Trick - Delayed Evaluation

- 如果控件拖动，需要更新非常耗时的运算，那么每次拖动小步都会计算从而让拖动的用户体验非常卡。
- 解决方法是，合并拖动的行为，让用户拖动到目标位置，用户停下来不再拖动之后，再进行耗时的计算

- on_change () {
  - cleanTimeout($saved_timeout)
  - $saved_timeout = setTimeout (->() { very heavy calculation }, 1000)
- } # 用户1000ms没有动这个控件，就开始做耗时计算

Customized Widgets - Timer

```
class Timer
    def self.gen_html(option)
        timer_jsrb = '''
        $$[:setInterval].call( -> {
            var_set("timer",Time.now.to_s)
            trigger_on_change("timer")
        },1000)
        '''

        RenderWrap.before_jsrb("timer.javascript",timer_jsrb)

        widget_id = OpalBinding.binding(option[:binding],nil,self)
        "<span id='#{widget_id}'></span>"
    end

    def self.update_change
        "inner_html="
    end

    def self.fetch_change
        "inner_html"
    end

    def self.change_event
        ""
    end
end

def timer(option)
    Timer.gen_html(option)
end
```

```
    RenderWrap.load(Task.load("#{$task.name}::Timer"))
    RenderWrap.load(Task.load("#{$task.name}::timer"))

    RenderWrap.html =
'''
    <h4>Timer</h4>
    Current Time: <%= timer binding: :timer %><br/>
    counter: <%= text binding: :counter %>
    <%= var :counter, 1 %>

    <%= on_change :timer, %(
        :counter = :counter + 1
        mark_dirty("counter")
    )%>
'''
```