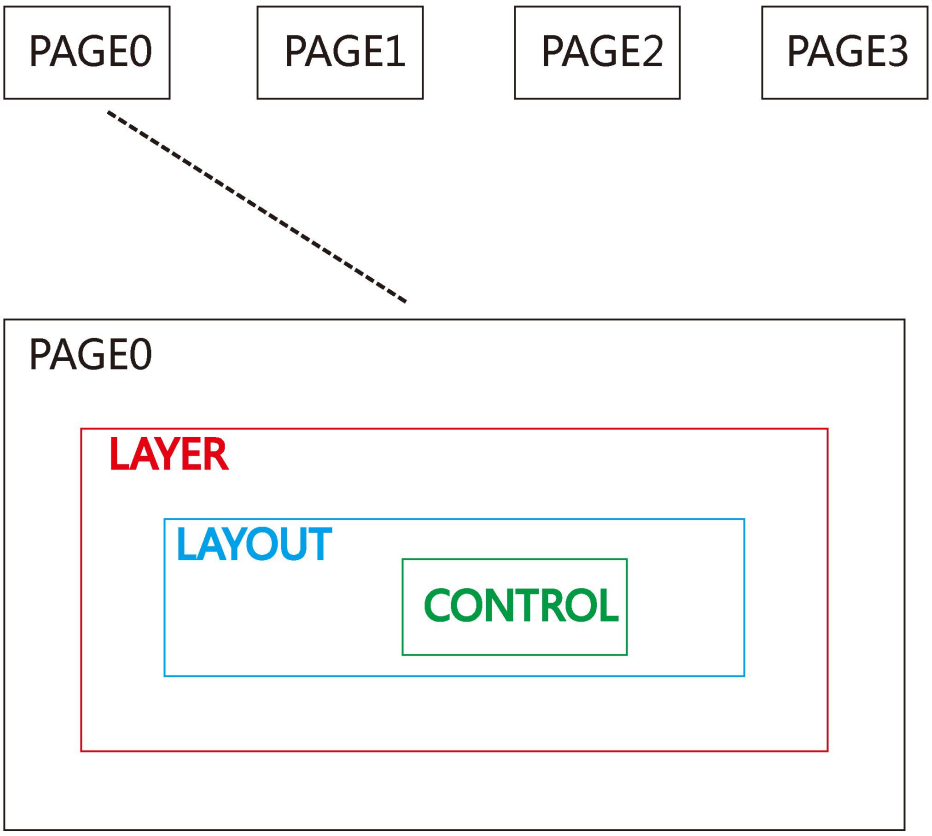
UI 框架

一、UI 集合的组成结构

UI 主要由不同的 PAGE 页面组成，PAGE 为最顶层单元，下层依次为 LAYER 图层、LAYOUT 布局、CONTROL 控件，如下图示例。一个 PAGE 下可以有多个 LAYER，LAYER 内可以有多个 LAYOUT，LAYOUT 内也可以有多个 CONTROL。

一般来说，一个 APP 模式使用一个 PAGE 来显示，那么不同的模式的显示切换只需要切换 PAGE 即可。



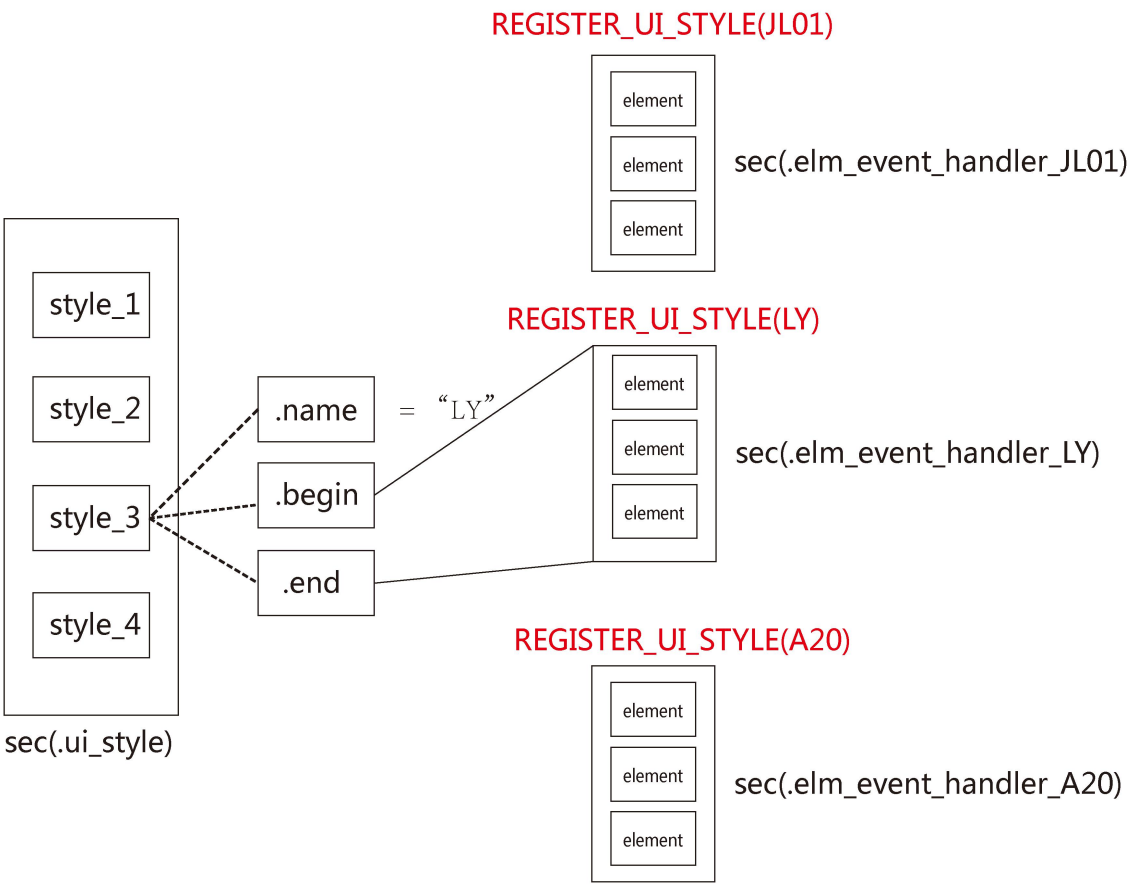
二、UI 风格与框架

为了方便用户开发不同的方案，我们使用 UI 风格进行管理，当 APP 代码不变，而只是想改变 UI 时，可以直接切换不同的 UI 风格。

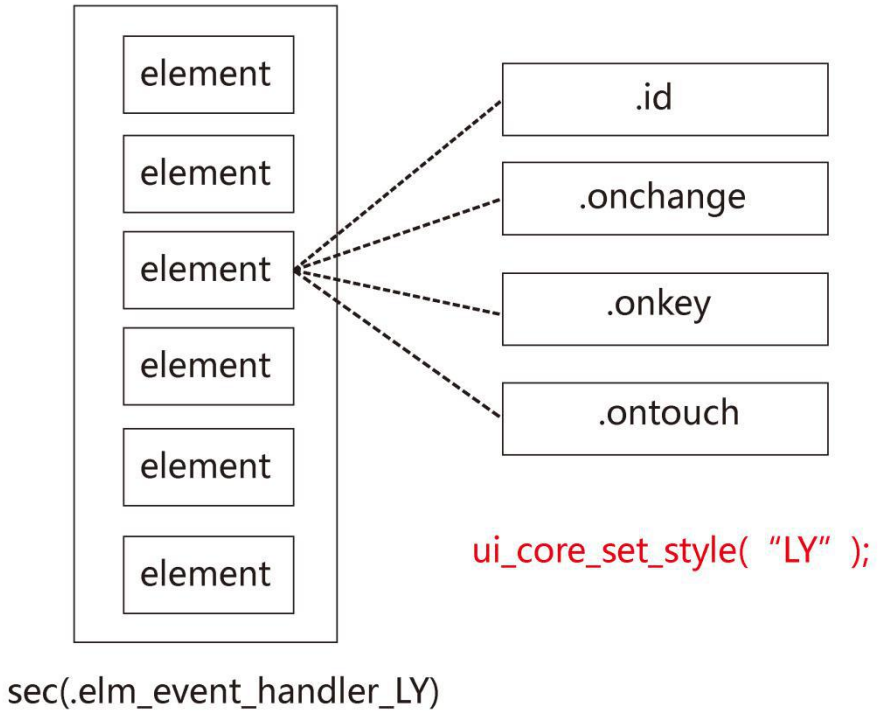
UI 风格的存放结构如下图，我们把不同的风格指针存放到 sec(.ui_style)段内，每个 style 都有三个成员，name 名字、begin 数据区起始地址、end 数据区结束地址。这样，代码就可以根据风格名字而寻找到数据区。

数据区内存放的都是该 UI 风格的 element 单元回调，之后代码会根据单元 ID 来寻找该区域内的 element，调用其回调。

比如下图有多个风格，名为"LY"的 UI 风格数据区域由 begin 和 end 确定，其指向数据段 sec(.elm_event_handler_LY)，数据区内为各个被注册的 element 回调。要注册该风格，需要在对应的.C 文件调用 REGISTER_UI_STYLE(LY)，即可在 sec(.ui_style)生成对应风格，且生成对应的 element 回调存放数据区。



Element 回调用于在显示的时候提供给用户的钩子函数，每个 element 有三个回调函数，分别为.onchange，.onkey，和.ontouch，而 element 也具有唯一的 ID 号.id。在注册了 UI 风格和编写了回调代码后，要使用该风格，还需要调用 ui_core_set_style(“LY”)来使用“LY”风格。如下图为“LY”风格 UI 的回调数据区示意图。



三、UI 单元之间的关系

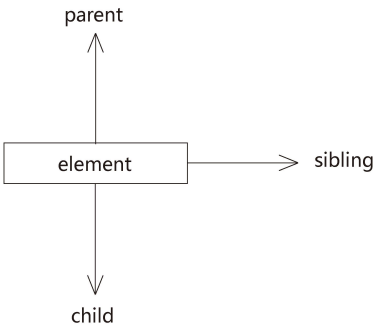
不同的 UI 单元之间具有连接关系，一个 element 单元具有三个指针，分别为 parent、child 和 sibling。

Parent 指向此单元的父控件，例如一个 LAYOUT 的 parent 指针指向其 LAYER。

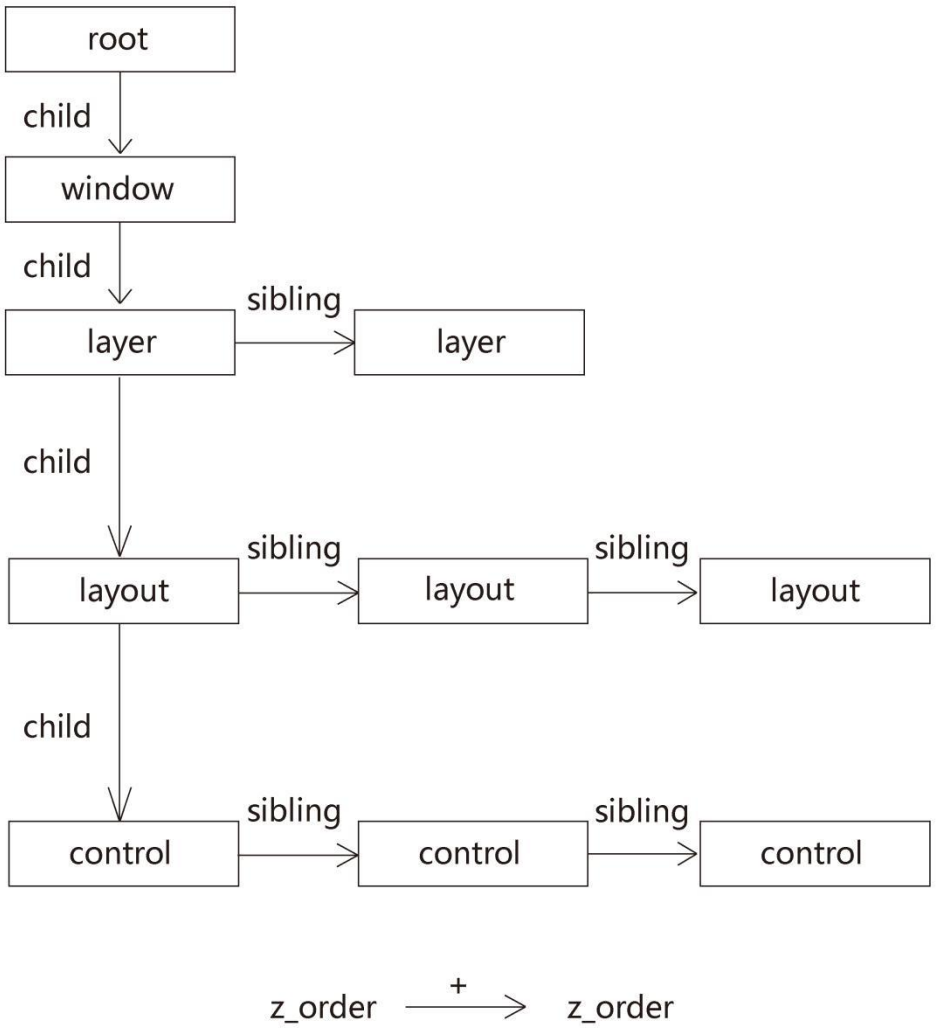
Child 指向此单元的子控件，例如一个 LAYER 的 child 指针指向其下的 LAYOUT。

Sibling 指向此单元的兄弟控件，例如一个 LAYOUT 的 sibling 指针指向其属于同一个父控件的另一个 LAYOUT。

Element 单元的三个指针如下图所示：

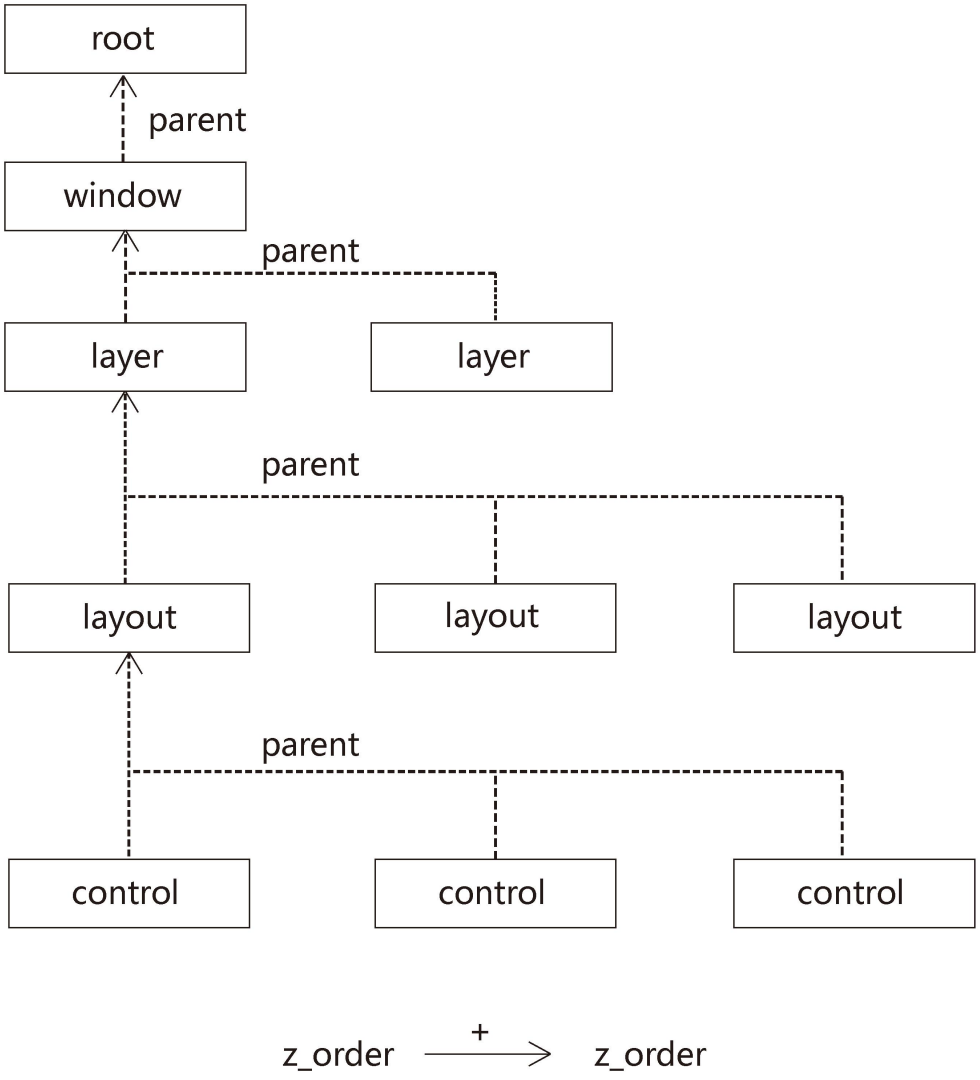


一个完整的 UI 关系正向链表如下图所示：



链表的最顶层为一个名为 root 的单元，下一个节点为 window（也就是 PAGE），window 下为图层 layer，layer 下为多个子控件 layout，layout 下又有多个子控件 control。而兄弟成员的节点连接方式是根据 z_order 变量由小到大排序的，z_order 为 UI 工程中该控件的层号，层号 0 即为最底层，层号越大意味着该控件的显示位置在越上层。有了 z_order 变量，UI 内核在显示一个单元的时候就能知道其上下位置，在显示了该单元后，内核会将其上层已经显示出来的单元重新显示，因而保持了 UI 单元的相互覆盖关系，单元的显示不会受到其它单元的影响。

一个完整的 UI 关系反向链表如下图所示：

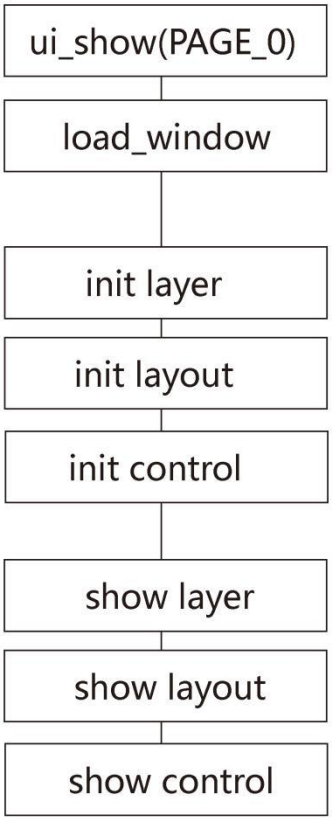


四、UI 显示的流程

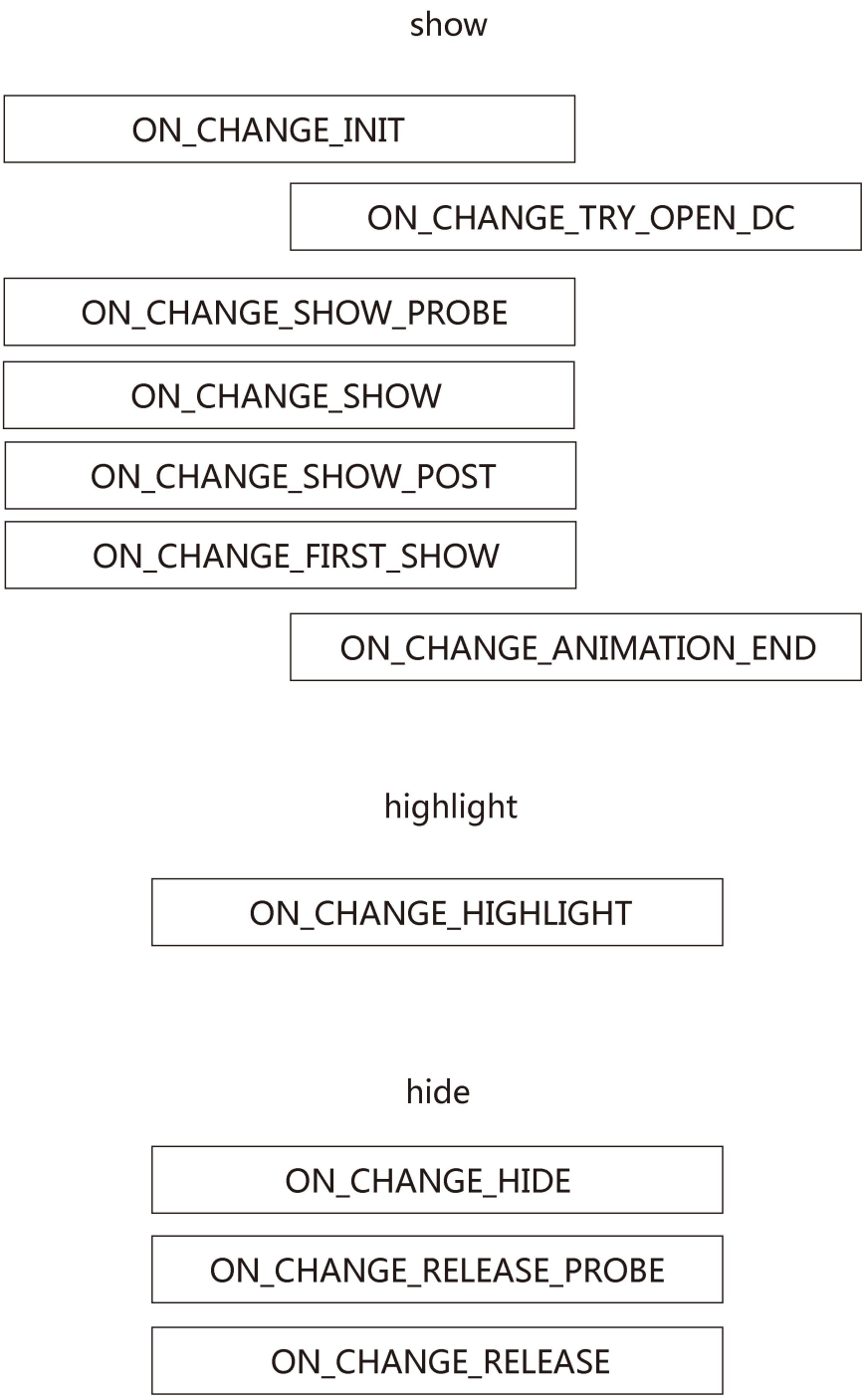
当我们要显示一个页面时，只需要调用 ui_show(PAGE)即可显示该页面的所有内容，页面内的控件有一个隐藏属性，该属性决定了此时是否会被显示出来，如果该属性使能，那么该控件就不会跟随其父控件而显示出来，必须单独调用显示函数来显示该控件。

页面的显示运行流程如下图，比如这里显示 PAGE_0，那么只需调用 ui_show(PAGE_0)即可。此时内核会调用 load_window 而从 FLASH 读取页面数据，然后根据数据初始化 layer、layout、control，初始化完成后会在显存绘制所有单元。

The process of showing a page:



当调用了显示函数后，UI 内核会在显示每个单元时，根据单元 ID 号寻找其 UI 风格的 element 回调区域，在不同的情况下调用单元的.onchange 回调。根据不同的情况，内核输入到.onchange 回调的参数不一样，用户可以在该回调内根据不同参数而进行不同处理。.onchange 的输入参数如下图所示：



在 show 的时候，使用的 onchange 回调参数如上图，在单元初始化时，会使用 ON_CHANGE_INIT，如果是 LAYER 单元，则会使用 ON_CHANGE_TRY_OPEN_DC。在绘制单元前，使用 ON_CHANGE_SHOW_PROBE，紧接着是 ON_CHANGE_SHOW，这两个参数并无多大区别，一般我们只解析 ON_CHANGE_SHOW_PROBE 即可。在绘制完单元后，内核会使用 ON_CHANGE_SHOW_POST，如果是在页面显示后的第一次绘制该单元，会紧接着使用 ON_CHANGE_FIRST_SHOW。ON_CHANGE_ANIMATION_END 是在动画控件播放完成的时候使用的。

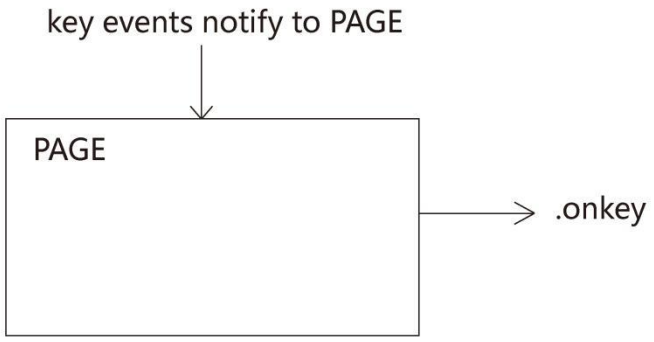
在调用了 highlight 函数来高亮某个单元时，内核会使用 ON_CHANGE_HIGHLIGHT 来调用 onchange 回调。

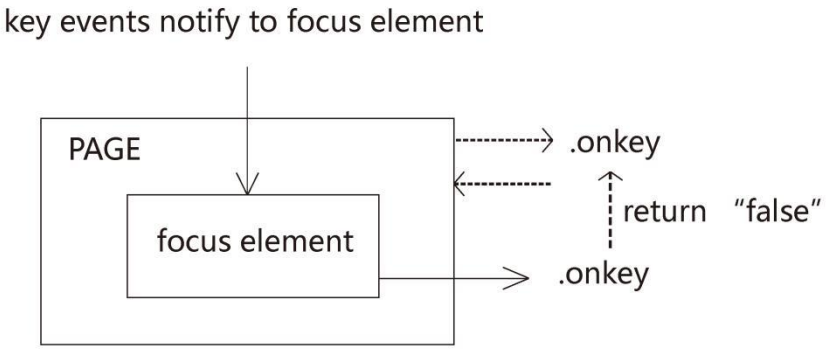
在 hide 的时候，使用的 onchange 回调参数如上图下部，被 hide 直接指定的单元会在隐藏前调用 ON_CHANGE_HIDE，而子控件不会调用该参数。接下来在释放前会调用 ON_CHANGE_RELEASE_PROBE，在释放时调用 ON_CHANGE_RELEASE。

五、UI 的事件处理

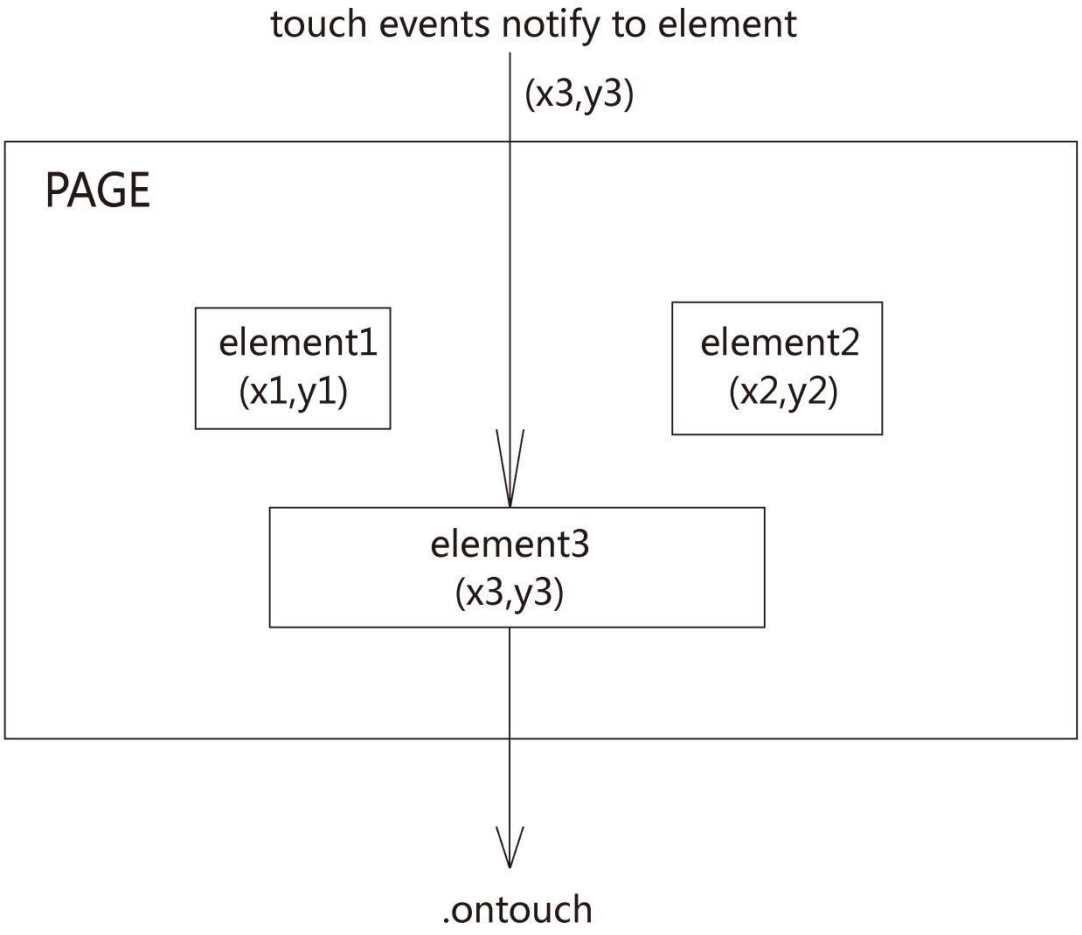
UI 的事件有两个，一个为 key 事件，另一个为 touch 事件，它们分别由函数 window_onkey 和函数 window_ontouch 来从页面开始分发到各个单元，由单元自己处理该消息。

Key 事件的发送过程如下图所示，一般情况下，key 事件只发送到 PAGE，由 PAGE 的.onkey 回调进行处理。而当显示了列表控件时，列表控件会将 key 事件 onfocus 到自己身上，此时 key 事件不再发到 PAGE 而直接先发到该列表控件，如果列表控件的 onkey 返回值为 true 则结束事件分发，如果返回值为 false 则继续将 key 事件重新分发给其父控件，一直到 PAGE 结束。





Touch 事件的发送则不同于 key 事件的发送，touch 事件为触摸屏的事件，该事件会有一个坐标参数（x,y），内核会根据该事件的发生坐标，寻找在 PAGE 中包含该坐标的控件单元，由该单元来接收该 touch 事件，进而调用其.ontouch 回调。如下图：



六、API 接口函数

1. 一般接口

int ui_show(int id);

说明：该函数为显示函数，任何控件都能够调用此函数来显示，输入参数为控件的唯一 ID 号。

int ui_hide(int id);

说明：该函数为隐藏函数，任何控件都能够调用此函数来隐藏，输入参数为控件的唯一 ID 号。

int ui_highlight_element_by_id(int id);

说明：该函数为高亮函数，任何控件都能够调用此函数来进行高亮显示，高亮显示的属性需要在 UI 工具内创建该控件的 CSS_1 属性，该函数的输入参数为控件的唯一 ID 号。

int ui_no_highlight_element_by_id(int id);

说明：该函数为取消高亮函数，任何控件都能够调用此函数来进行取消高亮显示，取消高亮显示的属性为 UI 工具内该控件的 CSS_0 属性，该函数的输入参数为控件的唯一 ID 号。

2. 图片接口

int ui_pic_show_image_by_id(int id, int index);

说明：该函数为图片显示函数，图片控件有一个包含了多张图片的图片列表，如果要指定显示图片列表中的某张图片，则需要调用此函数，指定图片列表中图片的序号 index，序号从 0 开始递增，而 id 即为图片控件的唯一 ID 号。

int ui_pic_set_image_index(struct ui_pic *pic, int index);

说明：该函数为图片显示函数，一般在控件回调中调用，图片控件有一个包含了多张图片的图片列表，如果要指定显示图片列表中的某张图片，则需要调用此函数，指定图片列表中图片的序号 index，序号从 0 开始递增，而 pic 即为图片回调输入的单元结构体指针。

3. 文本接口

int ui_text_show_index_by_id(int id, int index);

说明：该函数为 strpic 类型文本控件的显示函数，index 为该文本控件的文本列表的索引号，

用于指定显示某条文本，而 id 即文本控件的唯一 ID 号。

int ui_text_set_index(struct ui_text *text, int index);

说明：该函数为 strpic 类型文本控件的文本索引设置函数，一般在控件回调中调用，该函数用于将该文本控件的文本设置为指定 index 号的文本，index 为该文本控件的文本列表的索引号，text 为控件回调输入的文本单元结构体指针。

int ui_text_set_str_by_id(int id, const char *format, const char *str);

说明：该函数为 ascii 类型文本控件的字符串设置函数 format 是文本类型，需填写为”ascii”，而 str 为字符串指针，指向需要显示的字符串，比如“Hello world!”，id 为即文本控件的唯一 ID 号。

int ui_text_set_str(struct ui_text *text, const char *format, const char *str);

说明：该函数为 ascii 类型文本控件的字符串设置函数，一般在控件回调中调用，format 是文本类型，需填写为”ascii”，而 str 为字符串指针，指向需要显示的字符串，比如“Hello world!”，text 为控件回调输入的单元结构体指针。

4. 列表接口

void grid_on_focus(struct ui_grid *grid);

说明：该函数为列表控件的 key 事件接管函数，意即调用了此函数之后，key 事件都先发送到该列表控件，目前内核在显示列表控件后，如果 onchange 回调没有注册或者回调返回 false，内核默认为该列表控件调用此函数，将 key 事件集中到该控件。grid 为列表单元结构体指针。

void grid_lose_focus(struct ui_grid *grid);

说明：该函数为列表控件的 key 事件取消接管函数，意即调用了此函数之后，key 事件都不发送到该列表控件，目前内核在隐藏列表控件后，如果 onchange 回调没有注册或者回调返回 false，内核默认为该列表控件调用此函数，将 key 事件重新发送到 PAGE。grid 为列表单元结构体指针。

5. 电池接口

void ui_battery_level_change(int persent, int incharge);

说明：该函数是当前页面电池控件的电量显示函数，用于改变电池控件的状态，persent 的输入范围是 0 到 100，内核会根据控件所选择的图片进行计算，符合百分比范围的图片将会被显示，而 incharge 参数则是充电状态，1 为充电，显示充电图标，0 为不充电，继续显示电量。

6. 时间接口

int ui_time_update_by_id(int id, struct utime *time);

说明：该函数是时间控件的时间更新函数，time 是一个有用户输入的时间指针，时间控件将会根据该参数显示对应时间，id 则为时间控件的唯一 ID 号。