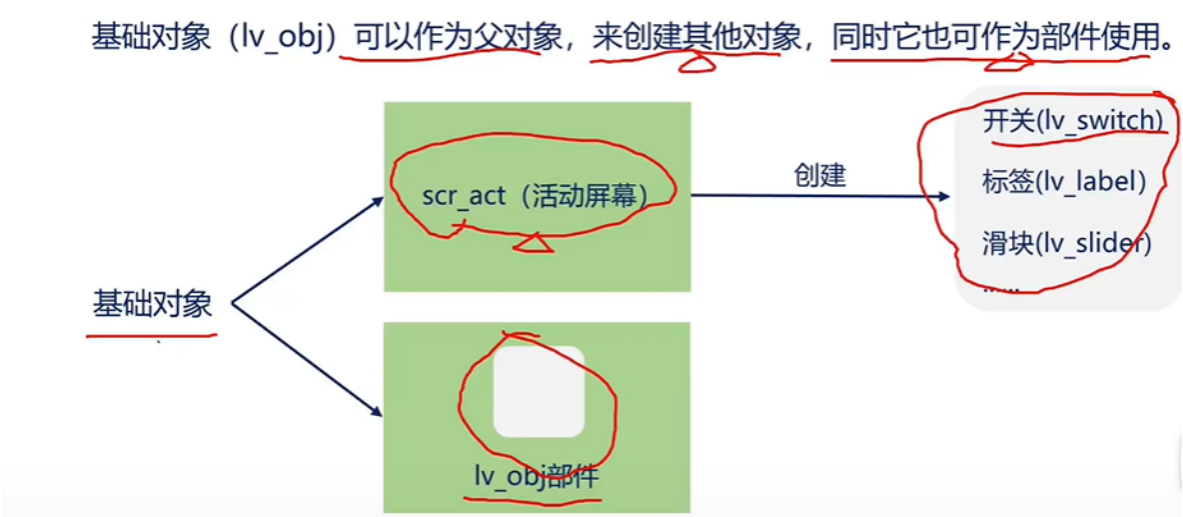
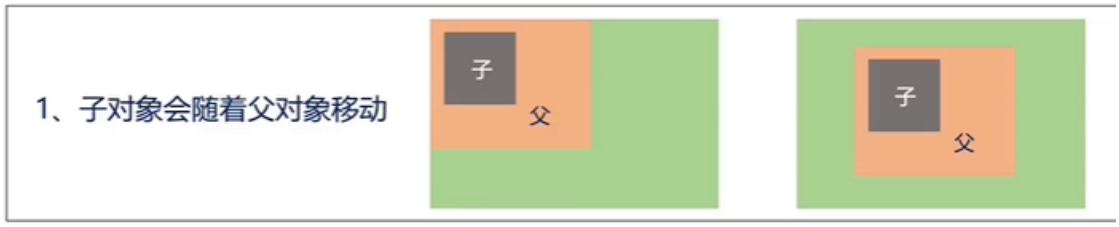
LVGL_STUDY

1. 基础对象简介：




```
lv_obj_t *name = lv_obj_creat(lv_scr_act());          //lv_obj是lv_scr_act的子对象
```

2. 父对象和子对象的关系



3. 部件的基本**属性**

- 大小

```
lv_obj_set_width(obj,new_width);
lv_obj_set_height(obj,new_height);
lv_obj_set_size(obj, new_width,new_height)
```

- 位置

设置部件位置时，坐标原点在父对象的左上角

```
lv_obj_set_x(obj, new_x);
lv_obj_set_y(obj, new_y);
lv_obj_set_pos(obj, new_x,new_y);
```

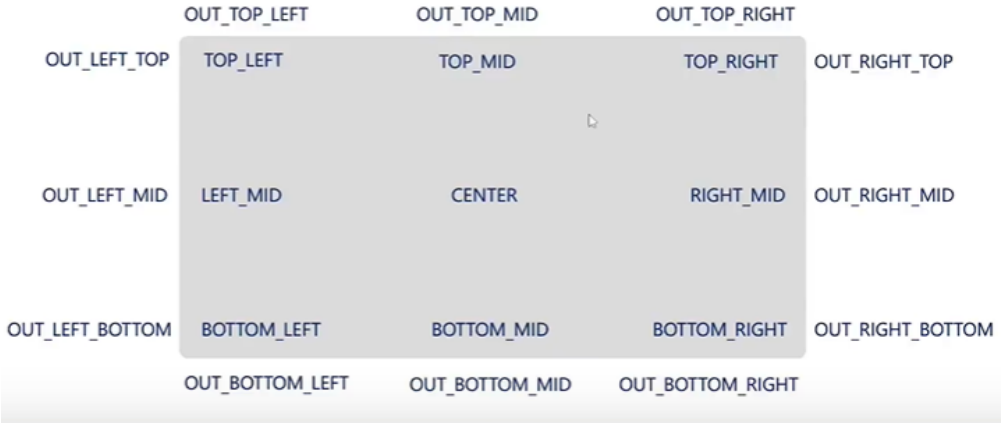
- 对齐

- 参照父对象对齐

```
lv_obj_set_align(obj, LV_ALIGN_...);     //参照父对象对齐
lv_obj_align(obj, LV_ALIGN_..., x, y);     //参照父对象对齐，再进行偏移
```

子对象能选择灰色框里面的9种对齐方式

对齐的模式



- 参照其他对象对齐（无父子关系）

```
lv_obj_align_to(obj_to_align,obj_referece,LV_ALIGN_...,x,y);
//obj_referece 参考对象
```

可以选择全部对齐方式，但是要注意父对象的范围

- 样式（用于设置部件的外观，以优化界面和实现用户的交互）
  - 添加普通样式

    特点：共用。它就类似于一个共用的样式套装，用户可以往里面

    添加所需要修改的样式内容（例如背景颜色、文本颜色等），当我们将这个样式套装应用到某

    个部件时，其所包含的样式内容将会被全部应用到该部件中。如果用户界面中有很多样式相同的部分，则建议使用此方法，这可以使样式设置变得非常高效。

```
 static lv_style_t style;
    lv_style_init(&style);                                  //初始化样式
    lv_style_set_bg_color(&style,lv_color_hex(0xf11ff));  //设置背景颜色

    lv_obj_t *obj = lv_obj_create(lv_scr_act());             //创建一个部件
    lv_obj_add_style(obj, &style,LV_STATE_DEFAULT);        //设置部件样式
```

  - 添加本地样式

    本地样式的特点是：设置简单，针对性强。当用户界面的对象样式有较大差异时，可以使

    用本地样式进行单独的设置

```
lv_obj_t *obj = lv_obj_create(lv_scr_act());       //创建一个部件
lv_obj_set_style_bg_color(obj,lv_color_hex(0xf11ff),LV_STATE_DEFAULT);
//设置部件样式
```

- 样式生效的状态

```
enum {
    LV_STATE_DEFAULT     = 0x0000,      /* 默认状态 */
    LV_STATE_CHECKED     = 0x0001,      /* 切换或选中状态 */
    LV_STATE_FOCUSED     = 0x0002,      /* 通过键盘、编码器聚焦或通过触摸板、鼠标单击 */
    LV_STATE_FOCUS_KEY   = 0x0004,      /* 通过键盘、编码器聚焦 */
    LV_STATE_EDITED      = 0x0008,      /* 由编码器编辑 */
    LV_STATE_HOVERED     = 0x0010,      /* 鼠标悬停（现在不支持）*/
    LV_STATE_PRESSED     = 0x0020,      /* 已按下 */
    LV_STATE_SCROLLED    = 0x0040,      /* 滚动状态 */
    LV_STATE_DISABLED    = 0x0080,      /* 禁用状态 */
    ...
};
```

- 样式属性

lv_obj_set_style_**xx_xx**(obj,lv_color_hex(0xf11ff),LV_STATE_XXXX); //


```
enum {
    LV_PART_MAIN          = 0x000000,       /* 主体，像矩形一样的背景 */
    LV_PART_SCROLLBAR     = 0x010000,       /* 滚动条 */
    LV_PART_INDICATOR     = 0x020000,       /* 指示器，指示当前值 */
    LV_PART_KNOB          = 0x030000,       /* 手柄或旋钮，用于调整参数值 */
    LV_PART_SELECTED      = 0x040000,       /* 选项框，指示当前选择的选项 */
    LV_PART_ITEMS         = 0x050000,       /* 相似的元素，例如单元格 */
    LV_PART_TICKS         = 0x060000,       /* 刻度 */
    LV_PART_CURSOR        = 0x070000,       /* 光标 */
};
```

lv_obj_set_style_xx_xx(obj,lv_color_hex(0xf11ff),**LV_PART_XXXX**); //

- 事件
  - 添加事件：

```
lv_obj_add_event_cb(obj,event_cb,event_code,user_data);
//event_cb事件回调函数、event_code事件类型、user_data用户数据
```

事件类型：

- 输入设备事件
- 绘图事件
- 其他活动
- 特别活动
- 自定义事件

  - 删除事件

```
lv_obj_remove_event_cb(obj,event_cb);
```

- 不同事件共用回调函数`lv_event_code_t code = lv_event_get_code(lv_event_t e)`

  通过回调函数`event_cb`的输入值获取触发事件或部件
- 不同部件共用回调函数`lv_obj_t *target = lv_event_get_target(lv_event_t e)`