Plugin 类 所有的插件可以添加到编辑器中，它们是编辑器 state 的一部分，并且能够影响包含它的 state 和 view

new Plugin(spec: PluginSpec) 创建一个 plugin。

props: EditorProps 当前插件导出的 属性

spec: Object 当前插件的 配置对象。

getState(state: EditorState) → any 从编辑器的 state 上获取当前插件的 state。

PluginKey class

一个插件 key 用来标记一个插件，可通过搜索编辑器的 state 来找到它

new PluginKey(name: ?⁠string = "key") 新建一个 plugin key

get(state: EditorState) → ?⁠Plugin 用 key 从 state 获取到 key 对应的激活的插件。

getState(state: EditorState) → ?⁠any 从编辑器的 state 中获取插件的 state。

PluginSpec 插件配置对象

props: ? ⁠ EditorProps

通过插件的这个属性，添加到编辑器属性中。如果 props 是函数的话，则函数中的 this 将绑定到当前实例对象。

state: ? ⁠StateField <any>

允许插件定义一个 state 状态字段，编辑器整体 state 对象上配置一个额外的插槽，其插件可以拥有自己的状态

key: ? new PluginKey(name: ?⁠string = "key")

插件的标记，设置state 时也要指定一个 key 可以通过 key来访问指定插件 的配置和状态。

PluginKey 对象只有 2 个方法 get()获取插件 ， detState() 获取插件的状态

view: ?⁠fn(EditorView) → Object

设置与 DOM 或编辑器视图交互时使用此属性。 当插件的 state 与编辑器 view 有关联时将调该函数

returns: 返回一个对象，此对象有 2 个属性

update: ⁠fn(view: 编辑器视图, prevState:编辑器状态):编辑器 view更新时调该函数

注:更新可能是用户的操作或由事件触发的 transaction 更新视图,在此可获得编辑器 view 和更新 transaction 之前的 state

destroy: ?⁠fn() 当视图被销毁, 或接收不同插件的 state时调用

注: 页面重载等情况会销毁编辑器的 view

filterTransaction: fn(Transaction, EditorState) → bool

该函数在transaction 被应用到 state 之前调用，允许插件取消此次更新，通过 return false

appendTransaction: ⁠fn(transactions: [Transaction], oldState: EditorState, newState: EditorState) → ?⁠Transaction

追加一个transactions到 transactions 数组的末尾上去。 当其他插件追加一个 transaction 且在当前插件之后调用， 则当前插件的此函数会再一次调用。但

仅含最新 transaction 和state。它不会处理之前的 transaction