事件处理整体流程

- 编译阶段：处理为data中的on

```
(function anonymous() {
    with(this){return _c('div',{attrs:{"id":"demo"}},[
        _c('h1',[_v("事件处理机制")]),_v(" "),
        _c('p',{on:{"click":onClick}},[_v("this is p")]),_v(" "),
        _c('comp',{on:{"myclick":onMyClick}})
    ],1)}
})
```

- 初始化阶段：
  - 原生事件监听 platforms/web/runtime/modules/events.js
    - 事件也是作为属性处理
    - 整体流程：patch() => createElm() => invodeCreateHooks() => updateDOMListeners()
  - 自定义事件监听 initEvents **core/instance/events.js**
    - 整体流程：patch() => createElm() => createComponent() => hook.init() => createComponent... => _init() => initEvents() => updateComponentListeners()

> 事件监听和派发者均是组件实例，自定义组件中一定伴随着原生事件的监听与处理

调试验证

相关API

- $on() 监听自定义事件
- $emit() 派发自定义事件
- $once() 仅监听一次
- $off() 取消事件监听

彩蛋

在Vue当中，hooks可以作为一种event，在Vue的源码当中，称之为hookEvent。

```
<Table @hook:updated="handleTableUpdated"></Table>
```

场景：有一个来自第三方的复杂表格组件，表格进行数据更新的时候渲染时间需要1s，由于渲染时间较长，为了更好的用户体验，我希望在表格进行更新时显示一个loading动画。修改源码这个方案很不优雅。

callHook **src\core\instance\lifecycle.js**

```
export function callHook (vm: Component, hook: string) {
  // ...

  // 若包含hook事件，则一并派发
  if (vm._hasHookEvent) {
    vm.$emit('hook:' + hook)
  }
}
```

$on src\core\instance\events.js

```
const hookRE = /^hook:/
Vue.prototype.$on = function (event: string | Array<string>, fn: Function):
Component {
    // 若存在hook事件则添加标记
    if (hookRE.test(event)) {
        vm._hasHookEvent = true
    }
}
```

Vue双向绑定实现机制

v-model是mvvm类型框架中重要功能，给人印象最深刻的就是它，它是怎么实现的？且看下面分解：

测试代码

07-vmodel.html

编译阶段处理

v-model指令编译处理

输出的render函数

```
// 生成的渲染函数
(function anonymous() {
    with(this){return _c('div',{attrs:{"id":"demo"}},[
        _c('h1',[_v("双向绑定机制")]),_v(" "),
        _c('input',{directives:[{name:"model",rawName:"v-model",value:
(foo),expression:"foo"}],attrs:{"type":"text"},domProps:{"value":(foo)},on:
{"input":function($event)
{if($event.target.composing)return;foo=$event.target.value}})},_v(" "),
```

```
        _c('comp',{model:{value:(foo),callback:function ($$v)
{foo=$$v},expression:"foo"}})
    ],1)}
})

// input
_c('input',{
    directives:[{
        name:"model",
        rawName:"v-model",
        value:(foo),
        expression:"foo"}],
    attrs:{"type":"text"},
    domProps:{"value":(foo)},
    on:{
        "input":function($event){
            if($event.target.composing) return;
            foo=$event.target.value
        }
    }
})
// comp
_c('comp',{
    model:{
        value:(foo),
        callback:function ($$v) {foo=$$v},
        expression:"foo"
    }
})
```

初始化阶段：对节点赋值及事件监听

对节点赋值 platforms\web\runtime\modules\dom-props.js

事件监听 platforms\web\runtime\modules\events.js

额外的model指令 platforms\web\runtime\directives\model.js

自定义组件会转换为属性和事件 core/vdom/create-component.js