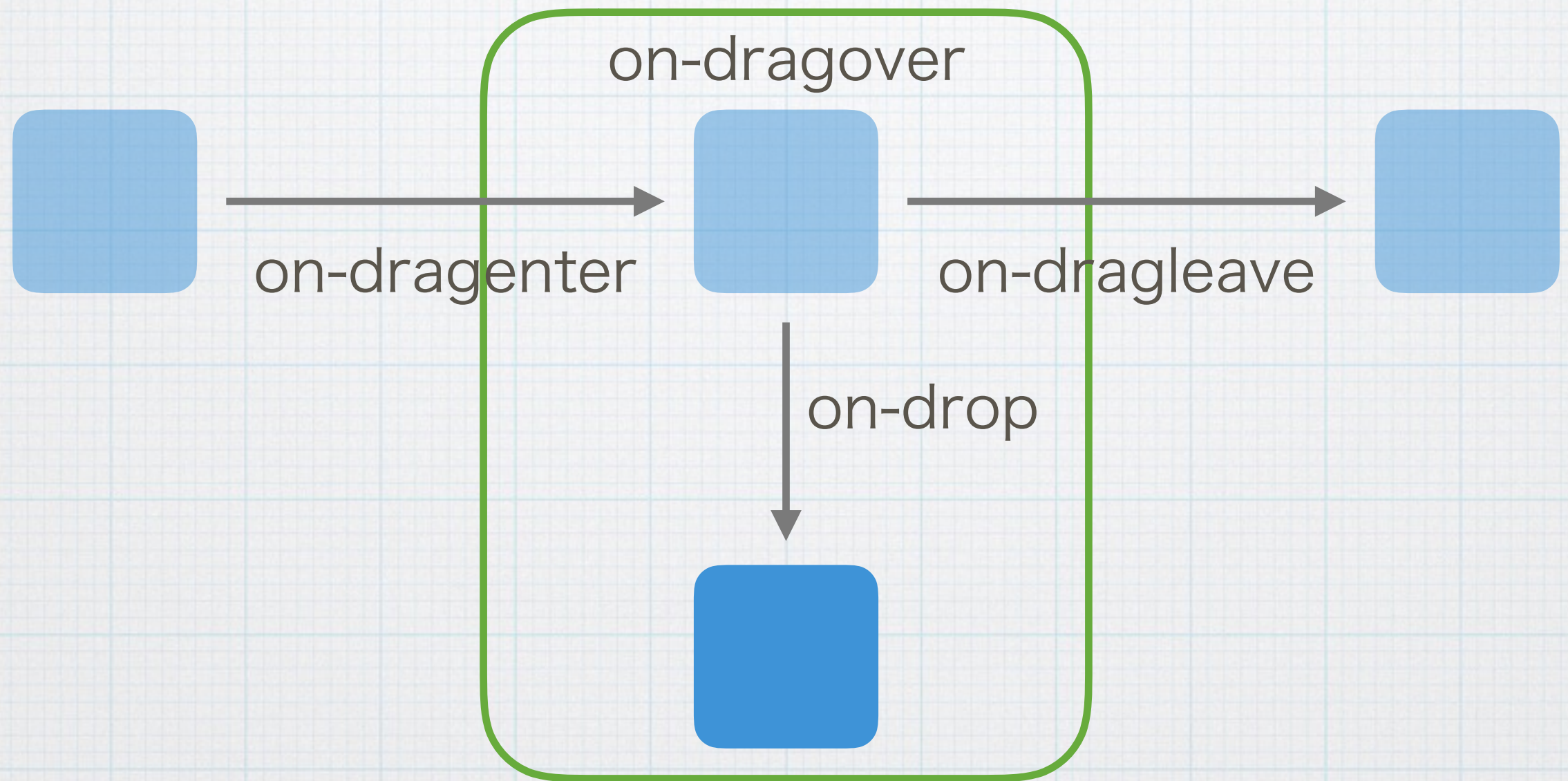


# 拖放事件（目标元素）





# 拖放事件（目标元素）





# 拖移示例





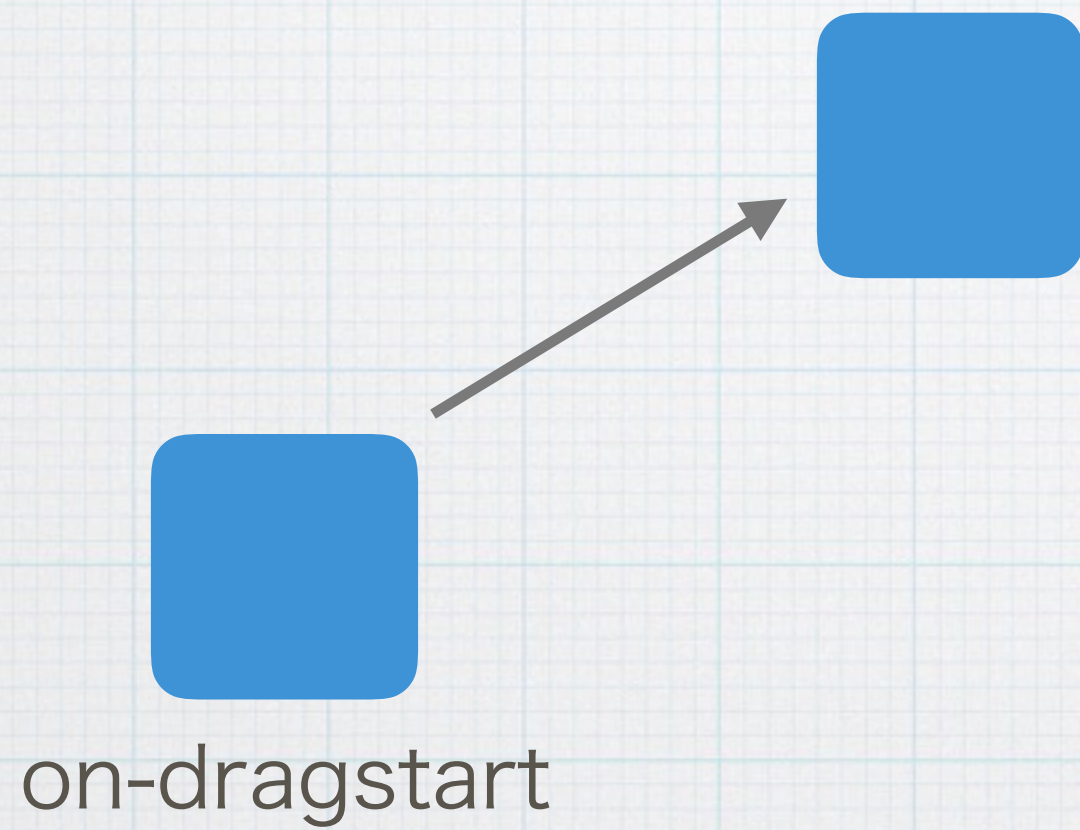
# 拖移示例



on-dragstart

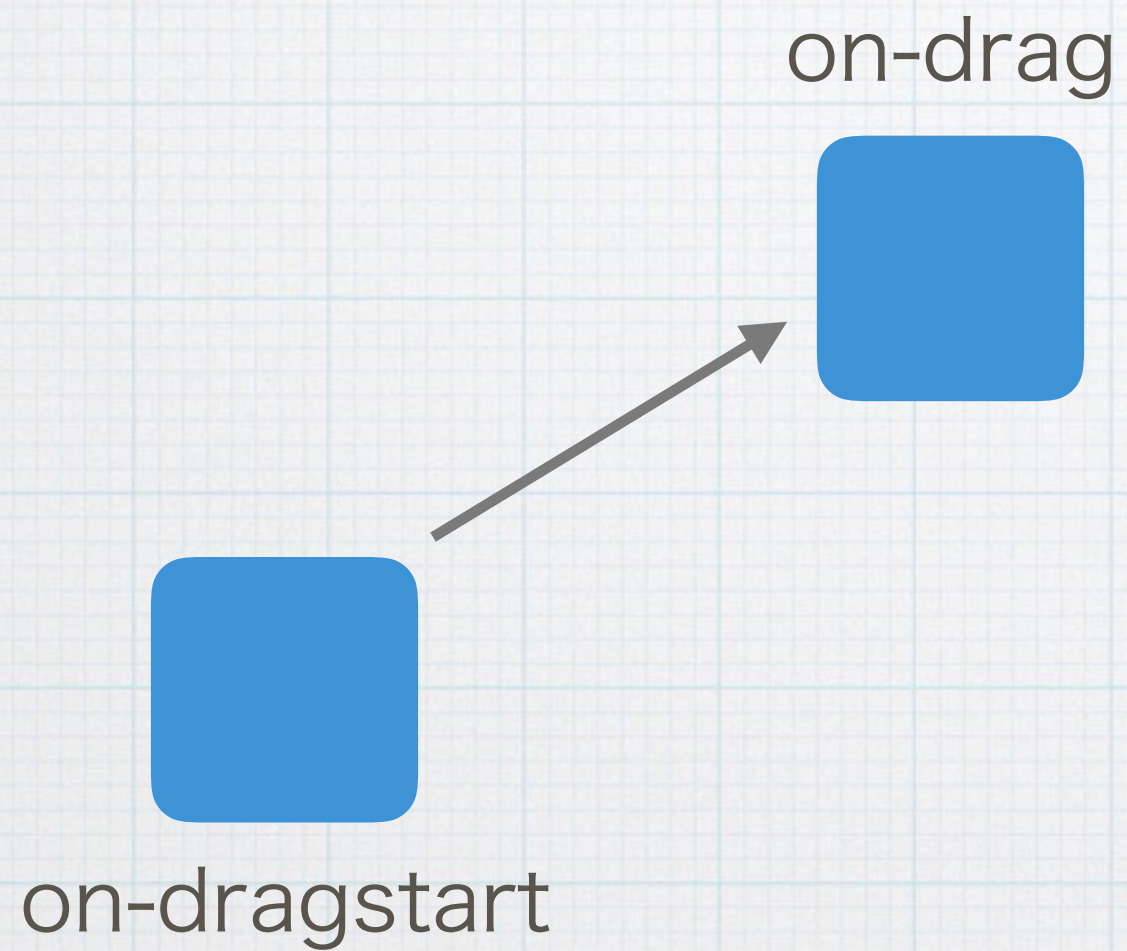


# 拖移示例



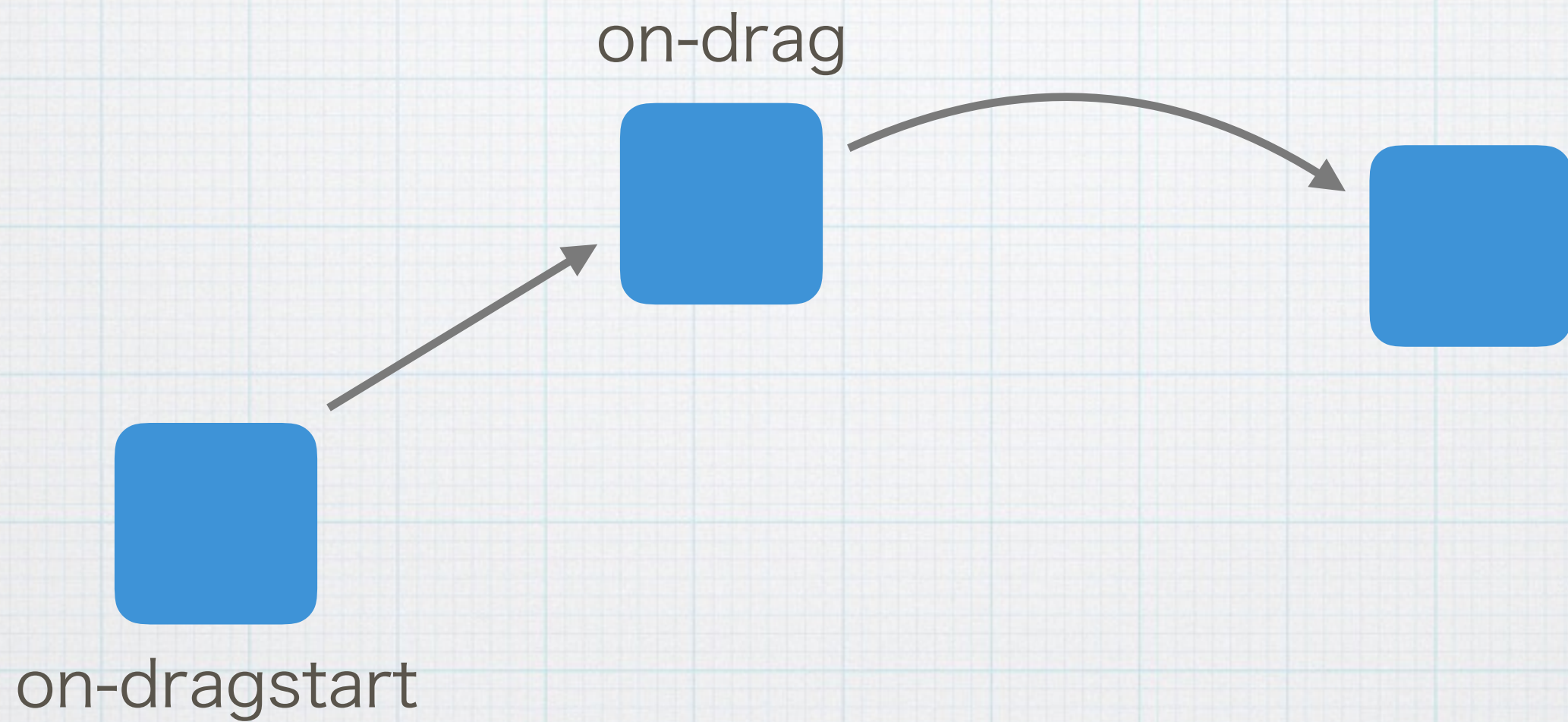


# 拖移示例



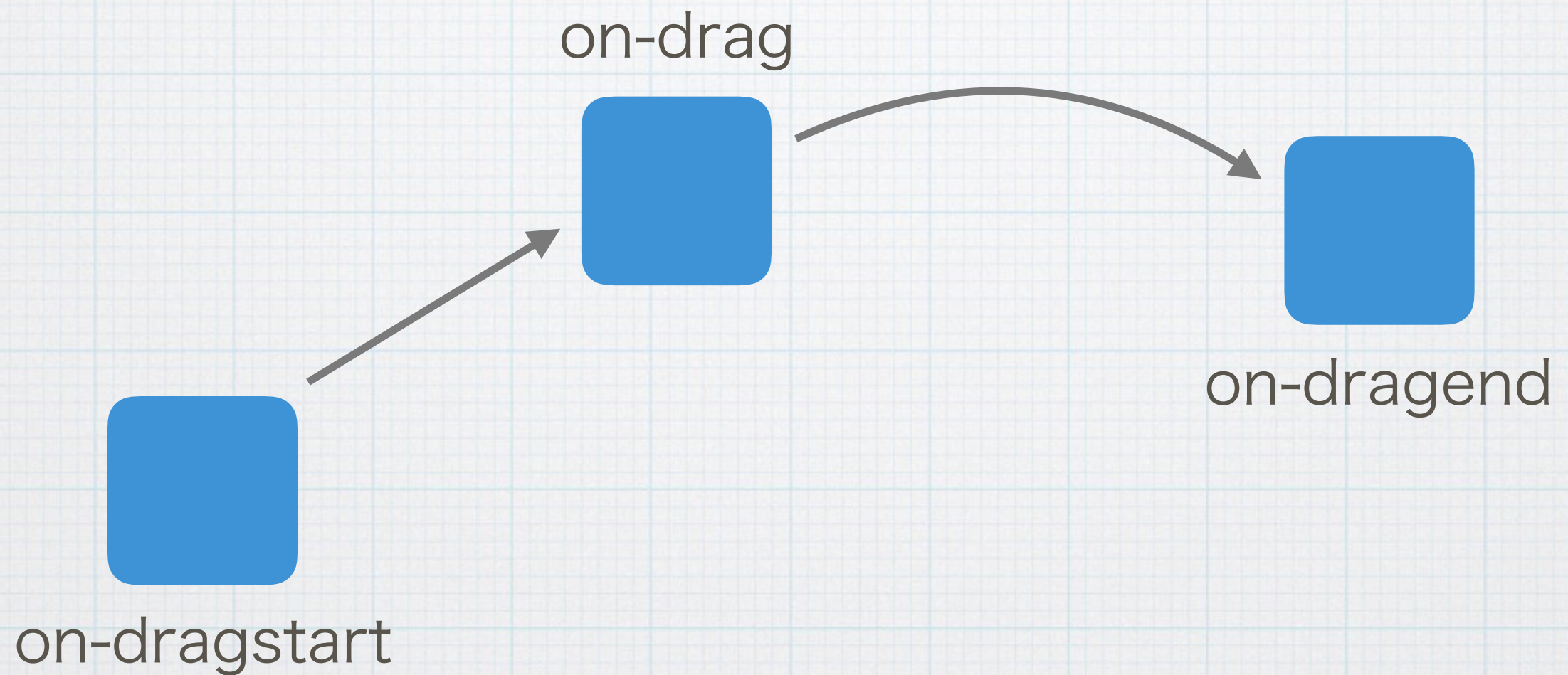


# 拖移示例



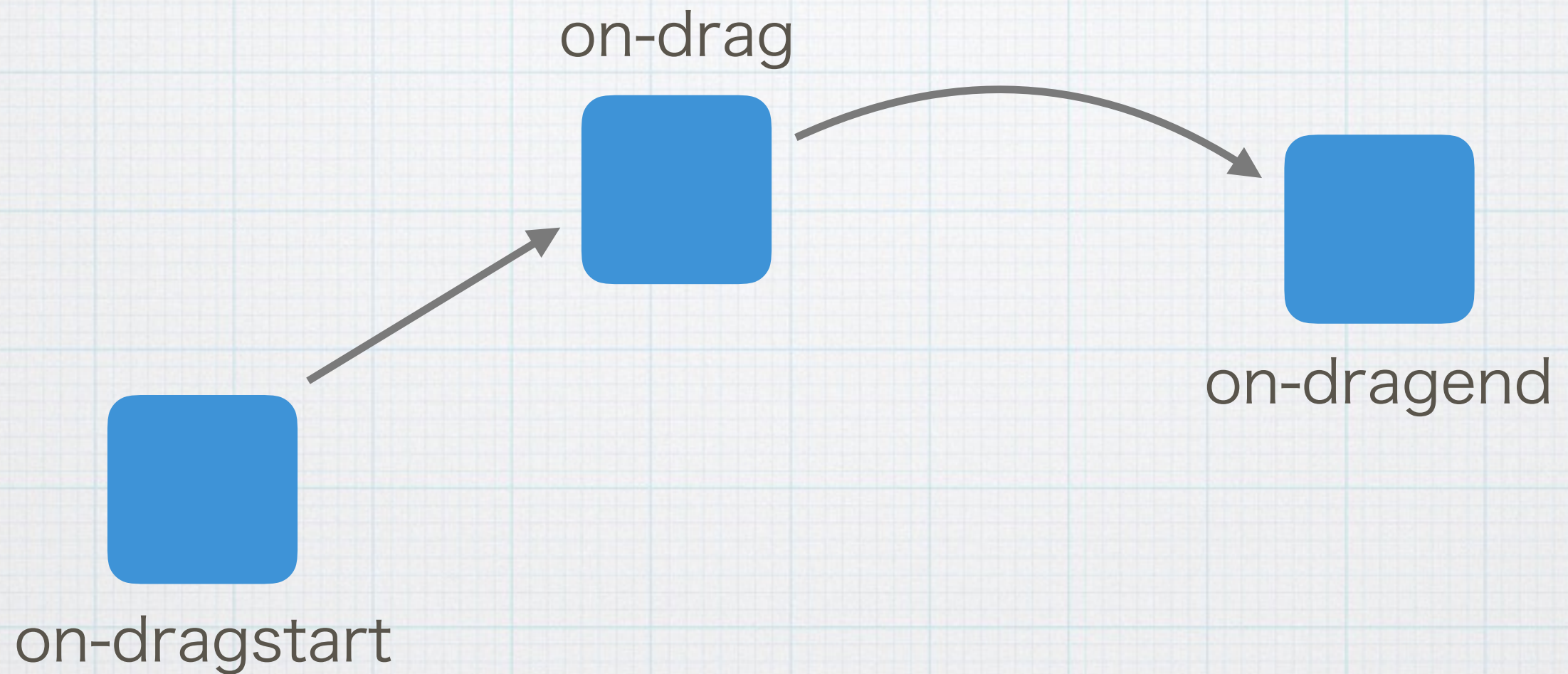


# 拖移示例





# 拖移示例



代理元素(proxy) === 起始元素(source)



# 拖移示例





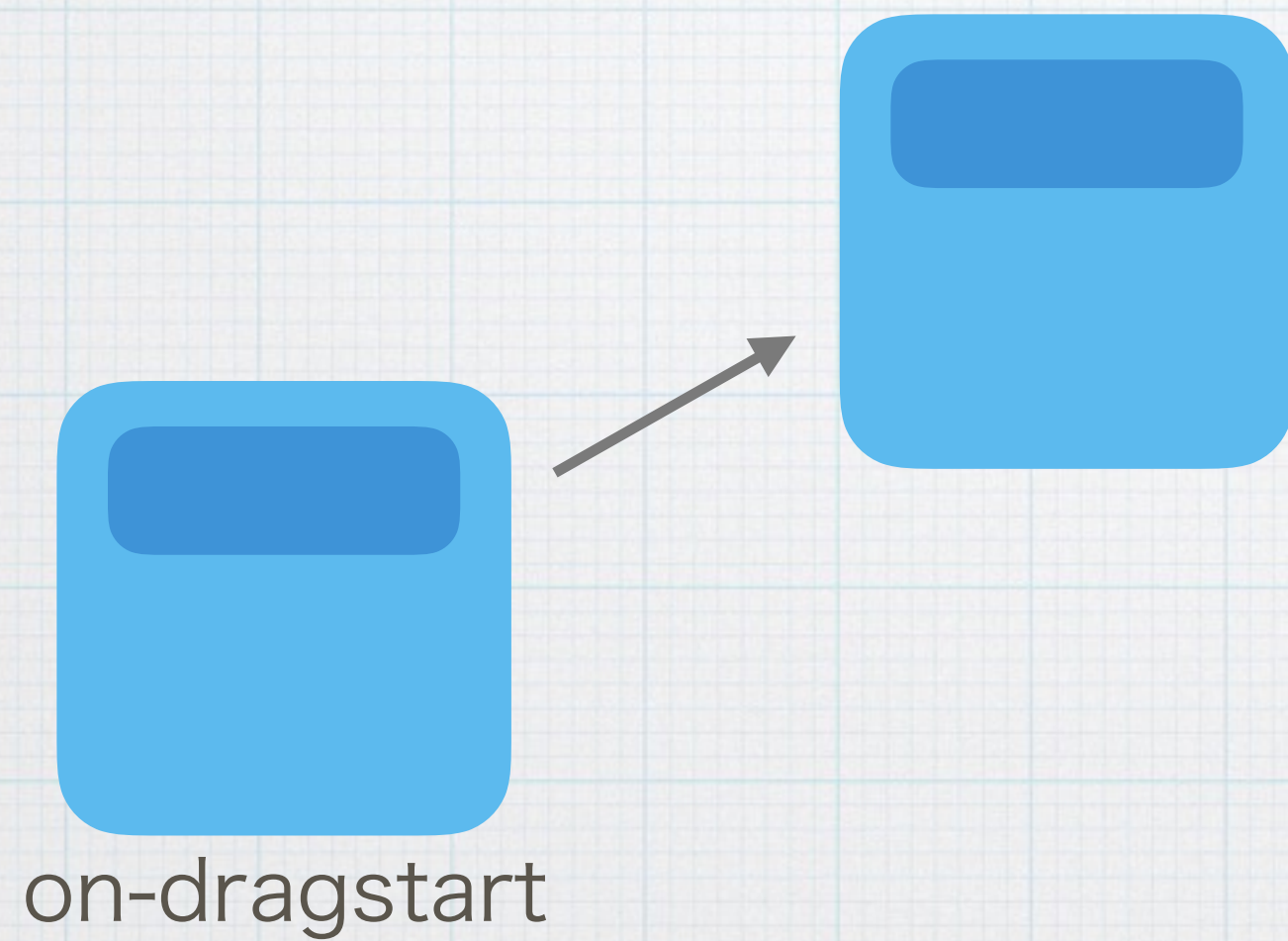
# 拖移示例



on-dragstart

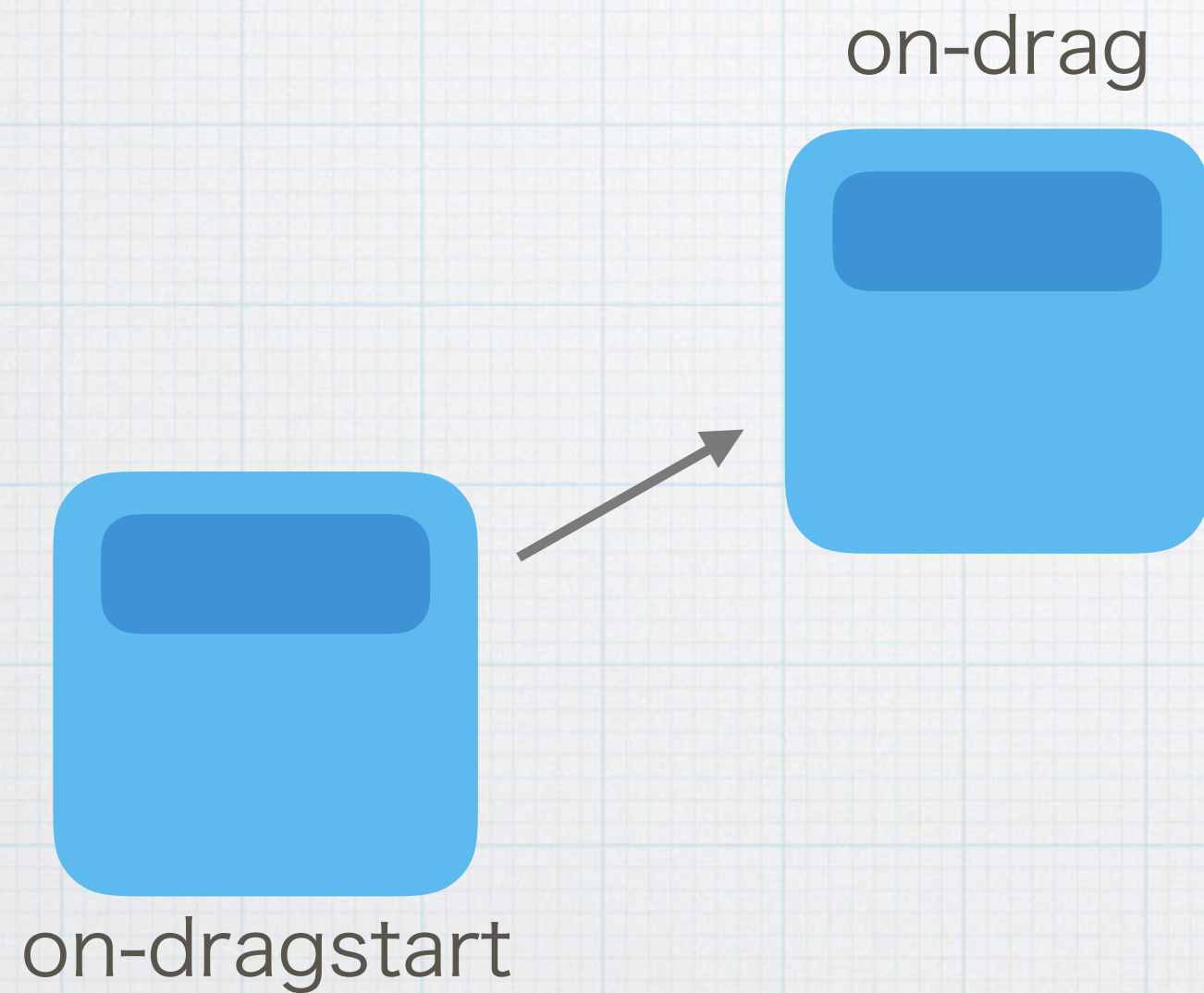


# 拖移示例



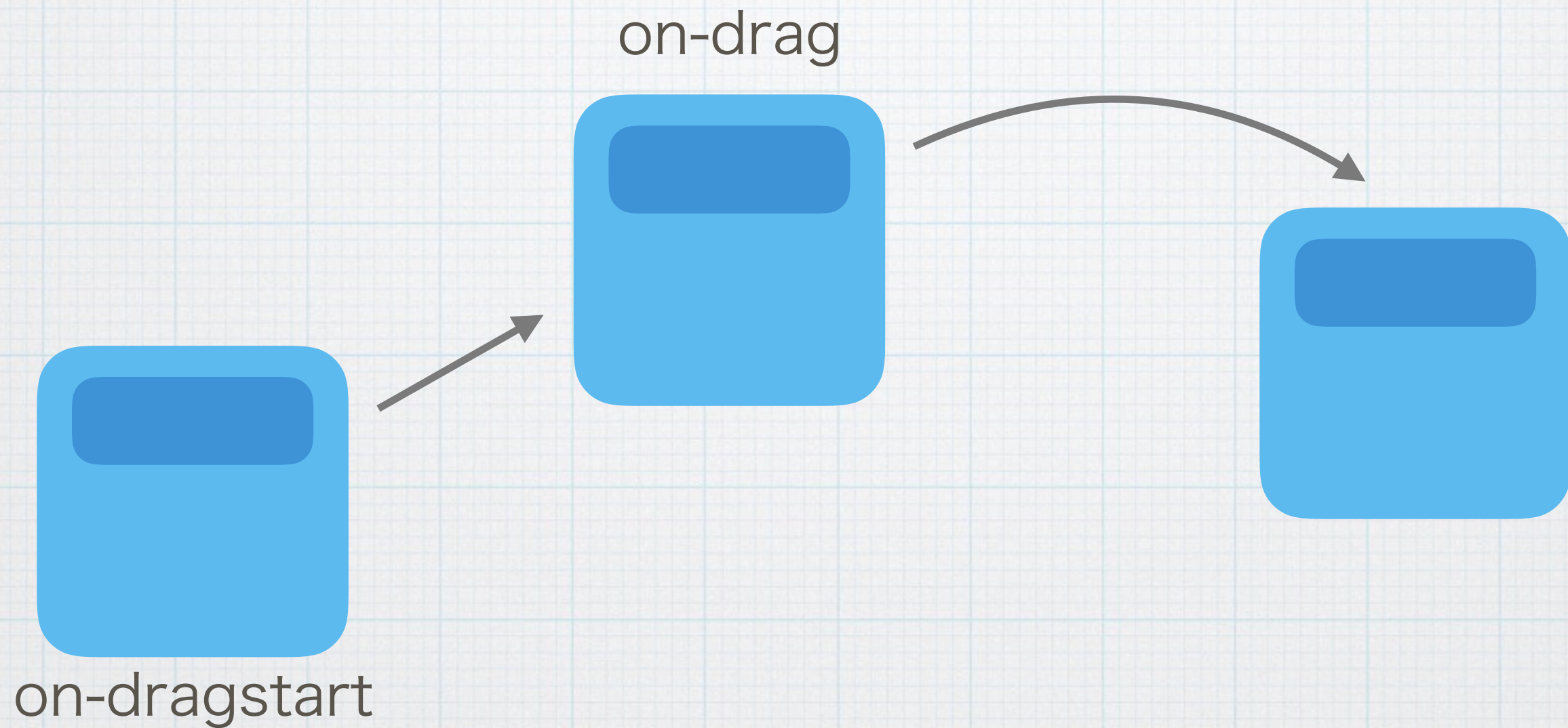


# 拖移示例



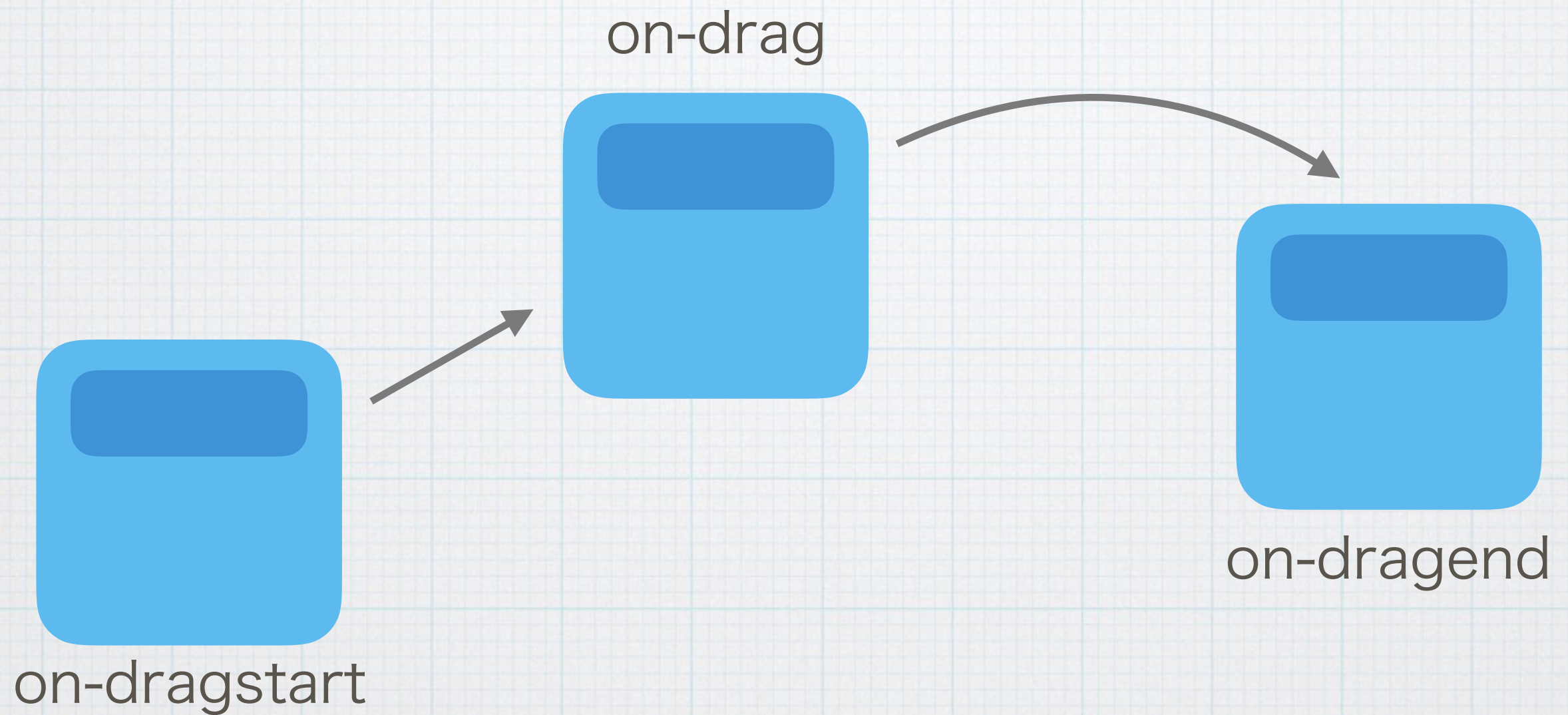


# 拖移示例



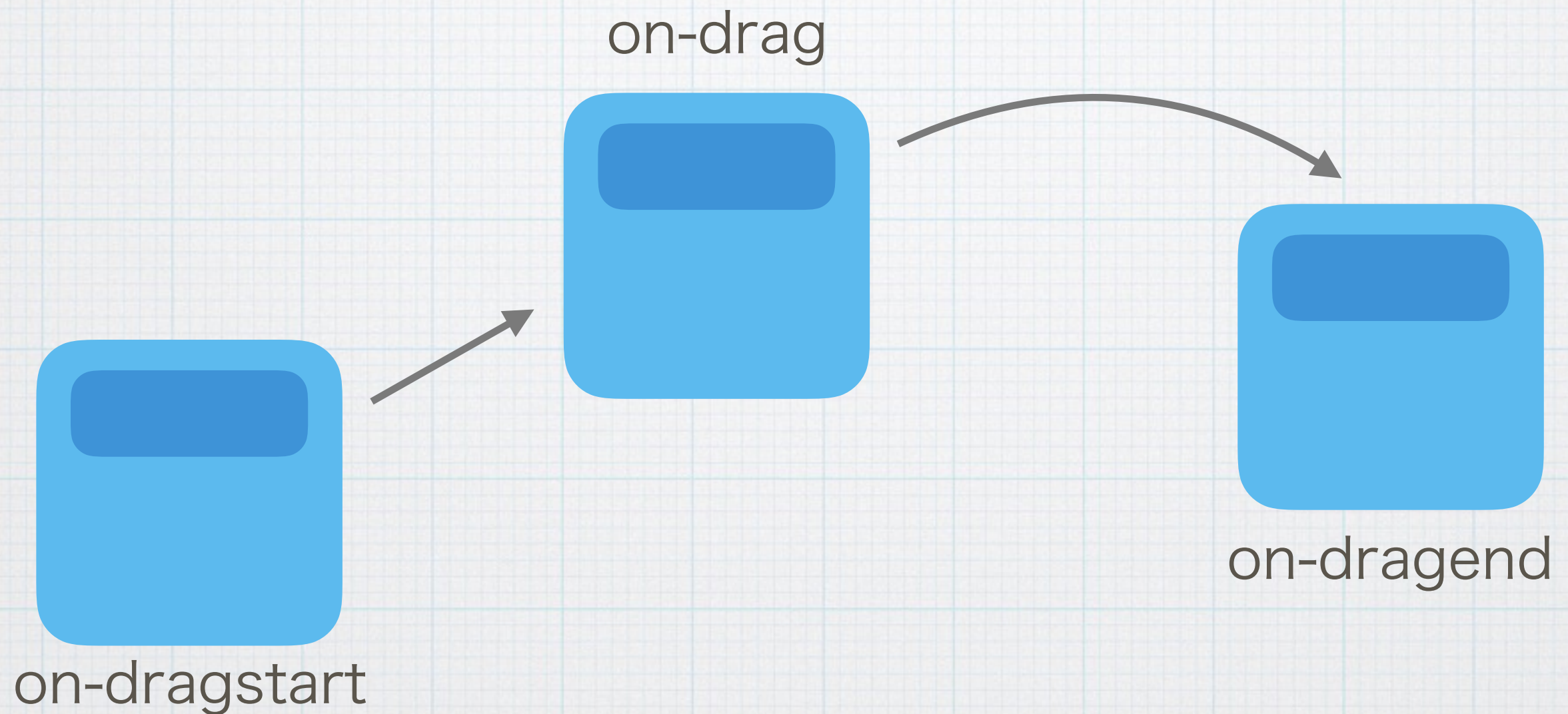


# 拖移示例





# 拖移示例



`proxy === source.parent`



HTML5



# HTML5 API

- 7个事件
- draggable属性
- e.preventDefault()



# HTML5 API

## ● DataTransfer对象

- setData(format, data), getData(format), clearData()
- setDragImage(image, x, y)
- effectAllowed, dropEffect
- types, items, files

## ● DataTransferItem, DataTransferItemList



# 案例

- 简单示例
- 图片上传预览
- 列表排序
- 网格排序



# 兼容性

IE	Edge <sup>*</sup>	Firefox	Chrome	Safari	Opera
		41	45	6	31
<sup>1</sup> 6		42	46	6.1	32
<sup>1</sup> 7		43	47	7	33
<sup>1</sup> 8		44	48	7.1	34
<sup>1</sup> 9		45	49	8	35
<sup>2</sup> 10	<sup>2</sup> 12	46	50	9	36
<sup>2</sup> 11	<sup>2</sup> 13	47	51	9.1	37
	<sup>2</sup> 14	48	52	TP	38
		49	53		39
		50	54		



# 兼容性





# 兼容性



# 兼容性

- IE10以下要做兼容
  - 默认只有和是自动支持
  - 其他元素要在onselectstart事件调用element.dragDrop()



# 兼容性

- IE10以下要做兼容
  - 默认只有和是自动支持
  - 其他元素要在onselectstart事件调用element.dragDrop()
- IE10以下不支持文件拖拽



# 兼容性

- IE10以下要做兼容
  - 默认只有和是自动支持
  - 其他元素要在onselectstart事件调用element.dragDrop()
- IE10以下不支持文件拖拽
- IE10以下DataTransfer类型有限



# 兼容性

- IE10以下要做兼容
  - 默认只有和是自动支持
  - 其他元素要在onselectstart事件调用element.dragDrop()
- IE10以下不支持文件拖拽
- IE10以下DataTransfer类型有限
- IE10以下没有dragImage



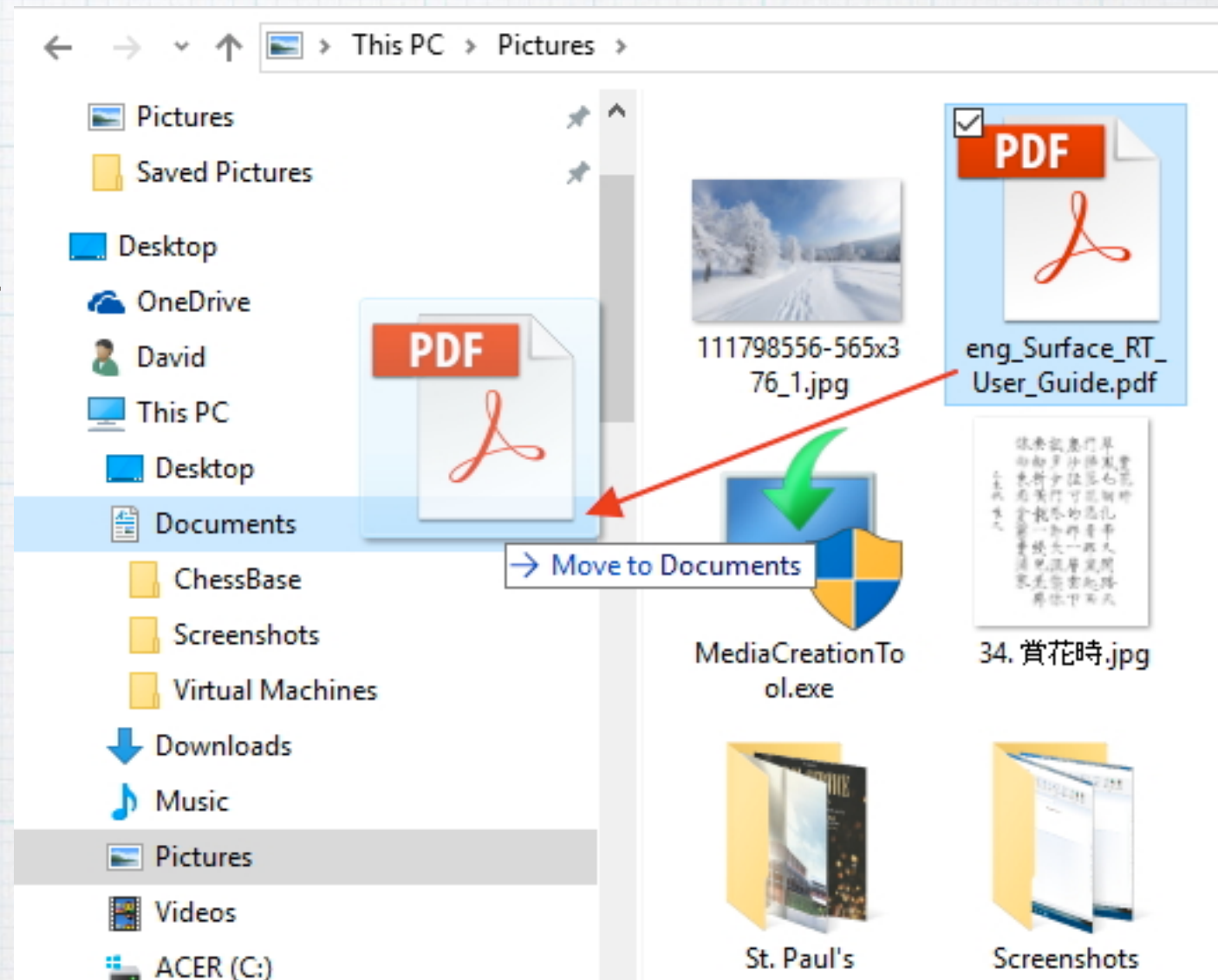
# 兼容性

- IE10以下要做兼容
  - 默认只有<a>和<img>是自动支持
  - 其他元素要在onselectstart事件调用element.dragDrop()
- IE10以下不支持文件拖拽
- IE10以下DataTransfer类型有限
- IE10以下没有dragImage
- IE10+(包括Edge)不支持setDragImage



# 兼容性

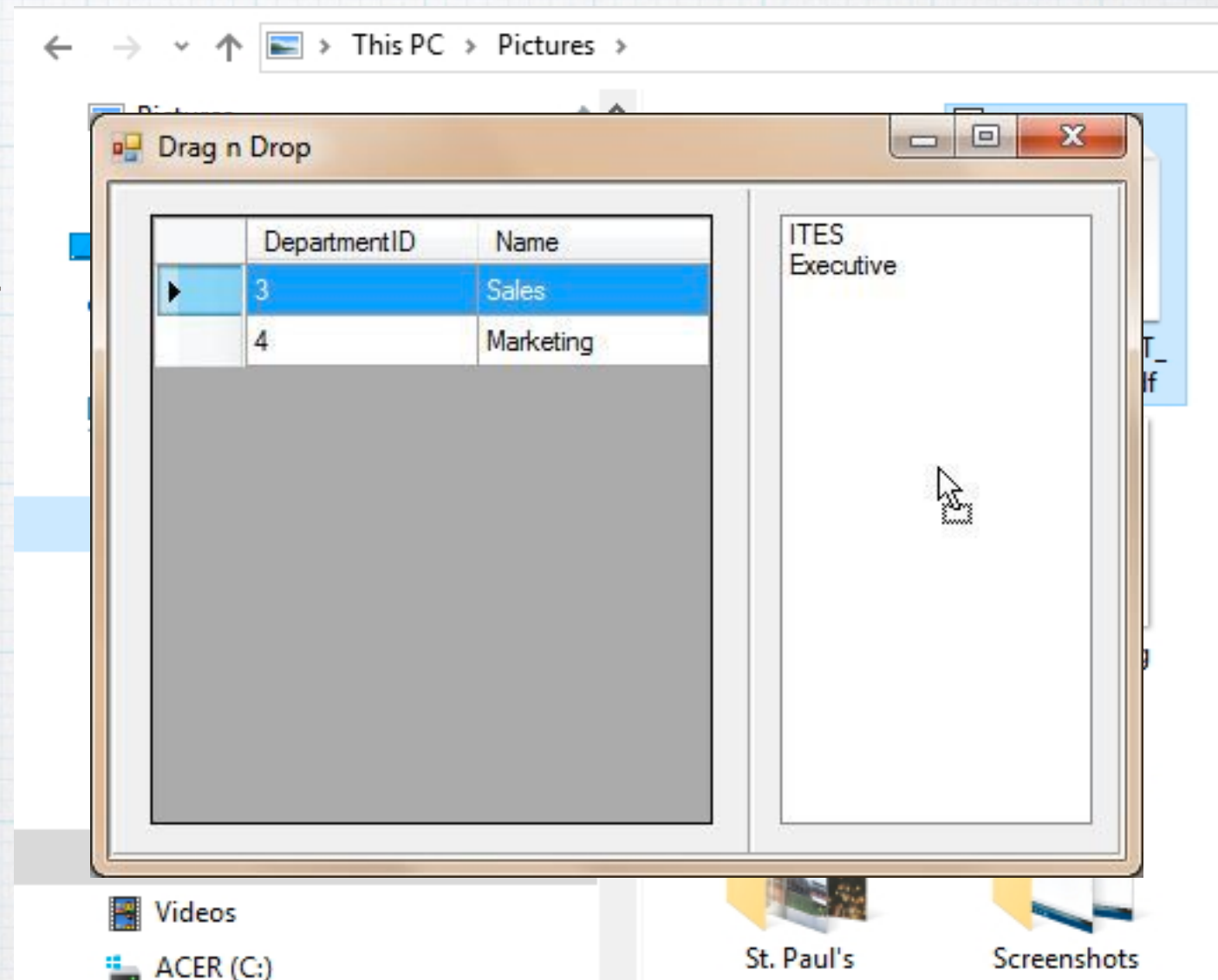
- IE10以下要做兼容
  - 默认只有<a>和<img>是自动支持
  - 其他元素要在onselectstart事件调用element.dragDrop()
- IE10以下不支持文件拖拽
- IE10以下DataTransfer类型有限
- IE10以下没有dragImage
- IE10+(包括Edge)不支持setDragImage





# 兼容性

- IE10以下要做兼容
  - 默认只有<a>和<img>是自动支持
  - 其他元素要在onselectstart事件调用element.dragDrop()
- IE10以下不支持文件拖拽
- IE10以下DataTransfer类型有限
- IE10以下没有dragImage
- IE10+(包括Edge)不支持setDragImage





# 兼容性

- IE10以下要做兼容
  - 默认只有<a>和<img>是自动支持
  - 其他元素要在onselectstart事件调用element.dragDrop()
- IE10以下不支持文件拖拽
- IE10以下DataTransfer类型有限
- IE10以下没有dragImage
- IE10+(包括Edge)不支持setDragImage





# 兼容性

- IE10以下要做兼容
  - 默认只有<a>和<img>是自动支持
  - 其他元素要在onselectstart事件调用element.dragDrop()
- IE10以下不支持文件拖拽
- IE10以下DataTransfer类型有限
- IE10以下没有dragImage
- IE10+(包括Edge)不支持setDragImage



[Demo在此](#)



# 前端的困境



# 前端的困境

- 对于拖放操作，官方库在IE10以下效果不理想



# 前端的困境

- 对于拖放操作，官方库在IE10以下效果不理想
- 对于拖移操作，官方库不处理，需要开发者自行封装鼠标事件，但研究成本较大



# 前端的困境

- 对于拖放操作，官方库在IE10以下效果不理想
- 对于拖移操作，官方库不处理，需要开发者自行封装鼠标事件，但研究成本较大
- 野生库太多太杂，引入产品中不是最佳的解决方案



# 前端的困境





# 前端的困境



面对拖拽，前端工程师有种淡淡的无力感



# 鼠标事件封装



拖拽的本质就是对鼠标事件的进一层封装



拖拽的本质就是对鼠标事件的进一层封装

“DragEvent inherits properties from MouseEvent and Event.”



# 事件分析





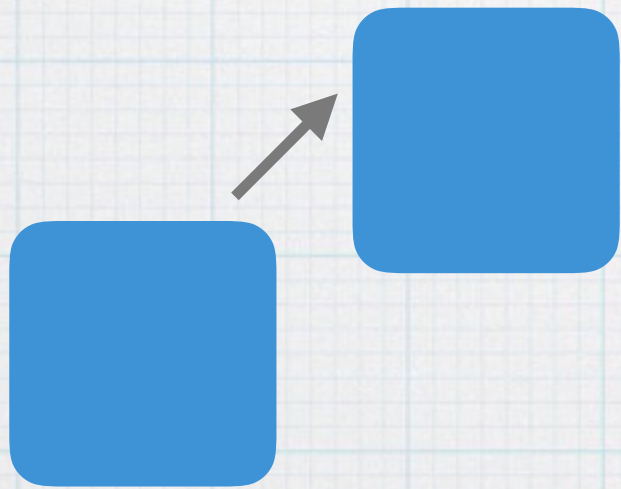
# 事件分析



on-mousedown



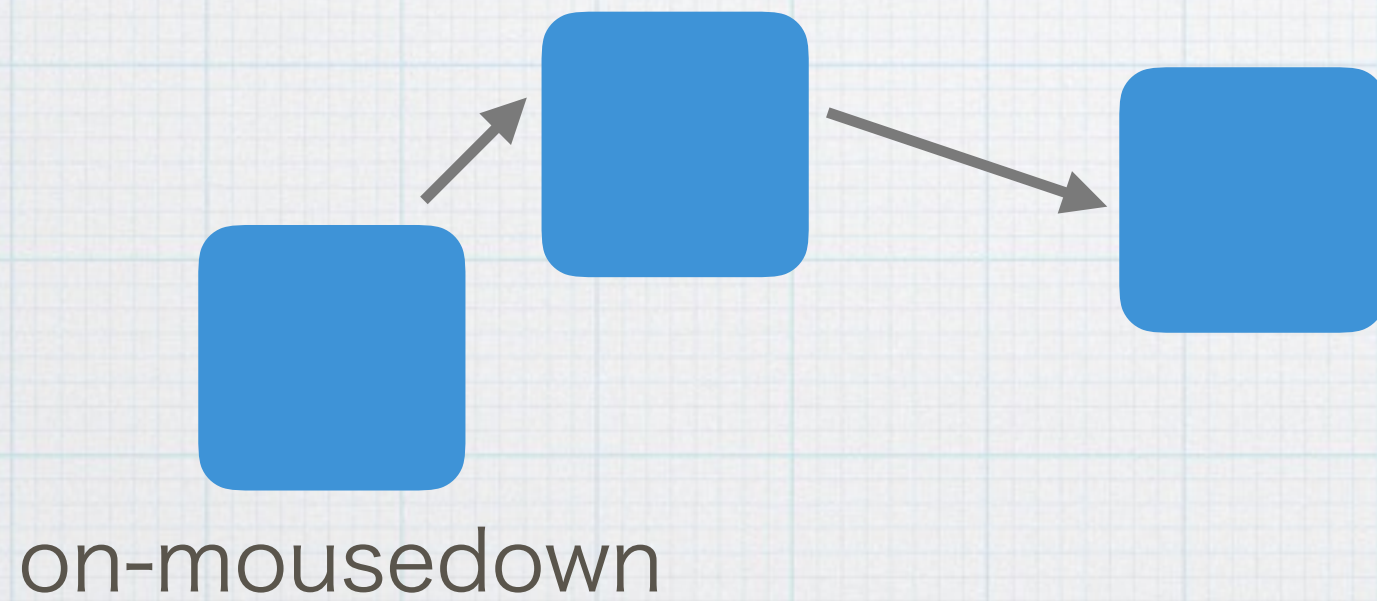
# 事件分析



on-mousedown

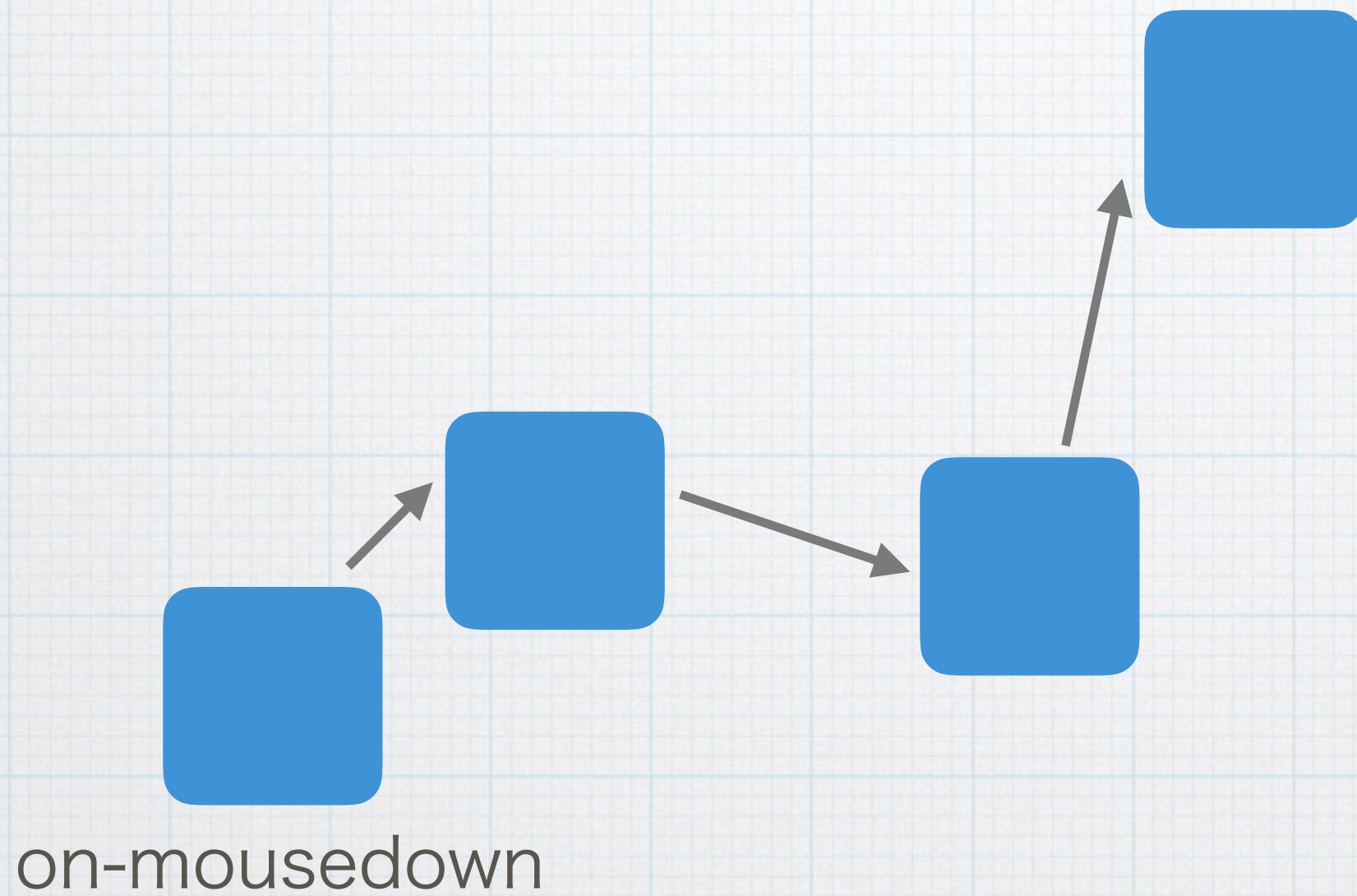


# 事件分析



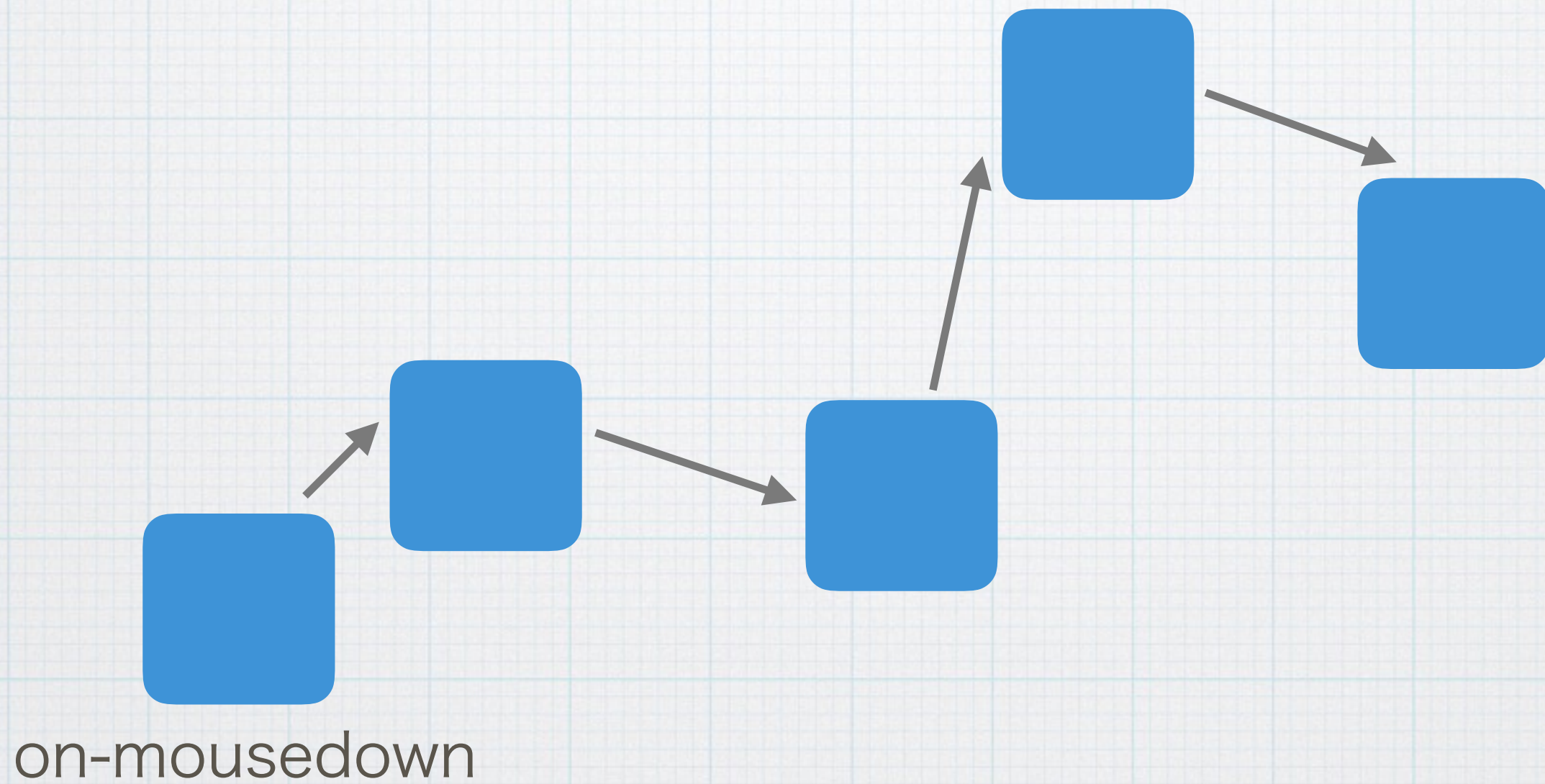


# 事件分析



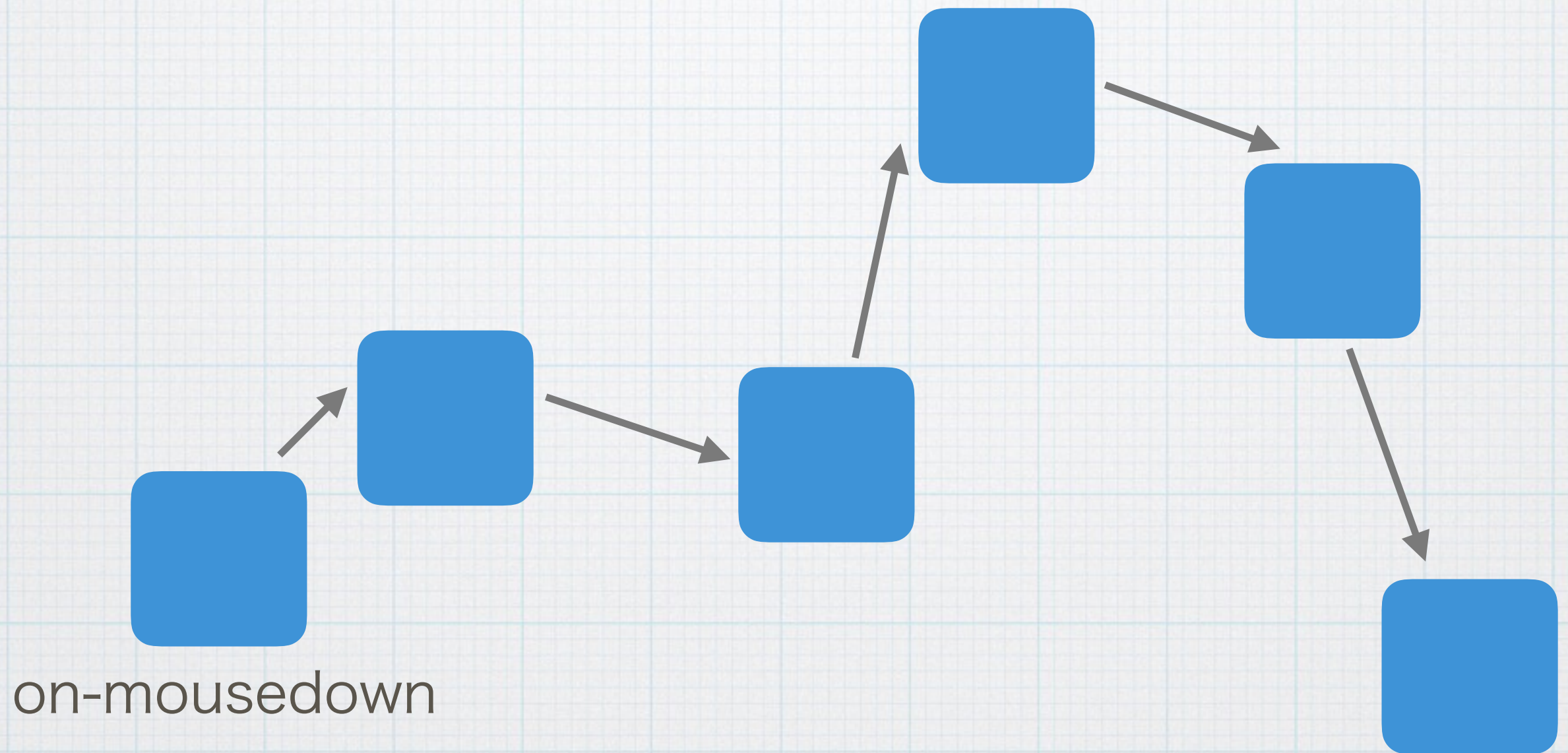


# 事件分析



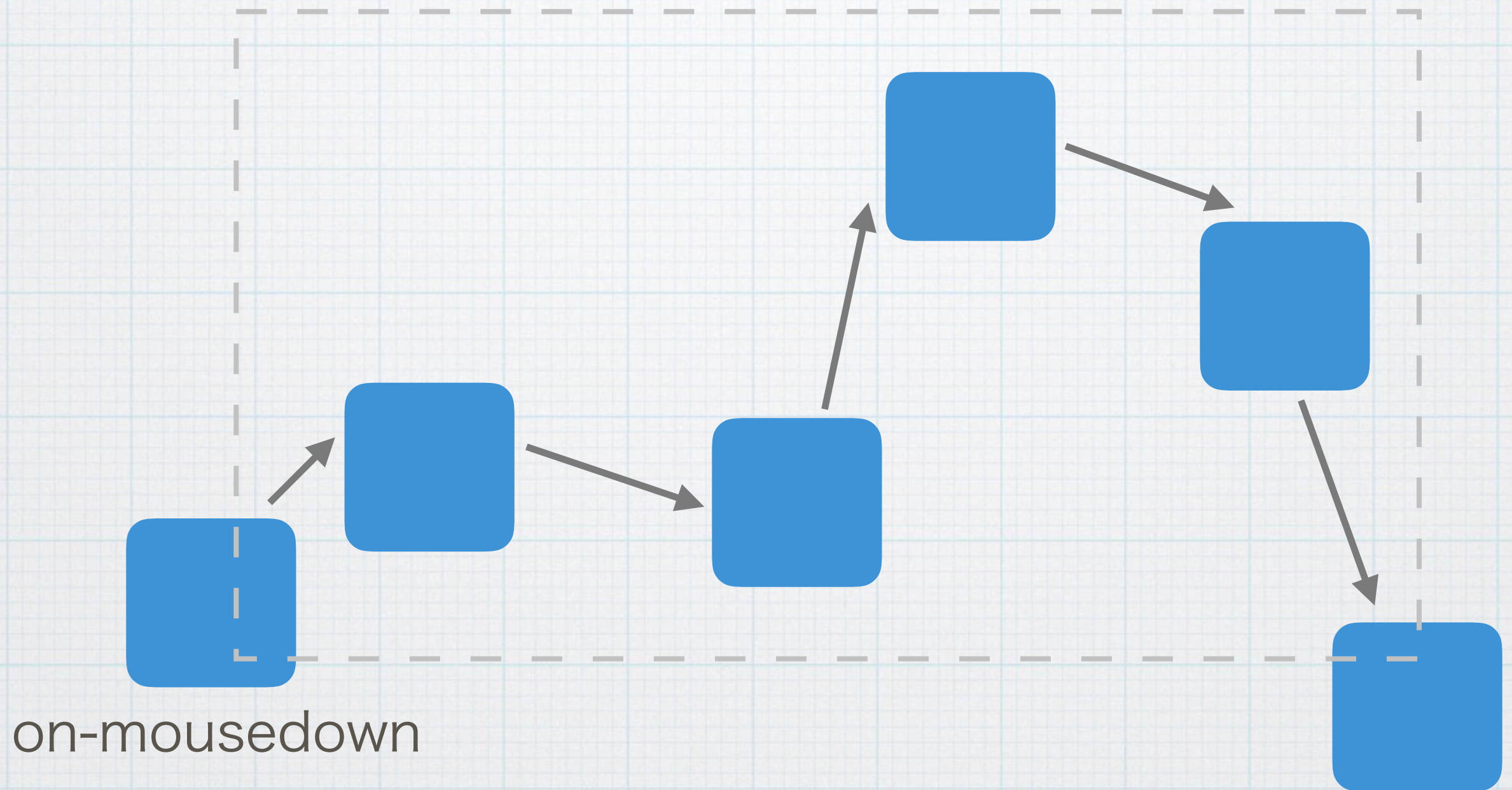


# 事件分析



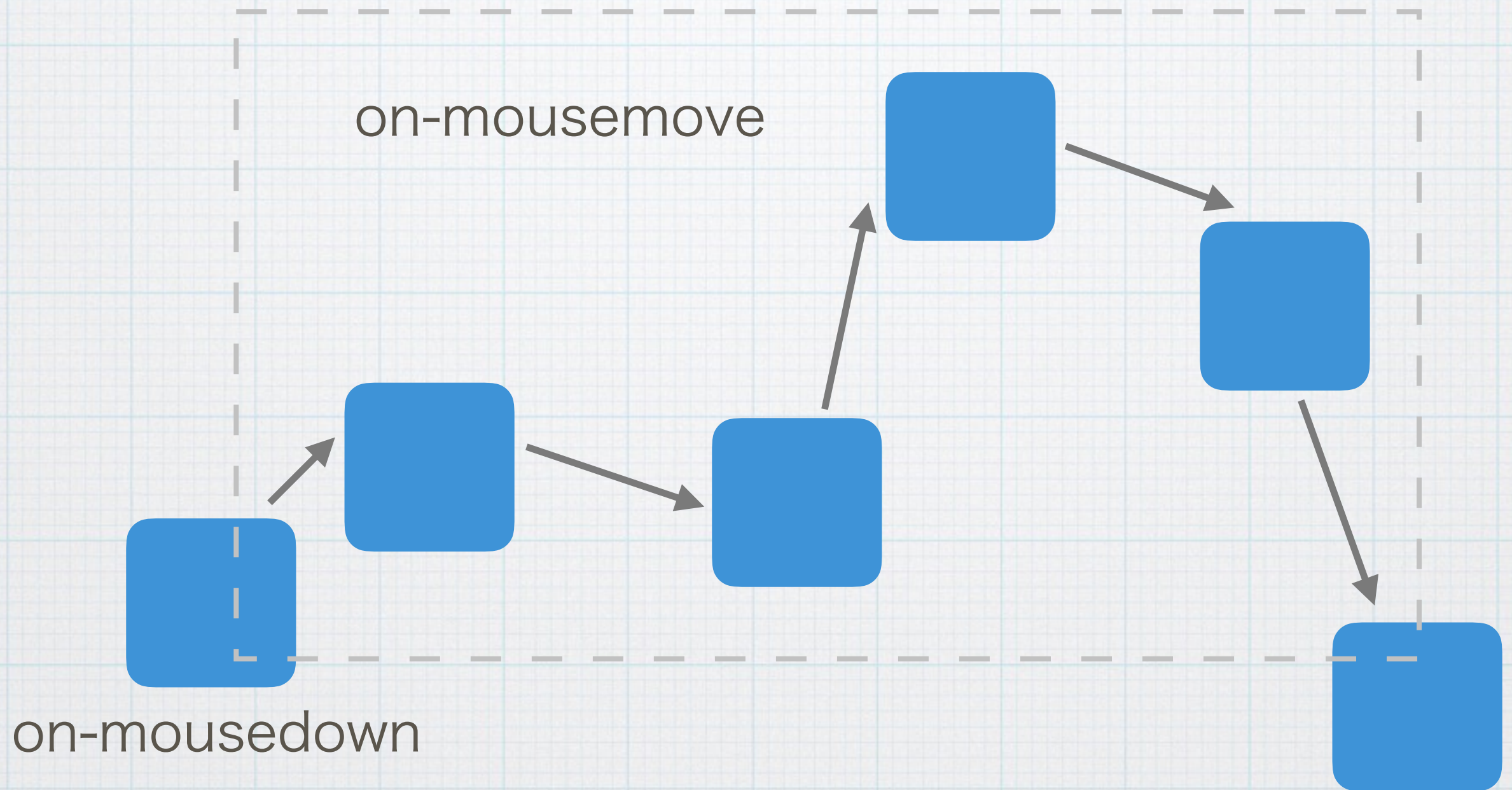


# 事件分析



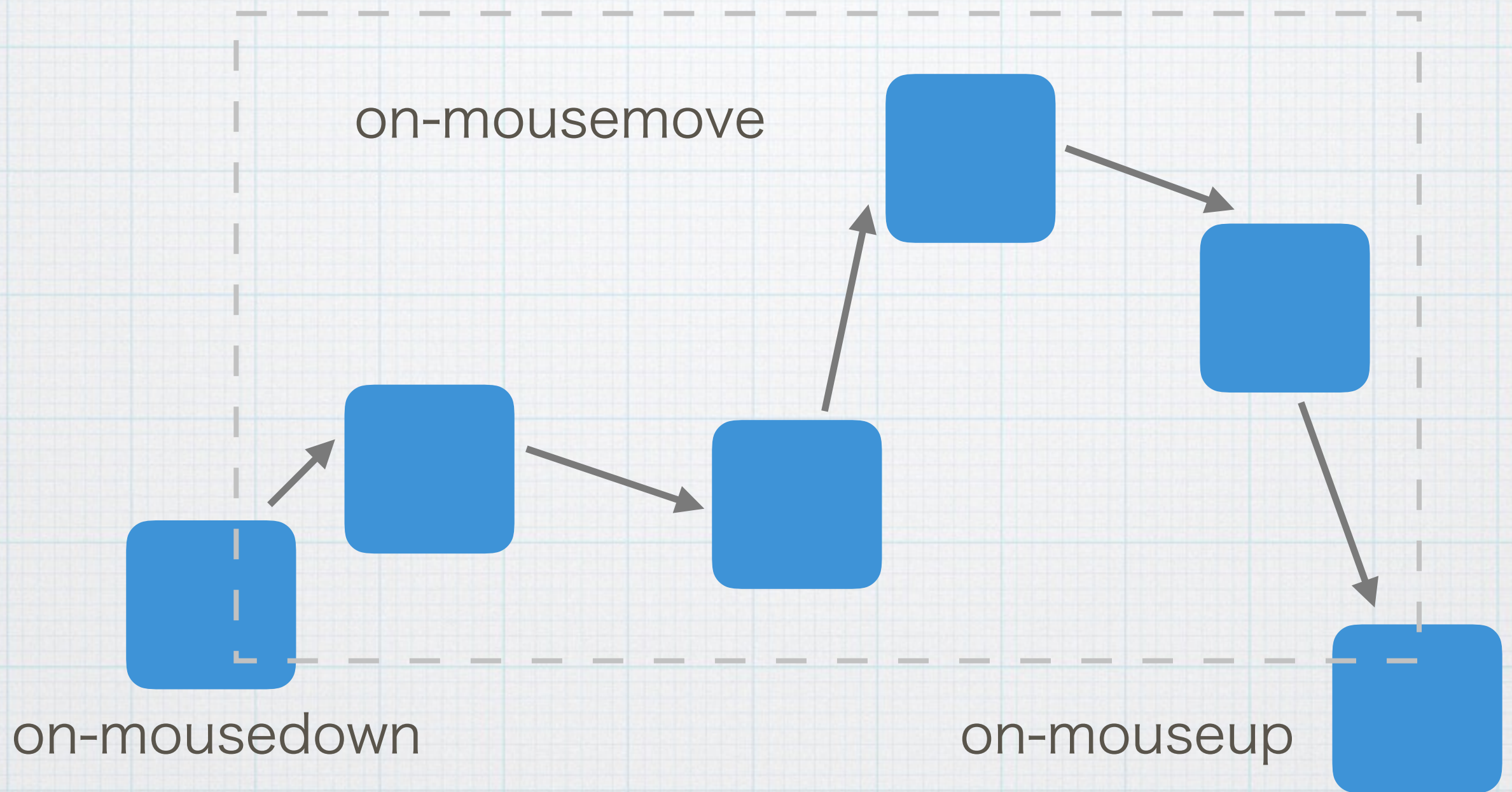


# 事件分析



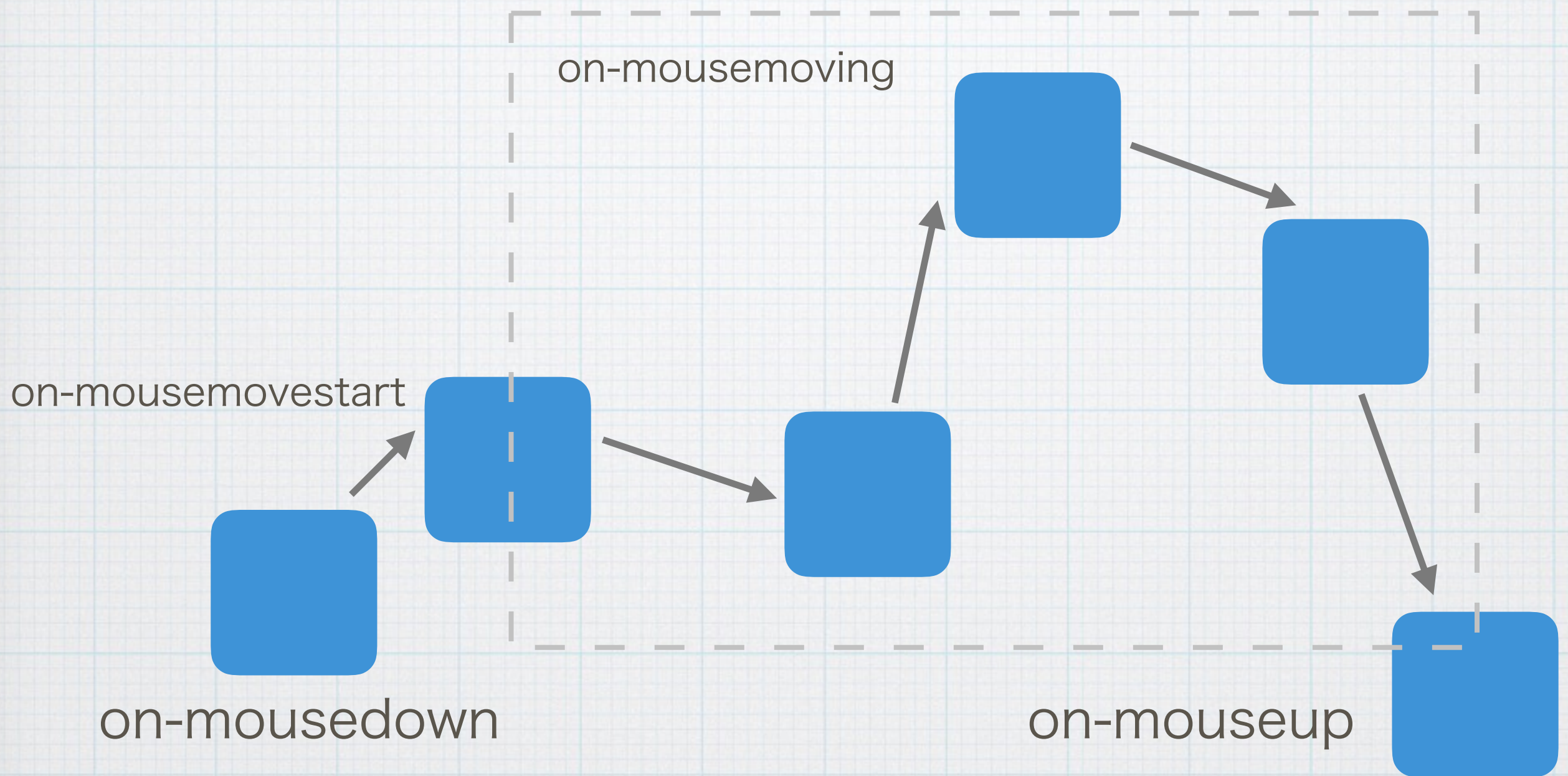


# 事件分析





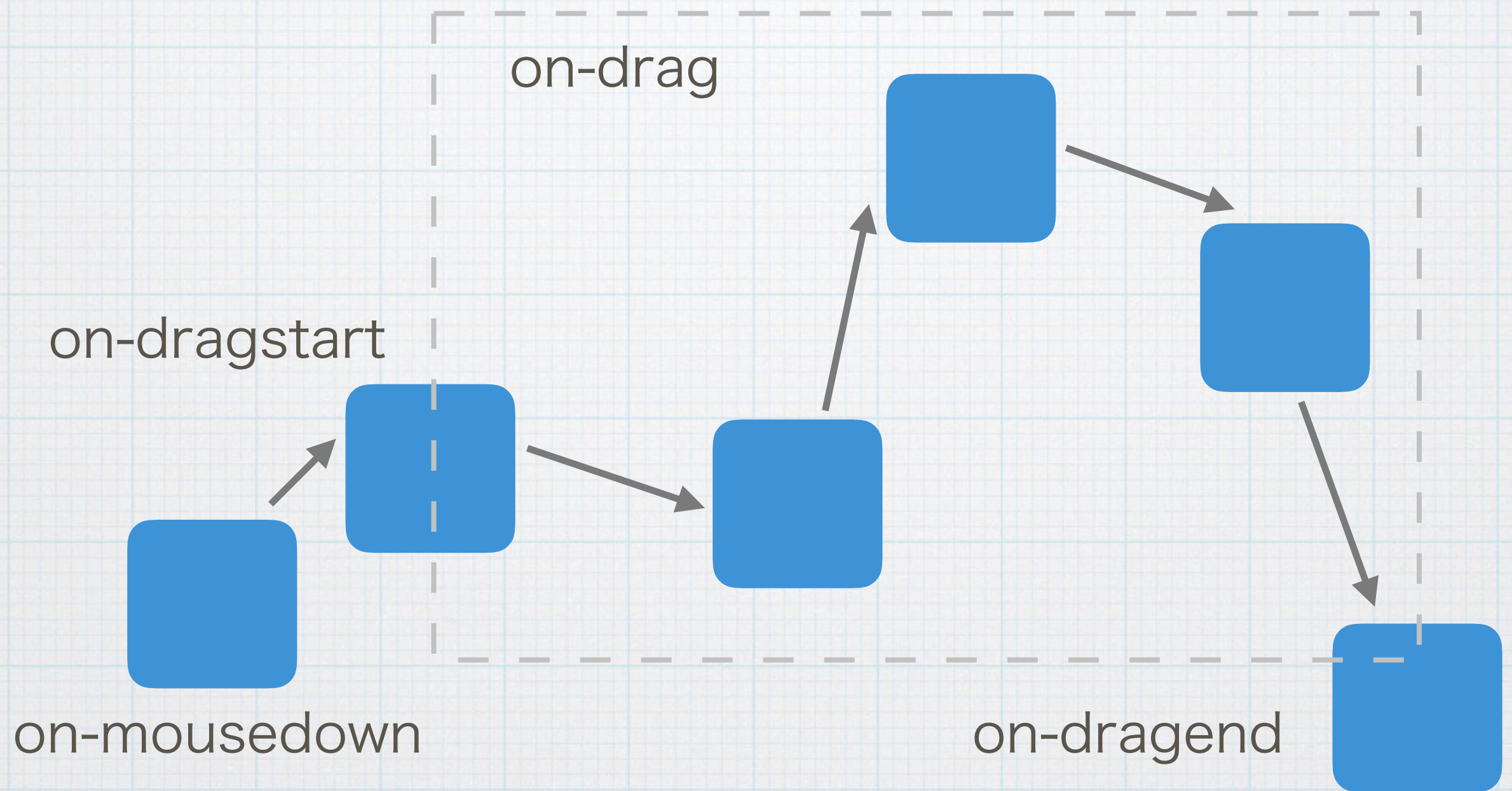
# 事件拆解



[Demo在此](#)



# 事件抛出





# 事件处理

- on-mousedown

- 绑定window的mousemove和mouseup事件



# 事件处理

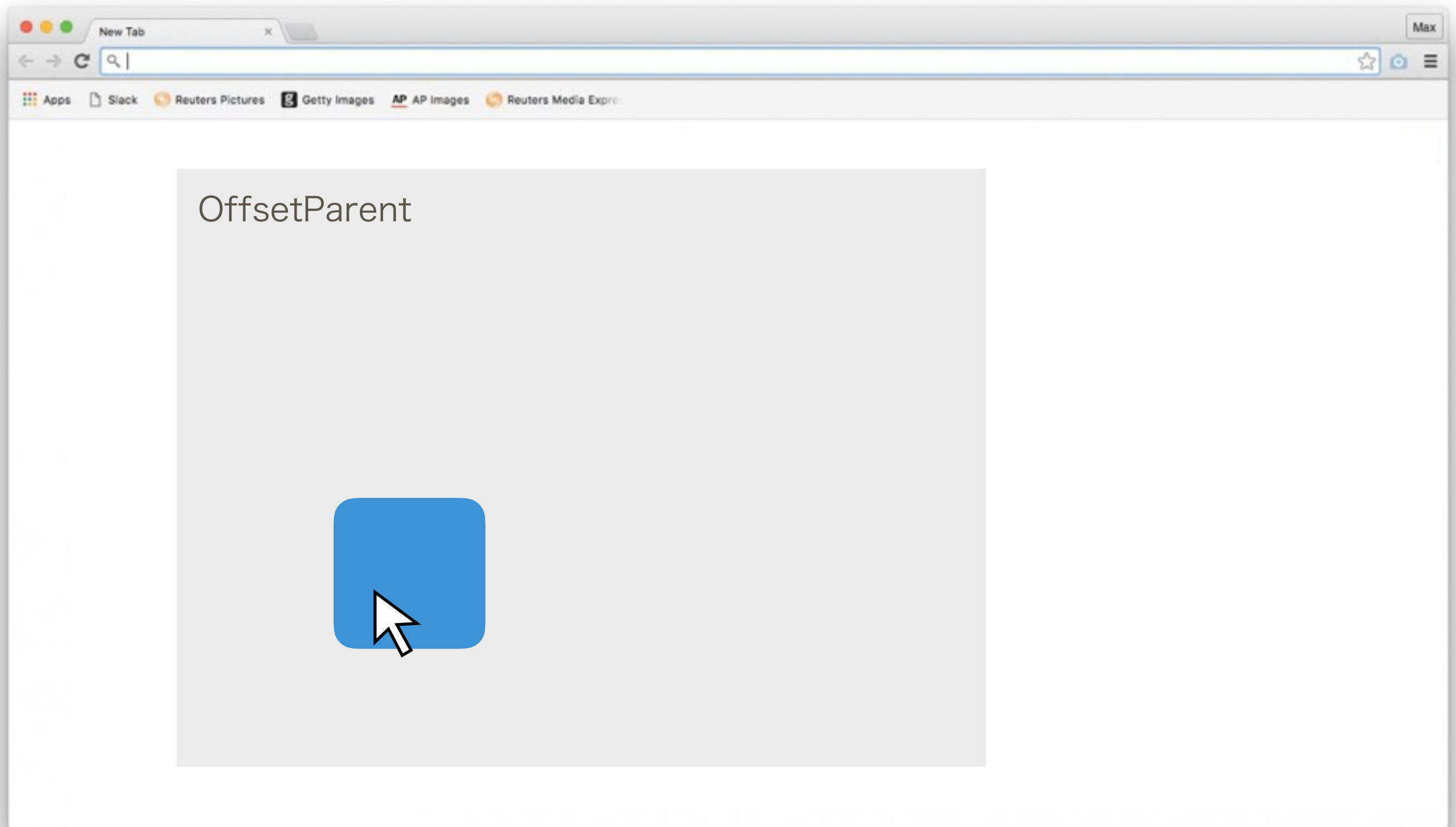
- on-mousemovestart (拖拽开始)
  - 初始化拖拽参数
    - 获取代理元素(proxy)
    - 记录鼠标的初始坐标(startX, startY)
    - 记录代理元素的初始位置(startLeft, startTop)
  - 抛出on-dragstart事件



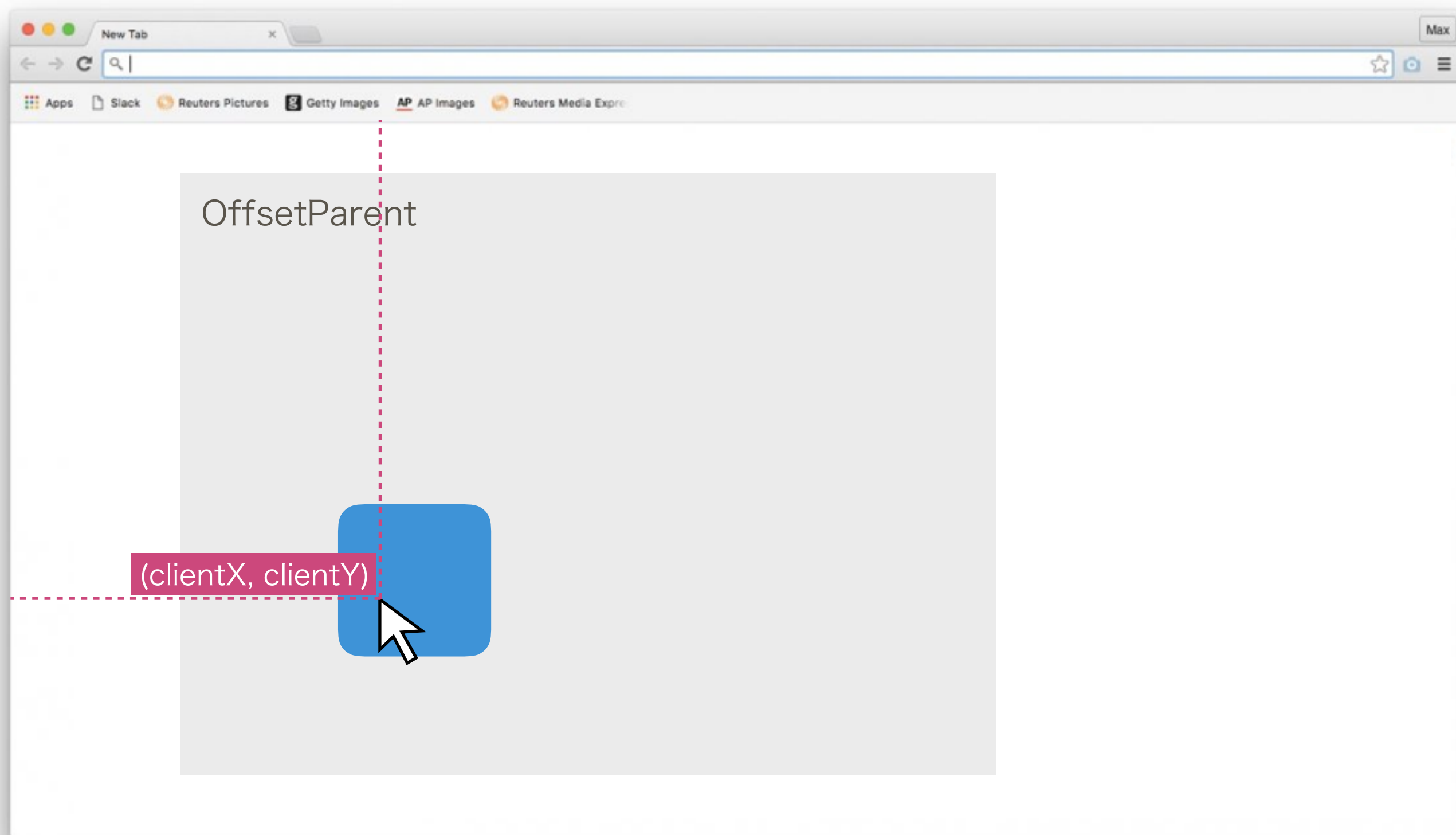
# 事件处理

- on-mousemoving (拖拽进行中)
  - 更新拖拽参数
    - 记录鼠标新的坐标(clientX, clientY)
    - 计算并设置代理元素新的位置(left, top)
  - 抛出on-drag事件

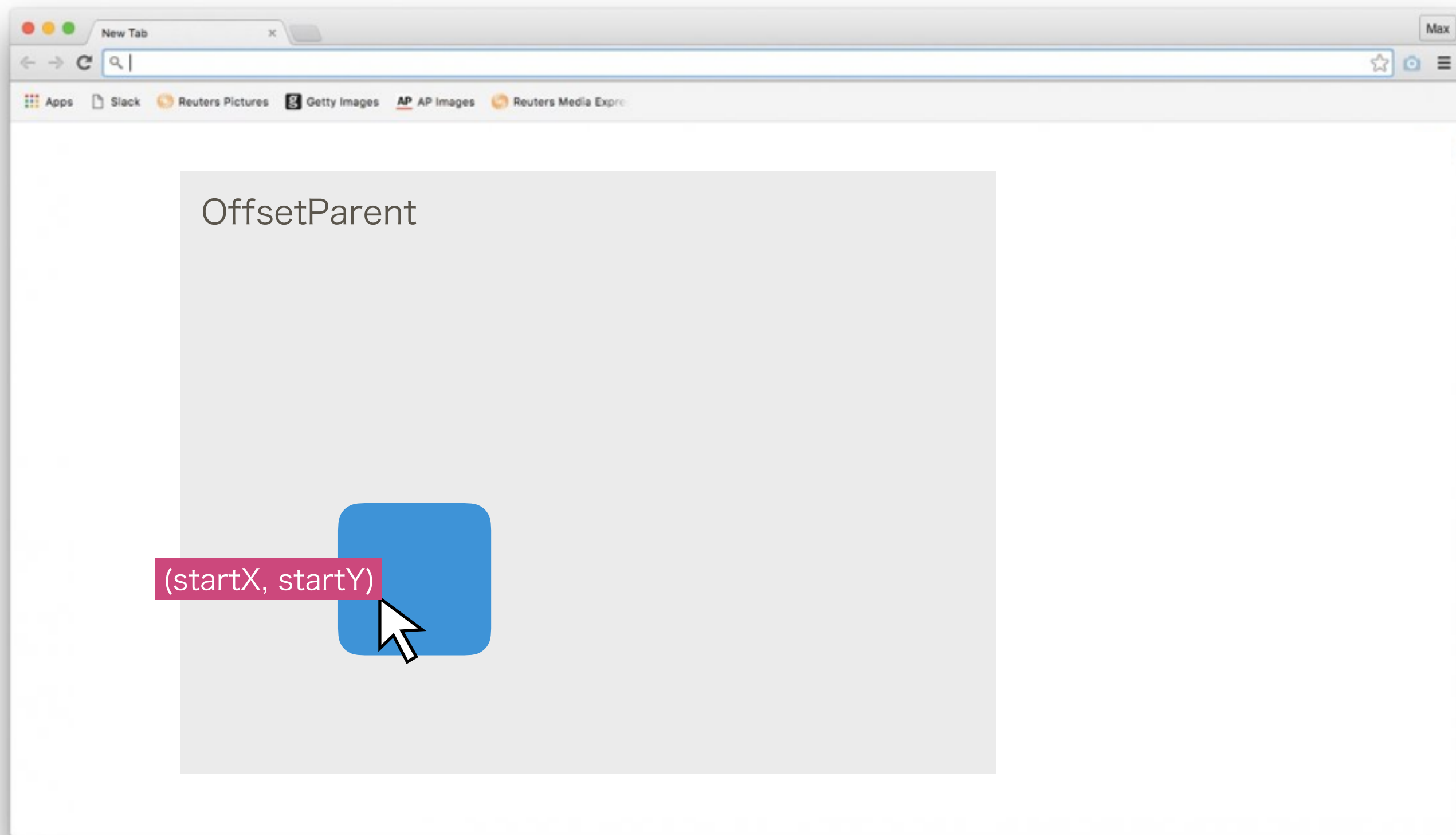




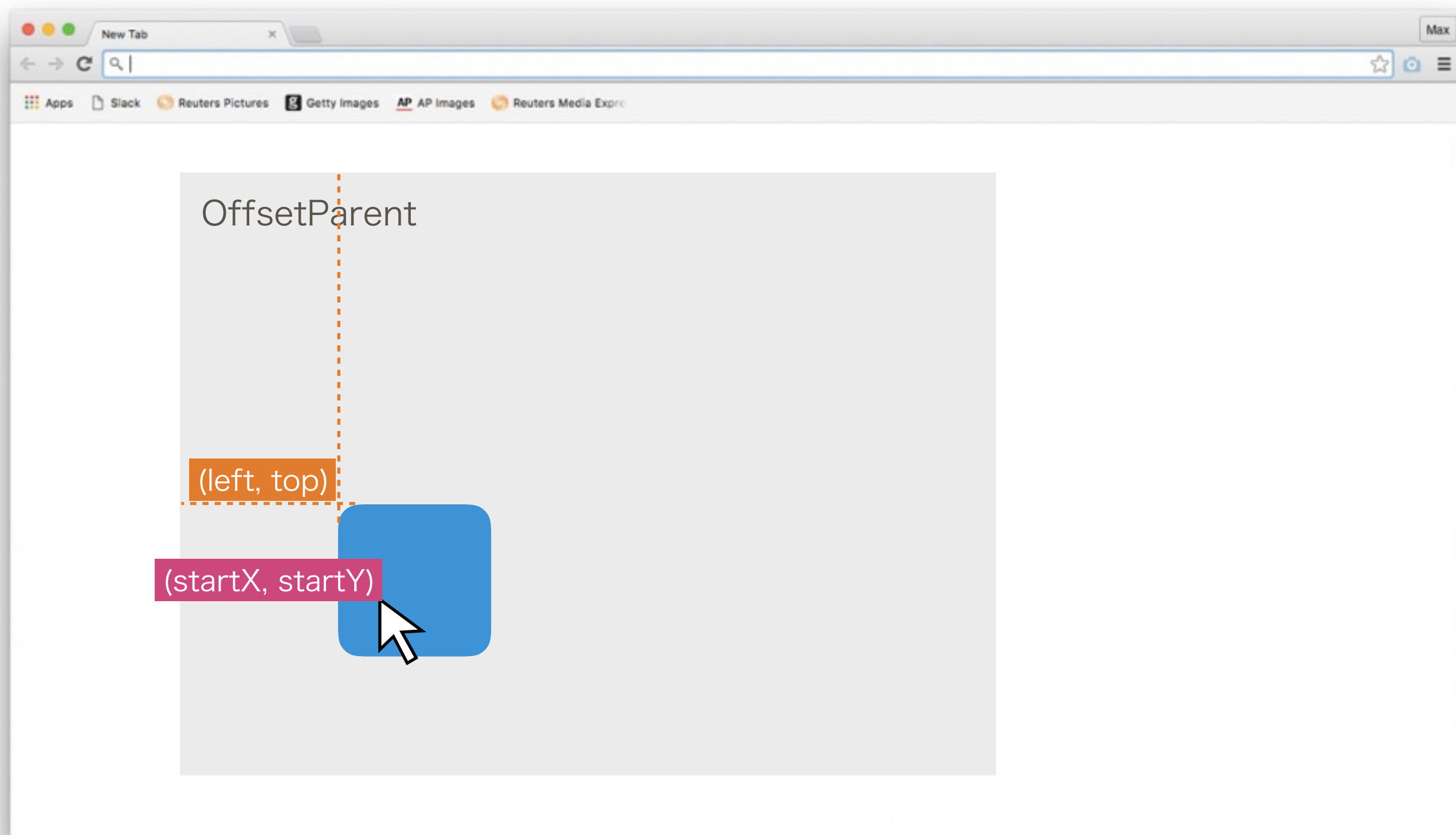




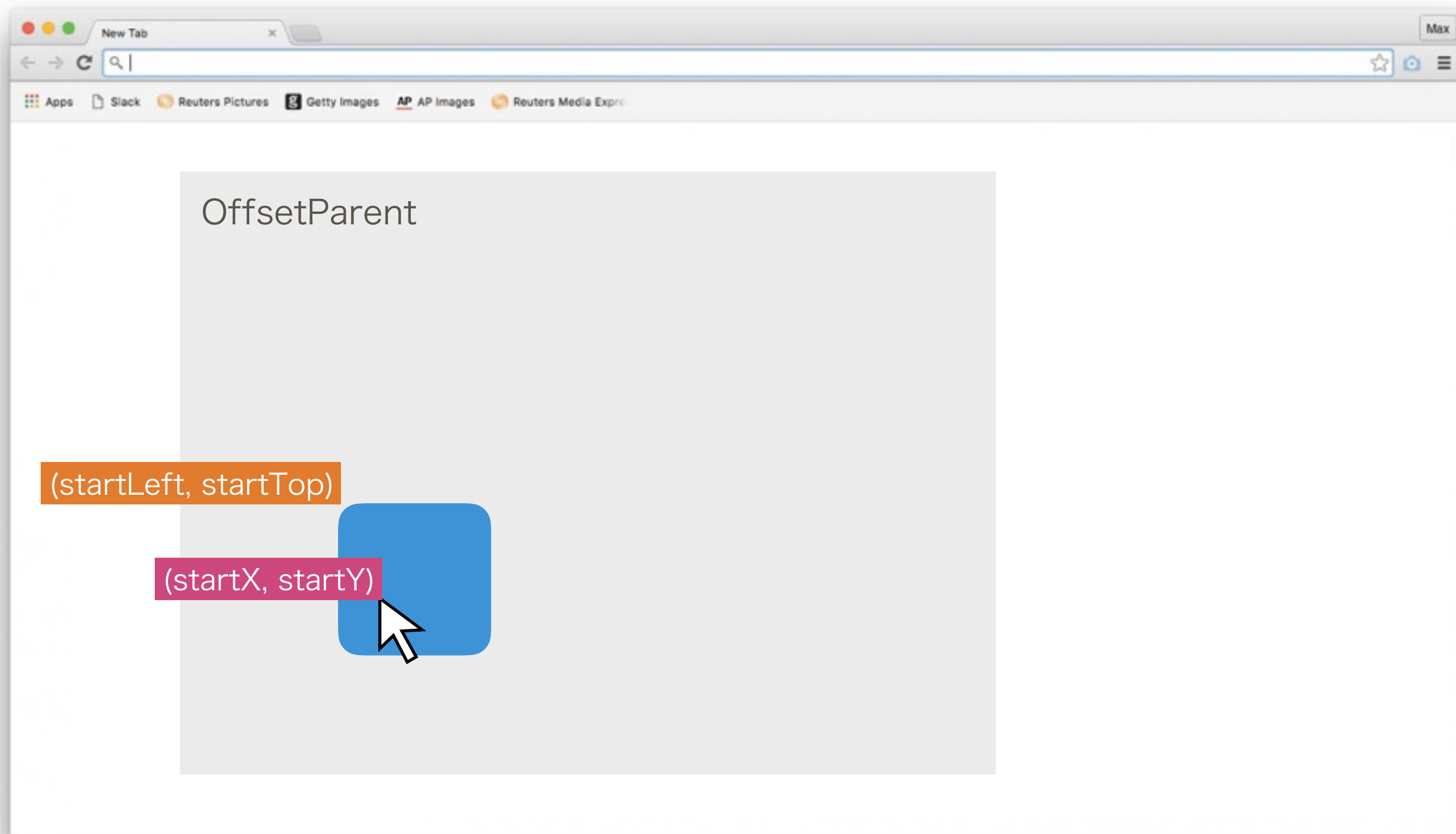




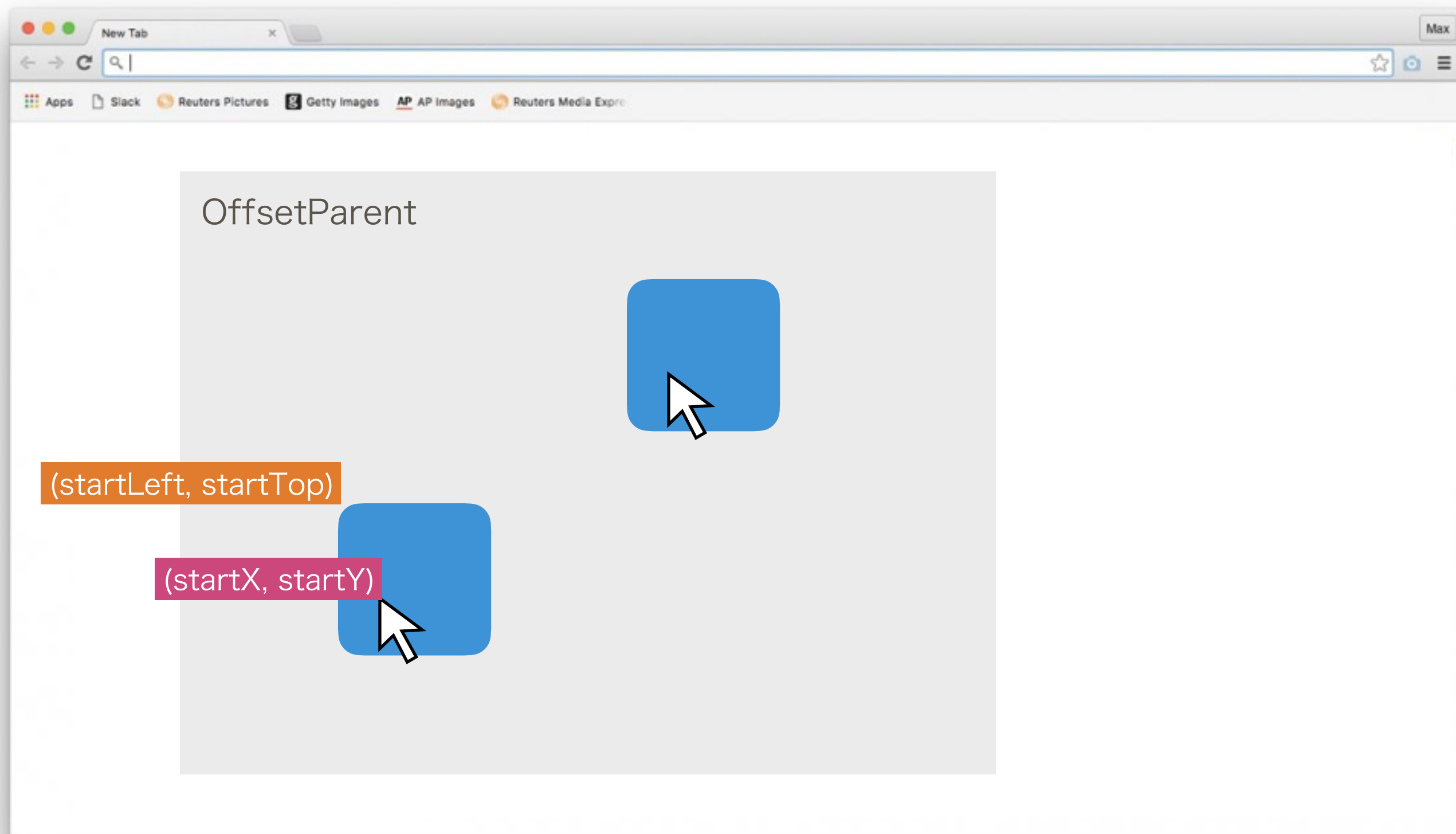




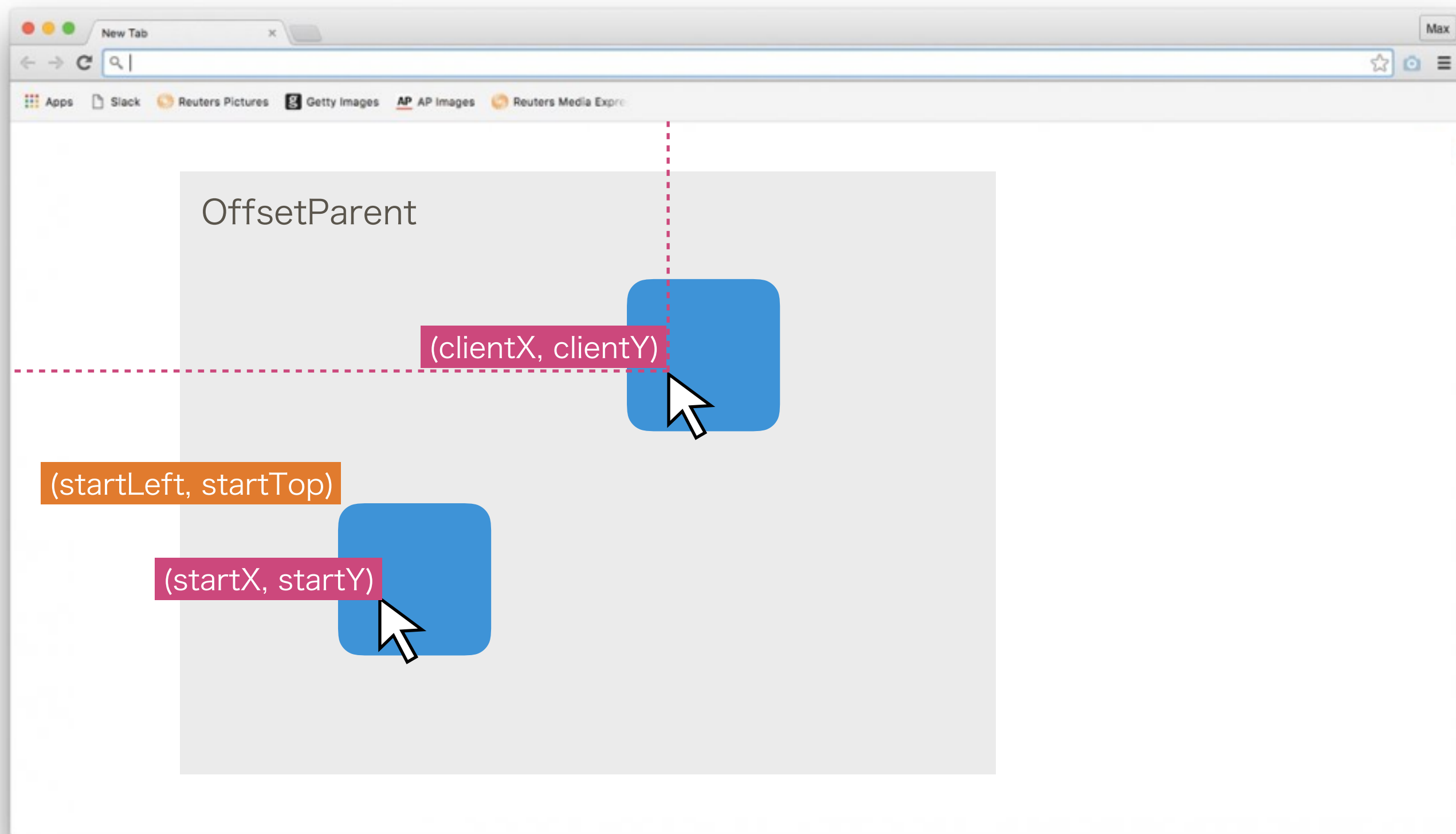




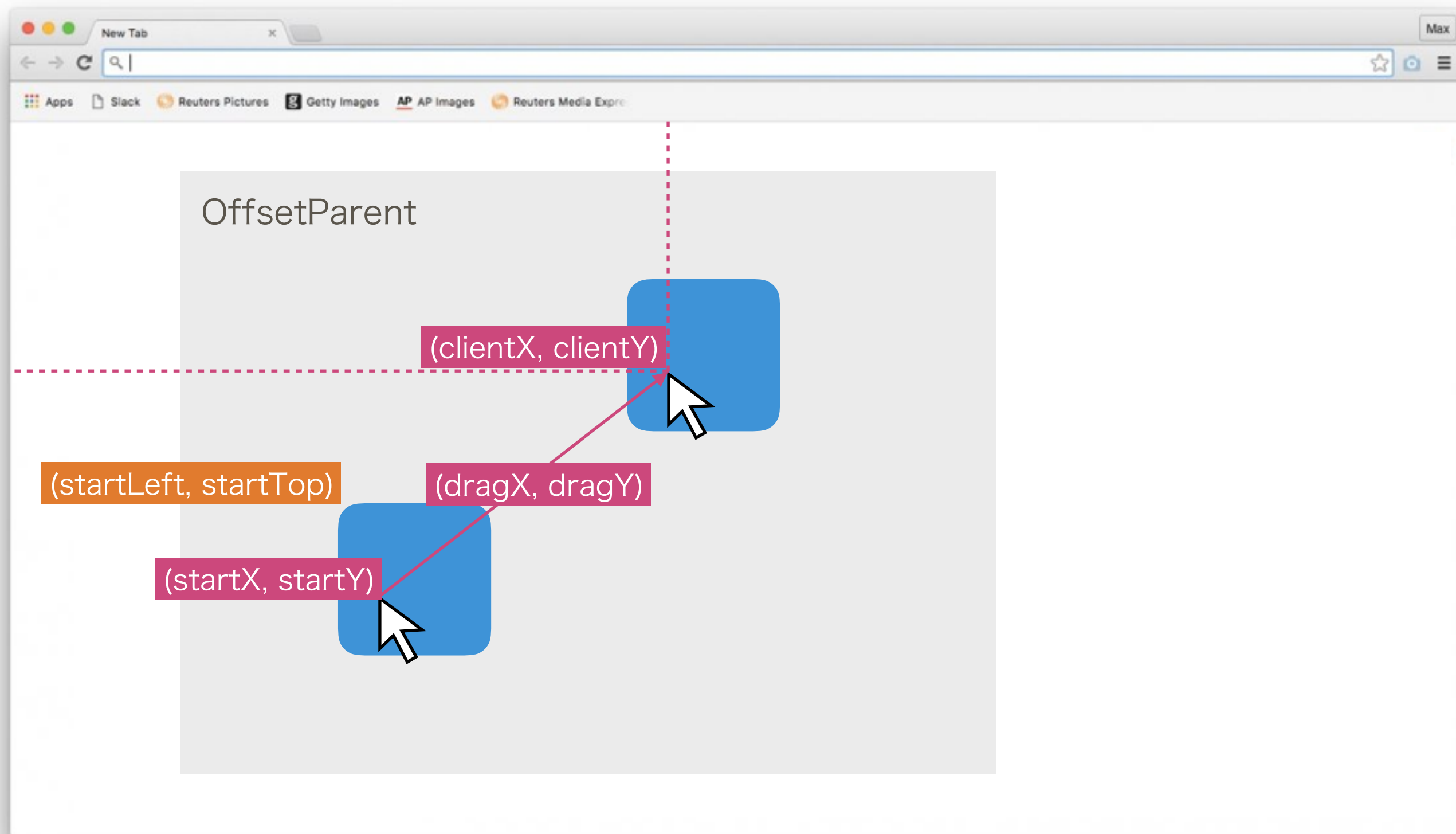




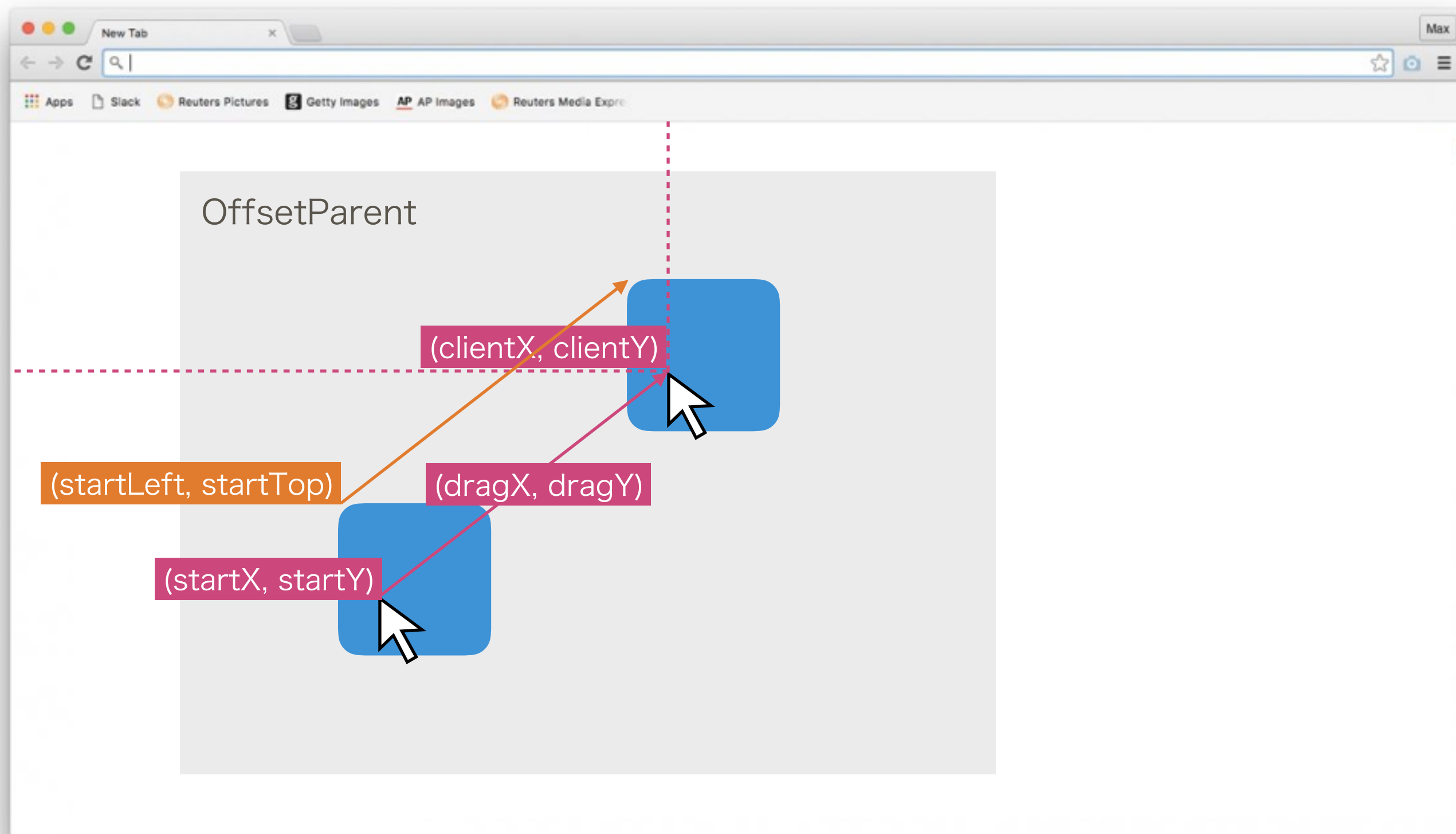




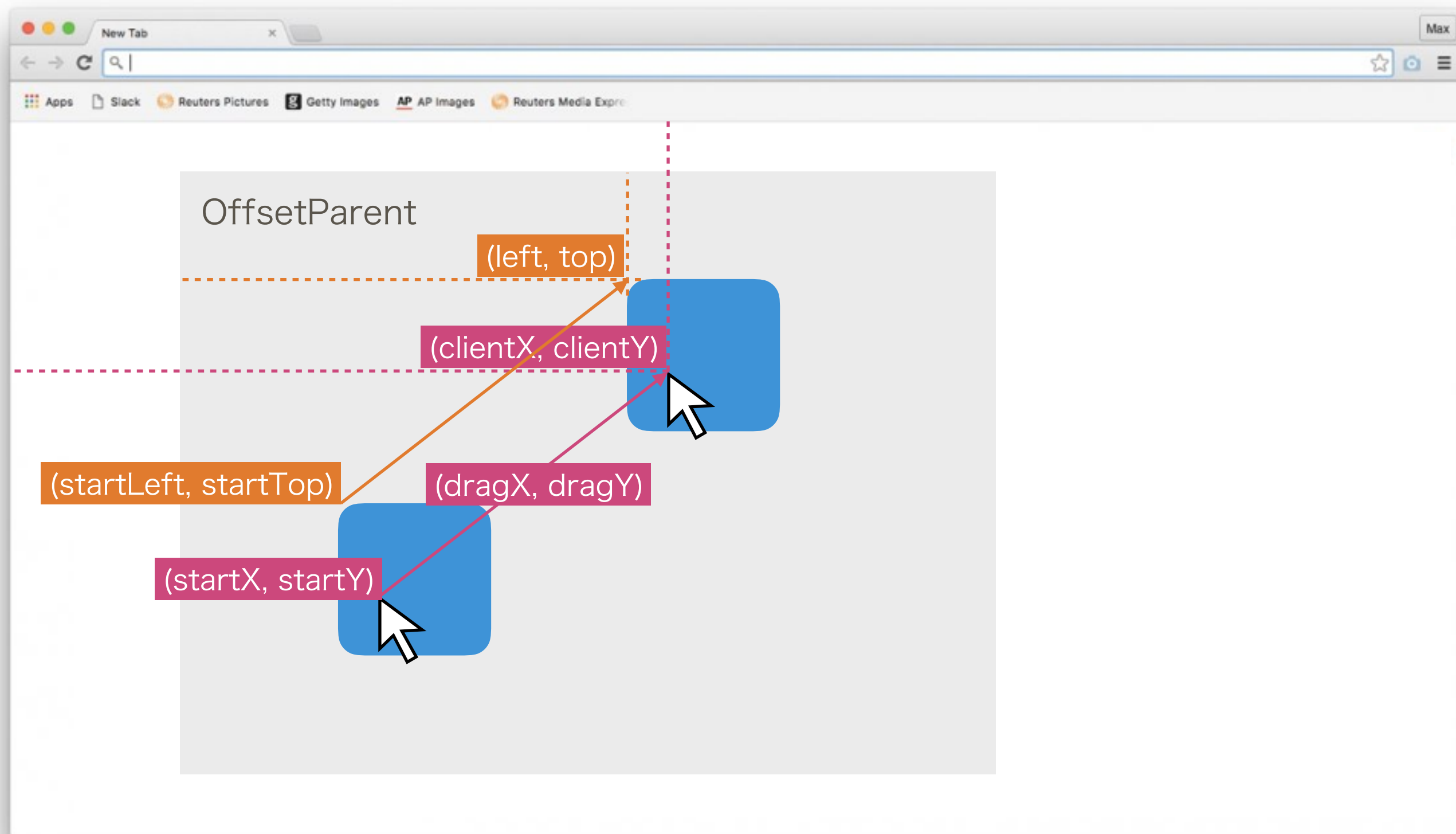




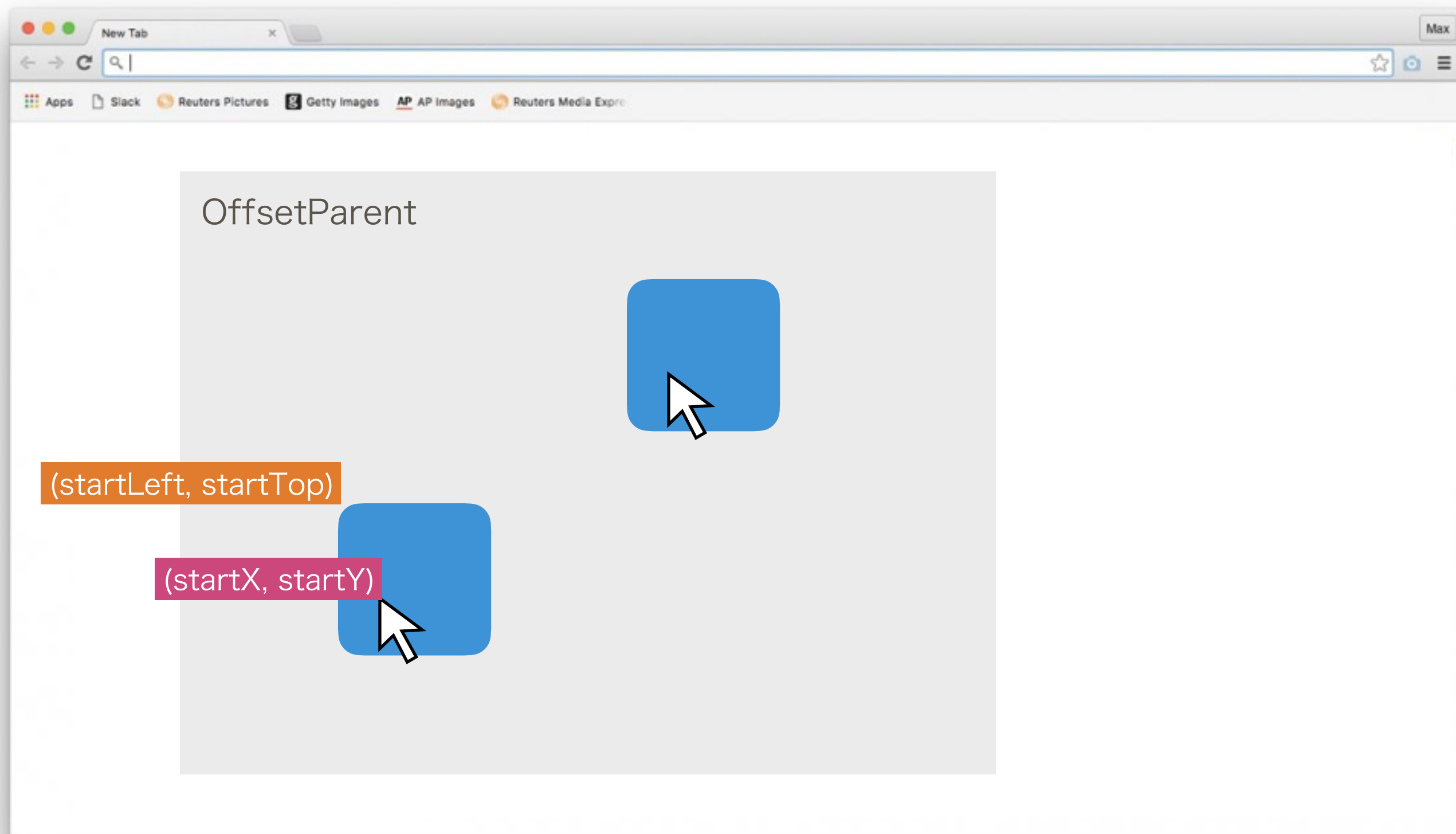




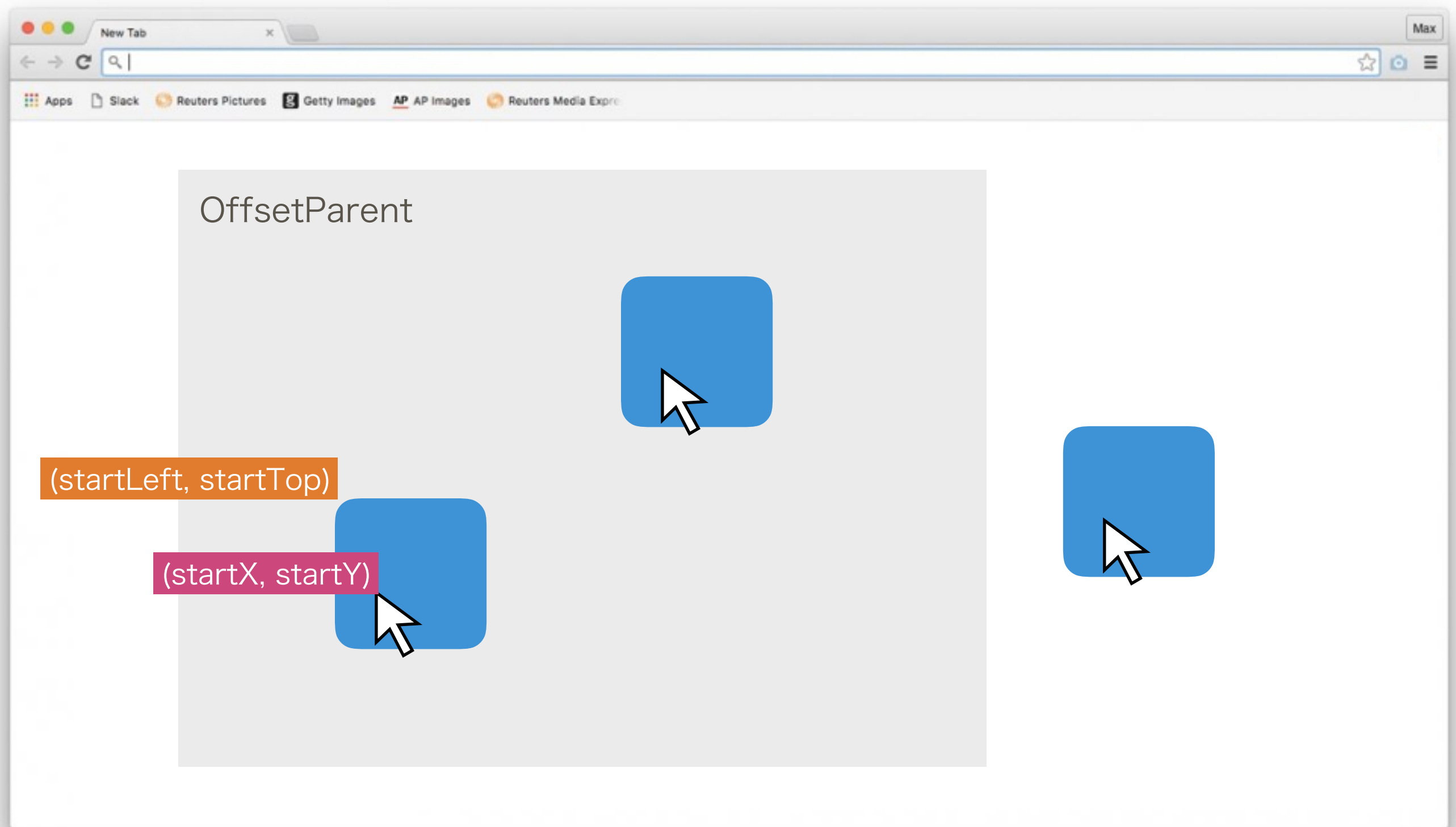




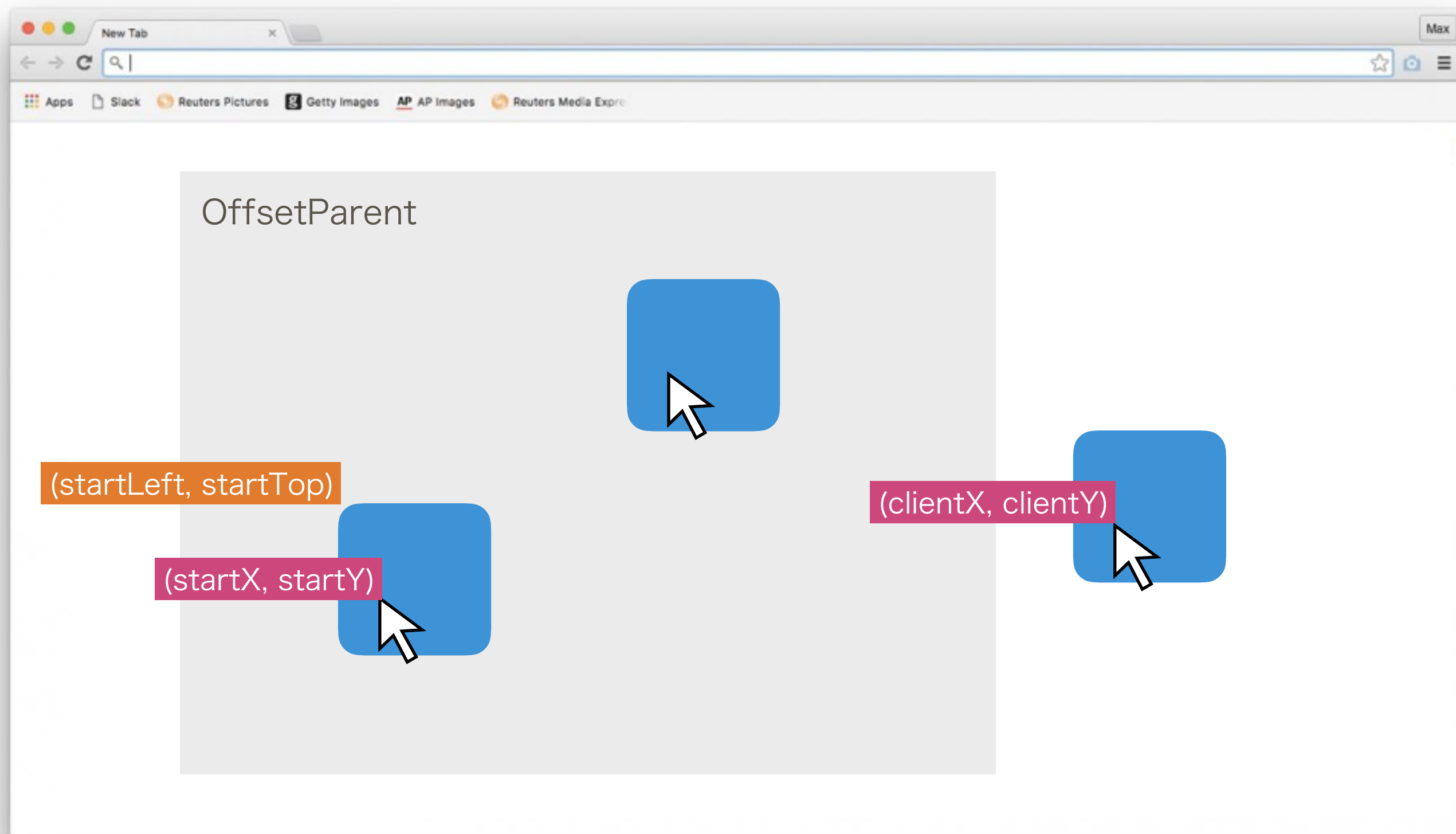




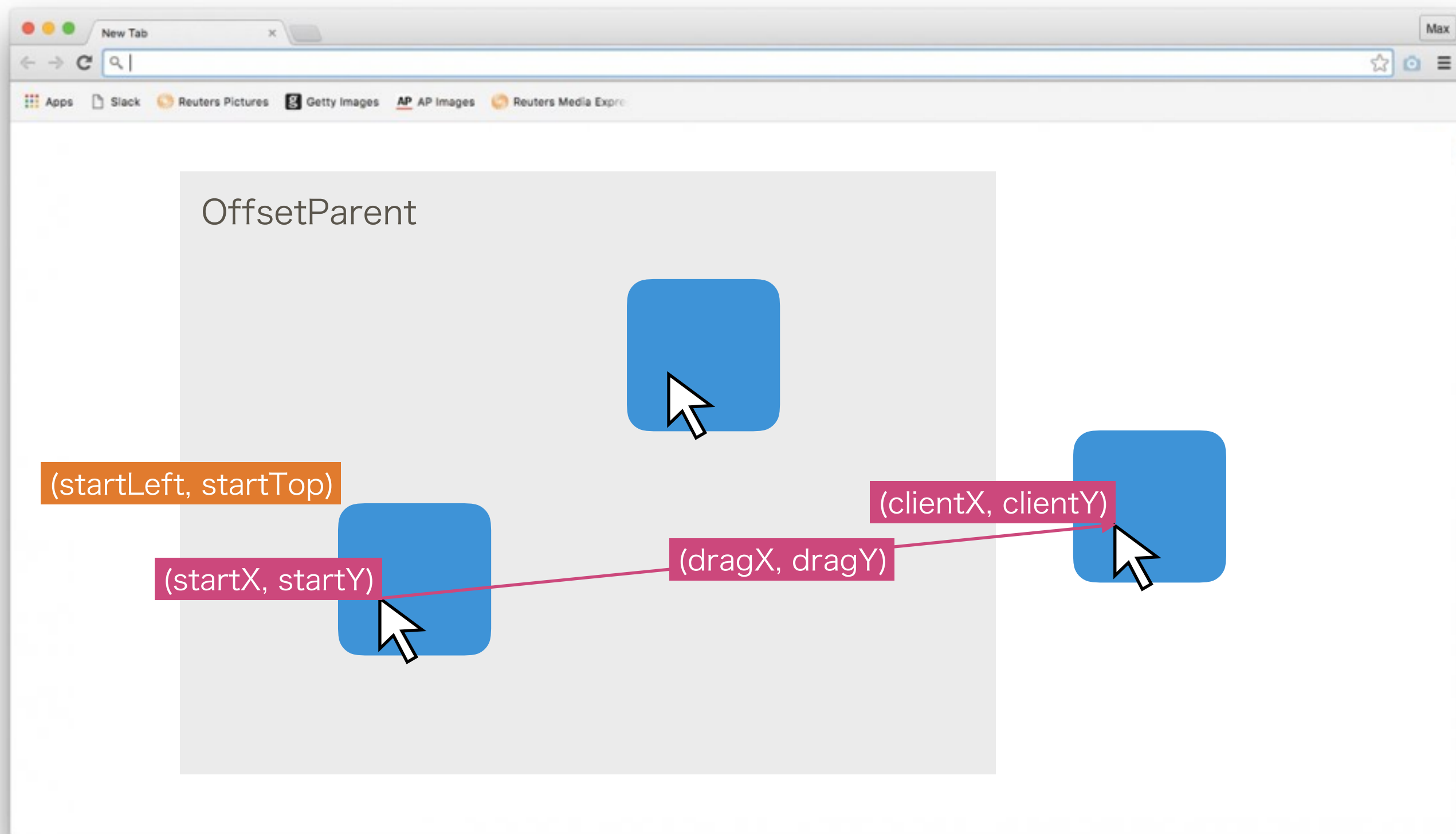




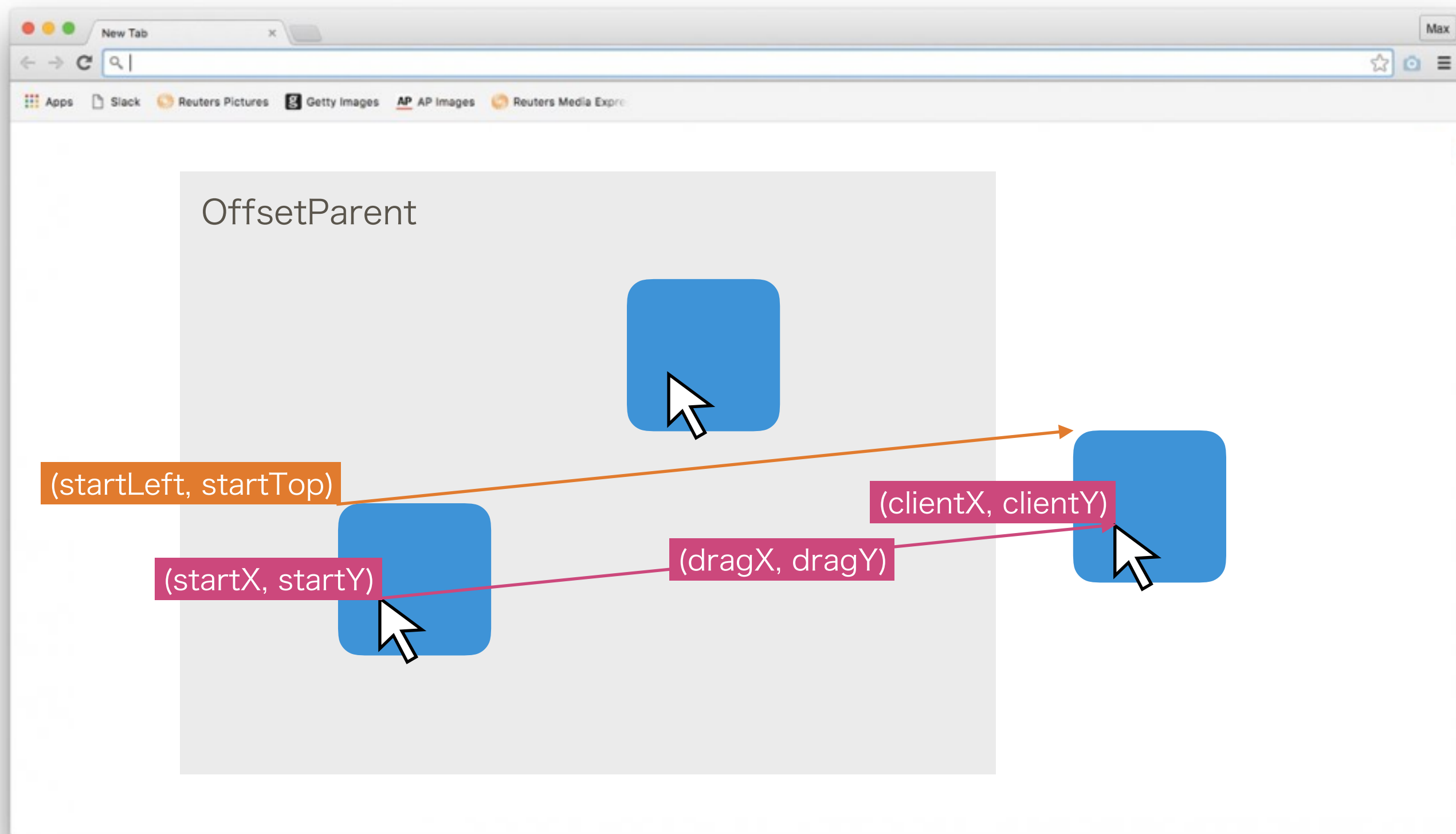




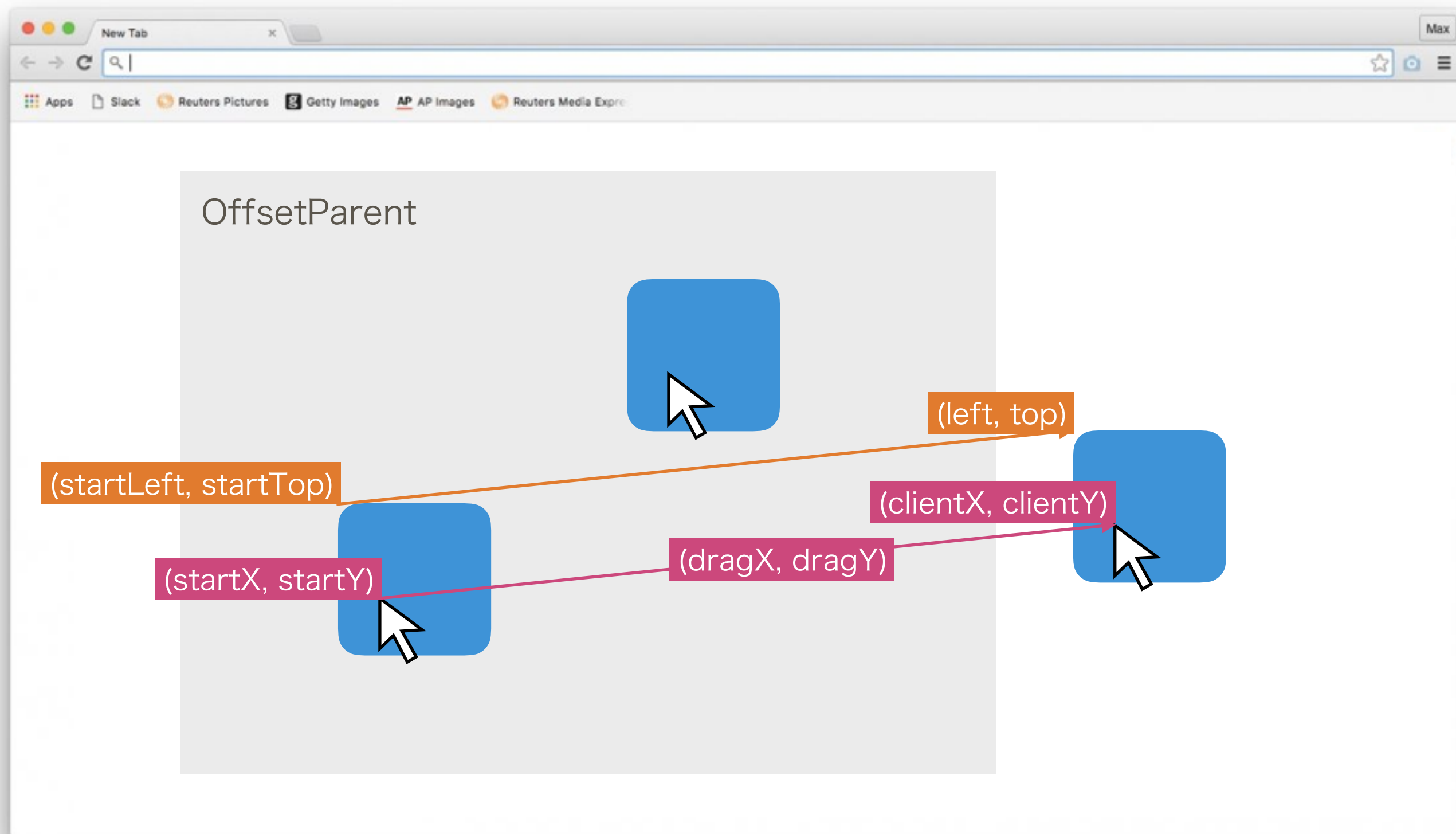














```
left = startLeft + dragX  
      = startLeft + clientX - startX  
top  = startTop + dragY  
      = startTop + clientY - startY
```

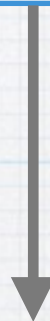


```
left = startLeft + dragX  
      = startLeft + clientX - startX  
top  = startTop + dragY  
      = startTop + clientY - startY
```

```
left = startLeft + dragX  
top  = startTop
```



```
left = startLeft + dragX  
      = startLeft + clientX - startX  
top = startTop + dragY  
     = startTop + clientY - startY
```



约束(restrict)

```
left = startLeft + dragX  
top = startTop
```



# 事件处理

- on-mousemoving（拖拽进行中）
  - 寻找并记录当前下方的目标元素
    - 抛出on-dragenter事件
    - 抛出on-dragleave事件
    - 抛出on-dragover事件



```
document.elementFromPoint(e.clientX, e.clientY);
```



```
document.elementFromPoint(e.clientX, e.clientY);
```

```
.z-dragProxy {  
    pointer-events: none;  
} /* IE11+ */
```



```
document.elementFromPoint(e.clientX, e.clientY);
```

```
.z-dragProxy {  
    pointer-events: none;  
} /* IE11+ */
```

```
proxy.style.display = 'none';  
var pointElement =  
document.elementFromPoint(e.clientX, e.clientY);  
proxy.style.display = '';
```



# 事件处理

- on-mouseup (拖拽结束)
  - 如果当前下方有目标元素
    - 抛出on-drop事件
  - 抛出on-dragend事件
  - 解绑window的mousemove和mouseup事件



# Regular实现方案



# 指令 vs 组件



# 指令 vs 组件

- 指令只能传一个参数，很受限制



# 指令 vs 组件

- 指令只能传一个参数，很受限制
- 指令无法继承



# 指令 vs 组件

- 指令只能传一个参数，很受限制
- 指令无法继承
- 在Regular的设计理念中，组件优先，指令功能弱化



# 指令 vs 组件

- 指令只能传一个参数，很受限制
- 指令无法继承
- 在Regular的设计理念中，组件优先，指令功能弱化

“因为组件概念的存在，regularjs中指令的作用被大大弱化（angular中将组件化与指令杂糅在了一起）”



Draggable



Draggable

MouseEvents

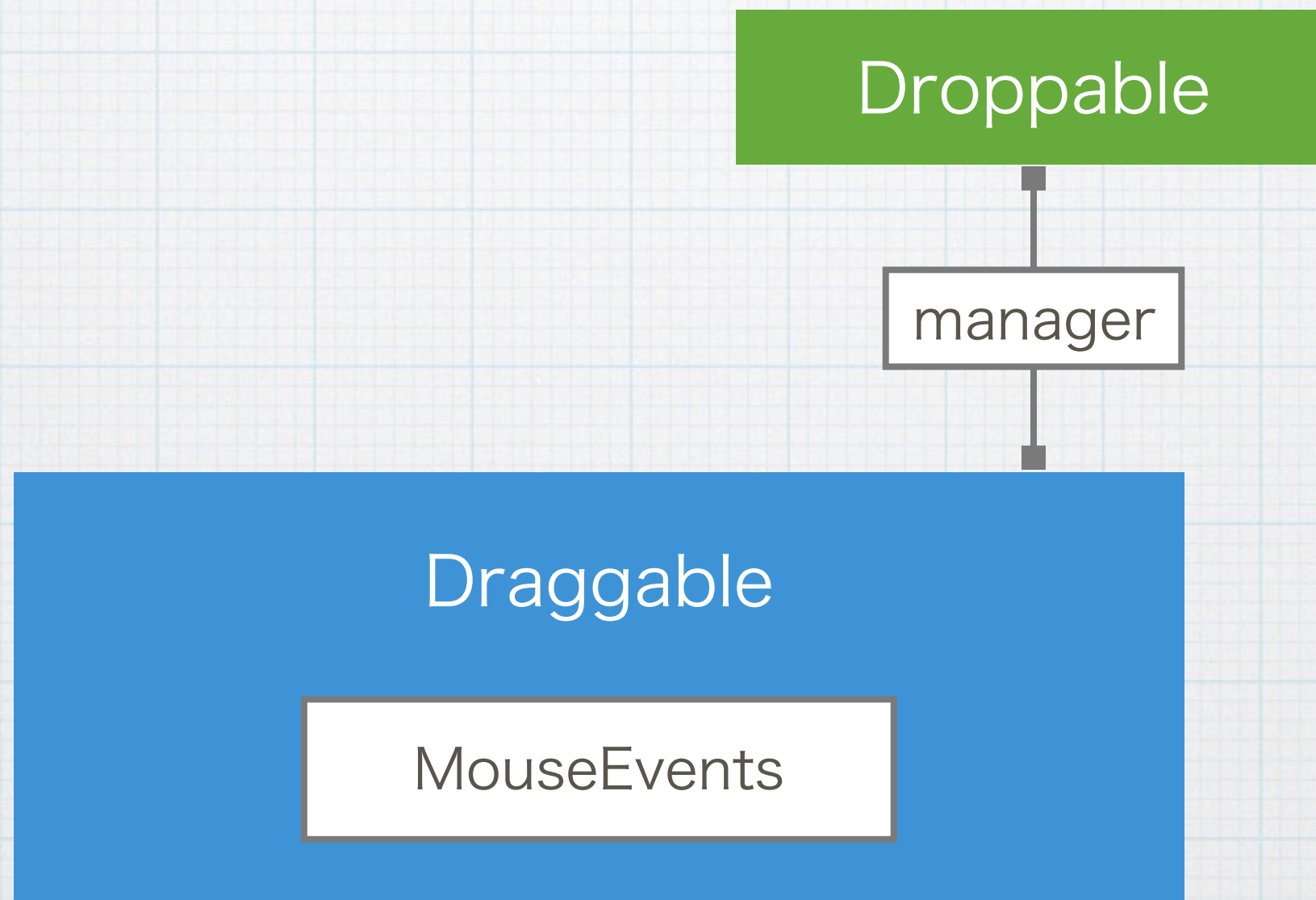


Draggable  
Droppable

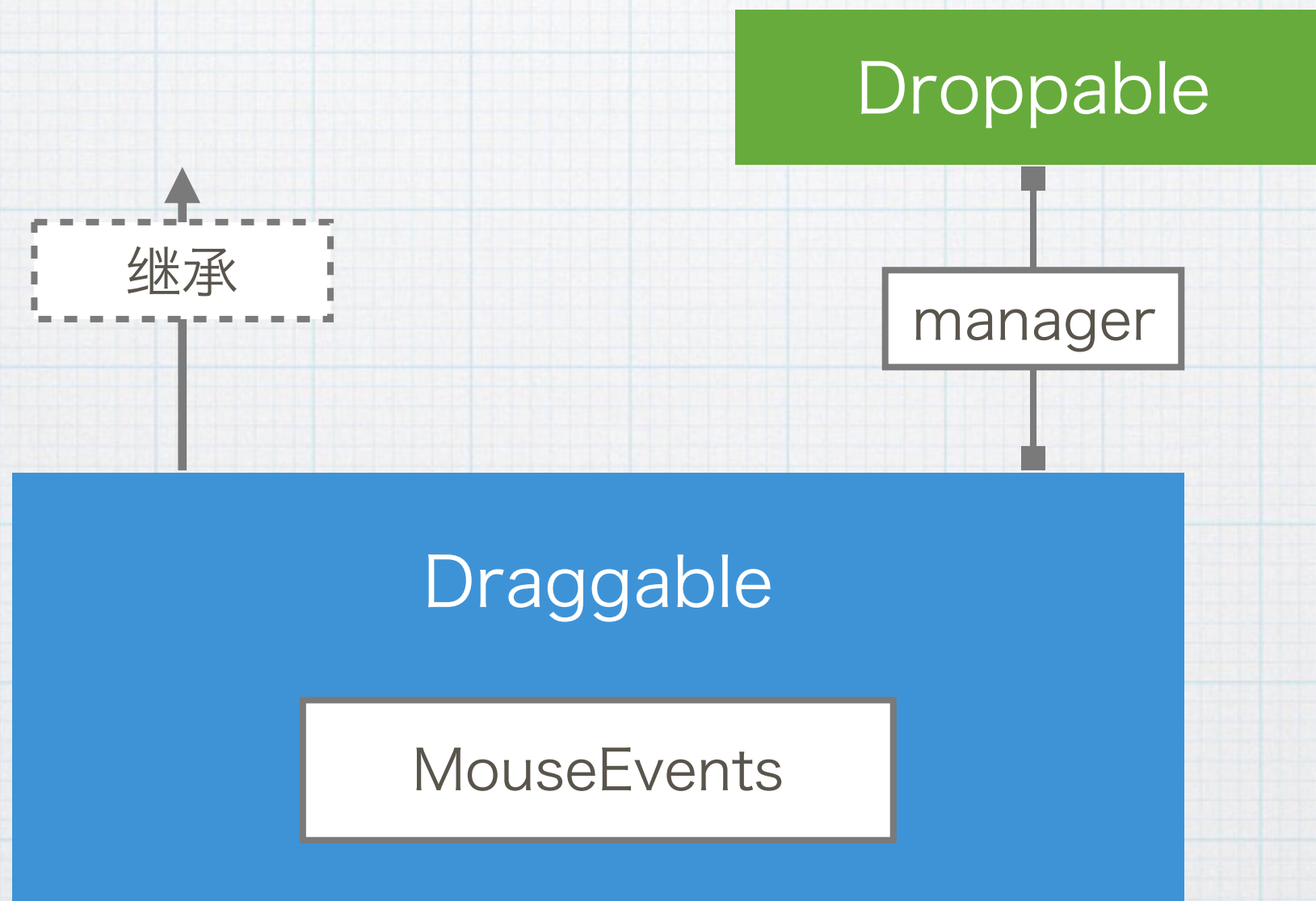
Draggable

MouseEvents

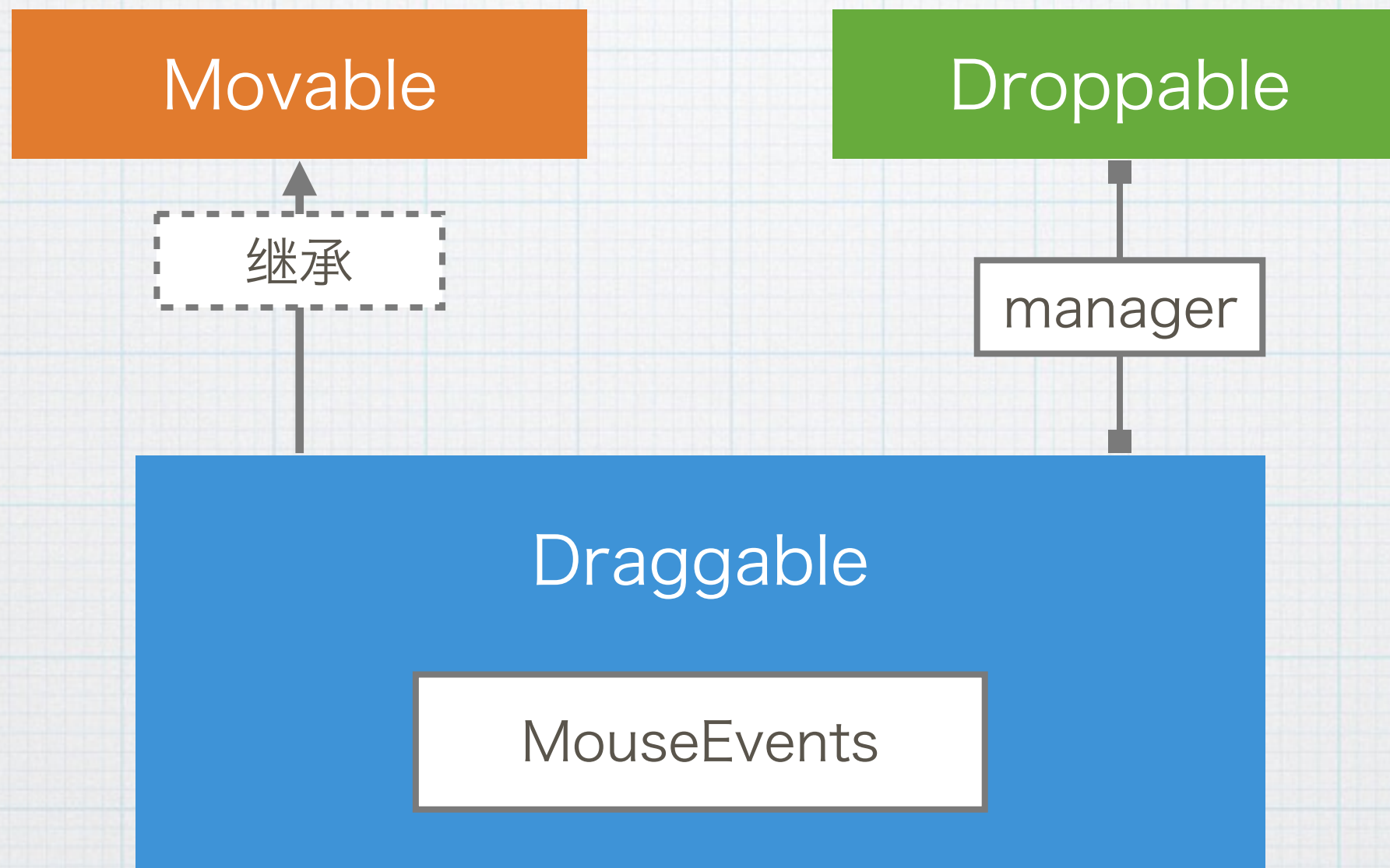




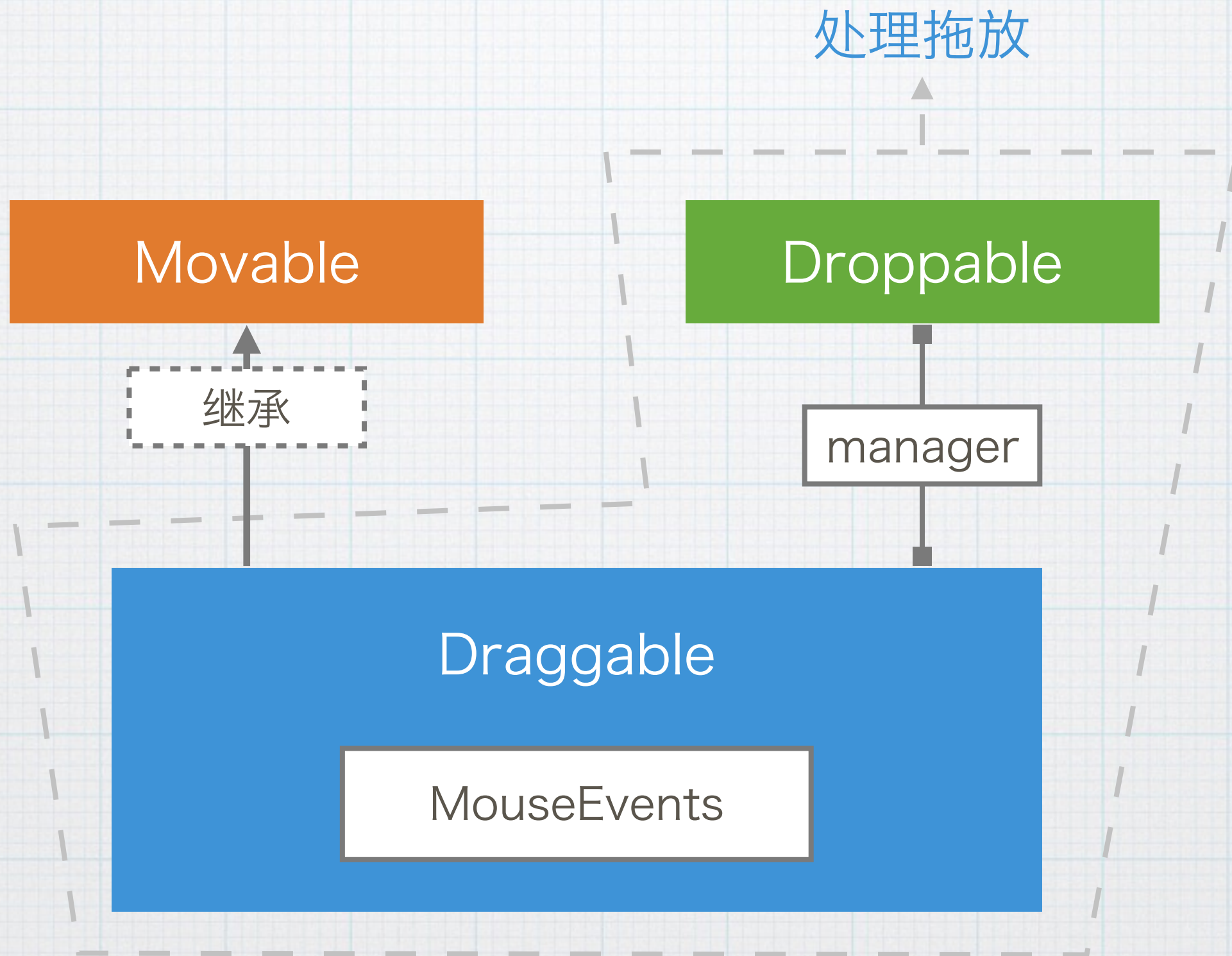




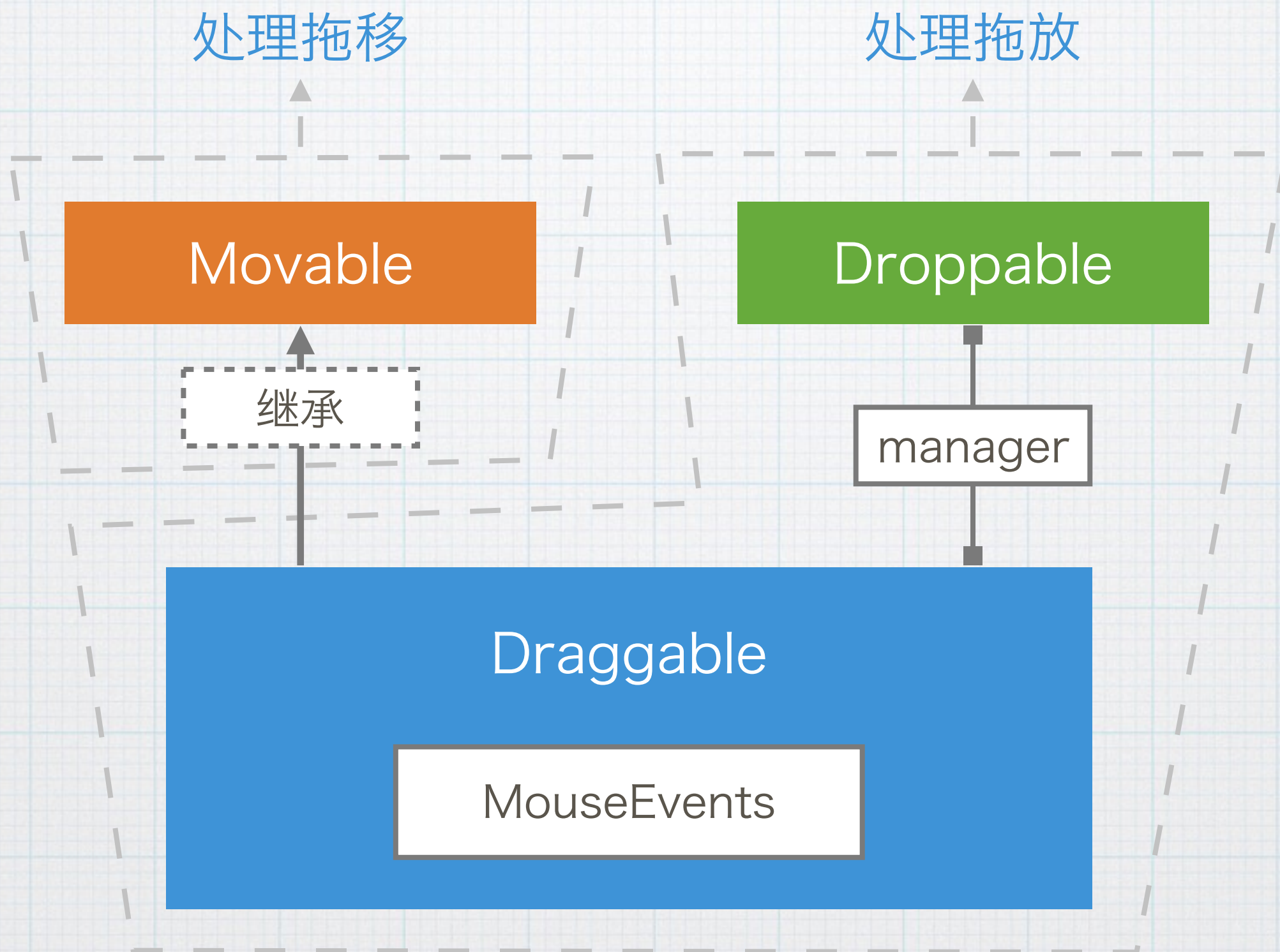














源码在此



# 使用案例

- ◎ 弹窗拖拽
- ◎ 拖拽约束
- ◎ 列表排序
- ◎ 网格排序
- ◎ Slider
- ◎ Paulette



# HTML5 vs 鼠标事件封装

HTML5	鼠标事件封装
只处理拖放	处理拖移和拖放
IE10以下效果不理想	可以兼容任意浏览器
使用时受限制	灵活
有一定的学习成本	研究成本高
与操作系统结合紧密 支持文件等操作	自成体系



# 总结







## ● 拖拽概述（拖拽的定义，拖拽的分类）



- ◎ 拖拽概述（拖拽的定义，拖拽的分类）
- ◎ 拖拽的基本流程（3个元素，7个事件）



- 拖拽概述（拖拽的定义，拖拽的分类）
- 拖拽的基本流程（3个元素，7个事件）
- HTML5（API，兼容性，拖放案例）



- 拖拽概述（拖拽的定义，拖拽的分类）
- 拖拽的基本流程（3个元素，7个事件）
- HTML5（API，兼容性，拖放案例）
- 鼠标事件封装（事件拆解和处理）



- 拖拽概述（拖拽的定义，拖拽的分类）
- 拖拽的基本流程（3个元素，7个事件）
- HTML5（API，兼容性，拖放案例）
- 鼠标事件封装（事件拆解和处理）
- Regular实现方案（源码，拖移案例，对比）



# Todos

- left&top vs transform
- autoScroll
- revert
- 与官方库相结合



# Todos

- ui-drag
- ui-sortable
- ui-resizable
- ui-slider
- ui-palette
- ui-splitter
- ui-window



# Thanks

Regular UI POPO交流群: 1319383