EventInterface类：

IEventSystemHandler：EventInterface里定义了和输入相关的接口，这些接口都继承IEventSystemHandler接口

继承IEventSystemHandler的接口及其作用：

1. IPointerEnterHandler，当指针（鼠标或触摸）进入对象区域时触发。
2. IPointerExitHandler，当指针离开对象区域时触发。
3. IPointerDownHandler，当在对象上按下指针时触发。
4. IPointerUpHandler，当指针释放时触发（不论之前是否在该对象上按下）。
5. IPointerClickHandler，当在对象上完成点击时触发。
6. IDragHandler，当在对象上拖动指针时持续触发。
7. IBeginDragHandler，拖动操作开始时触发一次（如记录开始位置方便复位）。
8. IEndDragHandler，拖动操作结束时触发一次。
9. IDropHandler，当拖动操作在其他对象上释放时触发（如进行交换UI）。
10. IScrollHandler，当鼠标滚轮滚动时触发。
11. IUpdateSelectedHandler，每帧对选中的对象调用.
12. ISelectHandler, 当对象被选中时触发（如通过键盘导航使得被选中UI高亮）。
13. IDeselectHandler，当对象取消选中时触发。
14. IMoveHandler，当移动事件发生时触发（如方向键输入）。
15. IInitializePotentialDragHandler，在拖动初始化时触发（在IBeginDragHandler之前）。
16. ISubmitHandler，当按下"提交"按钮（通常是键盘的Enter/Return键、游戏手柄的A/X按钮或触摸屏的确认操作）时触发
17. ICancelHandler，当按下"取消"按钮（通常是键盘的Esc键、游戏手柄的B按钮或移动设备的返回键）时触发

ExecuteEvents类：

负责执行事件，负责事件的分发（将事件从EventSystem的调度结果传递到具体的GameObject，并调用其对应的事件接口方法）。

UGUI里的事件的执行方式：指定某个接口类型，由Execute方法调用目标对象的接口方法。

EventInterface中定义的所有接口都会在ExecuteEvents中被调用。ExecuteEvents中声明了一个委托，定义事件触发时如何调用具体的事件处理方法，是 ExecuteEvents 类实现事件分发的关键工具。

文本

AI 生成的内容可能不正确。

Ps：这里委托的泛型参数T1，应该是EventInterface定义的所有接口类型

对EventInterface里的除IEventSystemHandler所有的接口声明了一个EventFunction类型的委托变量和方法，举个例子：

文本

AI 生成的内容可能不正确。

并且声明了相应属性，方便外部获取和调用只读的委托字段：

图形用户界面, 文本

AI 生成的内容可能不正确。

外部统一调用执行事件的方法：

文本

AI 生成的内容可能不正确。

在方法内部，通过GetEventList获得targetGameObject上的T类型的组件列表，然后遍历这些组件，并执行EventFunction<T>委托functor(arg, eventData)。

以pointerEnterHandler为例， functor这个方法实际上执行的是上面截图中声明的EventFunction类型的委托方法：

handler.OnPointerEnter(ValidateEventData<PointerEventData>(eventData));

也就是调用了IPointerEnterHandler类型的组件的OnPointerEnter方法。

Ps：

Execute<T>中用到的GetEventList<T>方法，以及关键注释

文本

AI 生成的内容可能不正确。

GetEventList<T>方法中用到的ShouldSendToComponent<T>方法及关键注释

文本

AI 生成的内容可能不正确。

ExecuteEvents中提供的其他方法：

ExecuteHierarchy<T>:通过GetEventChain获取target的所有父对象，并对这些对象（包括target）执行Execute方法

文本

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

GetEventHandler<T>方法：获取输入事件的实际响应对象

文本

AI 生成的内容可能不正确。