iOS 开发一定要尝试的 Texture(ASDK)

其实也没有 关注
3 2018.09.04 10:46:03 字数 3,760 阅读 7,614

前言

本篇所涉及的性能问题我都将根据滑动的流畅性来评判, 包括掉帧情况和一些实际体验

ASDK 已经改名为 Texture, 我习惯称作 ASDK

image

编译环境: MacOS 10.13.3, Xcode 9.2

参与测试机型: iPhone 6 10.3.3, iPhone 7 11.2.1, iPhone X 11.2.5, 默认 iPhone 6

TableView / TableNode 包含的 TableViewCell / CellNode: 默认复杂程度一般, 包含 1~2 张图片和 2~4 条文本展示, 图片有圆角

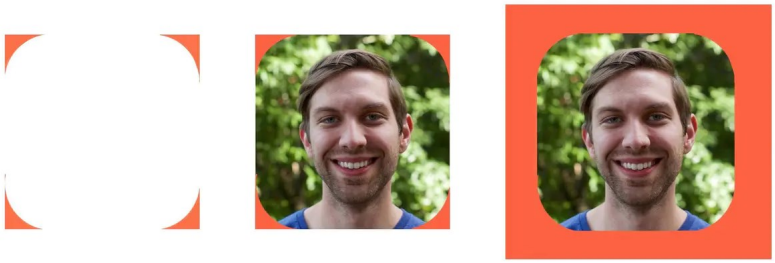
列表滑动卡顿的原因及优化

大牛们把原因都说的很清楚了, 导致的结果就是 16ms 不足以渲染一帧, 产生掉帧卡顿

下面是尝试过的一些优化:

圆角

- 使用一张圆角图片覆盖, 经典文章 Corner Rounding, HYBImageCliped 也是这么做的



image

- 异步裁剪图片: 通过 UIGraphics 对图片进行裁剪, 可能造成内存暴涨

行高缓存

老生长谈了, 除 UITableView-FDTemplateLayoutCell 之外, QMUI 中也有提供一套缓存行高方案

数据预加工

具体是在 JSON 转 Model 后把文本转为富文本, 处理一些弱逻辑等, 之后赋值就可以直接展示了

咳咳, 这个感觉不到什么效果

图形预加工

热门故事
凤凰男的醒悟:花了半辈子孝顺父母,换来妻离子散
扎心虐文:不是所有的女主最后都会选择原谅
闺蜜的一条朋友圈,结束了我和老公5年的婚姻
被戳穿的爱情是骗局,没被戳穿的变成了信仰

推荐阅读
转载:iOS页面保持流畅技巧
阅读 488
flutter解决重叠控件布局问题
阅读 214
Android-推荐一个智能刷新库SmartRefreshLayout
阅读 293
iOS自定义NavigationBar通顶的问题
阅读 308
官方还有这个控件?Android优雅实现小红点效果, so easy~
阅读 580

例如处理图片遮罩或固定的图标, 一般是直接使用多层视图实现

我曾尝试把三张小图绘制到一张大图上再进行展示, 于是乎, 异步复用问题除外, 内存炸了, 最终还是老老实实用多个视图实现

为什么要使用 ASDK

图形异步渲染

通常我们认为 UIKit 是不能渲染于非主线程的, 一旦你这么做, 就可能会导致崩溃, 无法正常显示等问题, 而 ASDK 为什么可以呢, 因为 ASDisplayNode 是线程安全的, Node 创建时, 不会立即在其内部新建 UIView 和 CALayer, 直到主线程第一次访问时才会生成对应的对象, 除此之外, 还通过图层预合成和基于 Runloop 的异步并发, 使其拥有更好的性能 ASAsyncTransactionGroup

这个特点带来的相关实际体验就是: 安心的进行异步绘图, 如圆角裁剪, 增加遮罩等, 这在 UIKit 中是足以毁灭人生的, 内存暴涨, 异步复用, 性能极差

不过低性能设备下还是会出现明显空白

<video src="https://img.didee.cn/video/ASDKDemo-Video.mp4" poster=https://img.didee.cn/video/post/ASDKDemo-Video.png!jpgwidth=100% controls="controls" preload="none" >Video

预加载数据和对象

首先来一张 Gif 体验一下, 实际上使用 ASDK 开发完成后对比也是如此, 有种网速变快了的错觉

ASDK 中的 ASRangeController, ASTableView, ASCollectionView 相对于 UIKit 原生控件的特点是可用于监控视图的可见区域, 维护工作区域, 在合适的时机触发网络请求以及绘制, 单元格的异步布局

[图片上传失败...(image-29e19e-1536029030806)]

这里推荐阅读: 预加载与智能预加载(iOS)

异于原生控件的复用机制

单一的 Cell

意思是某个 List 展示的样式只有一种, TableView 只需要注册一个 Cell

这种情况下, 如果常规的一些优化得当, 滚动的流畅性还是可以接受的(与 ASDK 差距微小, 但仍然肉眼可分辨)

此时的差距主要体现在列表某项数据第一次展示, 以及 TableView 在分页加载时产生的等待较长, 当然, 这两点也是可以继续优化和解决的

相反的, 也就是来回滑动已经展示过的数据, 两者的差距就非常小了, 大概是 59.7 - 59.9 和 59.9 的区别 (我瞎扯的)

综上, 优化得当的情况下, 单一的 Cell 情况下 UIKit 与 ASDK 的差距不明显

<video src="https://img.didee.cn/video/NiceFilm-Home.mp4" poster=https://img.didee.cn/video/post/NiceFilm-Home.png!jpg width=100% controls="controls" preload="none" >Video

多种 Cell

表示某 List 中有多种不同的样式, TableView 必须要通过注册 N 个 Cell 来实现

这种情况下, 假设有 5 种 Cell, 屏幕可同时展示 6 条 Cell, 此时若第一屏幕刚好展示的就包含全部 5 种 Cell , 那么后续的滑动情况将与单一的 Cell表现一致, 若第一屏幕展示的内容只包含一种, 其他 4 种没有在屏幕上出现过, 那么当某一种首次出现在屏幕上时, 便会出现明显的卡顿; 我尝试过解决这个问题, 提前创建所有的 Cell 实例对象, 缓存和复用相同的子视图, 异步预绘制为一张图片并缓存(坑), 都收效渐微

因 ASDK 支持预渲染, 与处理单种 Cell 没有区别, 依旧 59.9

复用的差别

TableView 的复用机制我是既爱又恨的, 方便之处在于直接与数据绑定后, 可以方便的更新和修改, 只需保证 setModel 简洁就 OK, 只是当业务绑定较多时就比较麻烦了

下面重点说说 TableNode, TableNode 的复用机制就是没有复用, 只有缓存, 每个 CellNode 都是全新的, 因此会有一些特殊的地方:

CellNode 与数据源没有绑定关系: 创建后就算把数据源删除, TableNode 依然可以正常展示

数据直接决定要创建一个怎样的 CellNode: 这一点很重要, TableViewCell 的展示大致为: 添加空假数据子视图 -> 数据填充 -> 刷新, 涉及布局或图文时会更复杂
CellNode 只有一步: 添加真数据的子视图; 因此可以直接根据业务逻辑对控件和布局做出处理, 而不用一次或多次刷新

Demo: 此处需求为每组一个大图 + N个小图, 每组 3 或 5 个

image

解决思路: TableView 的方式是创建 5 个, 根据数量显隐下面两个, 或者两种 Cell, 把 3 和 5 的情况分别对应, 除此之外, 最重要的是: 祈祷数据正常, 每组数据个数仅为 3 或 5

此时若使用 TableNode 就灵活多了, 可以根据需要(数据个数), 加入需要的子视图, 我的思路是把顶部的大图固定, 剩下的两个为一行进行添加, 就算总数为偶数也是没有任何额外消耗的, 具体参见 ASDKDemo

<video src="https://img.didee.cn/video/ASDKDemo-Muster.mp4"
poster=https://img.didee.cn/video/post/ASDKDemo-Muster.png!jpgwidth=100%
controls="controls" preload="none" >Video

Flex 布局

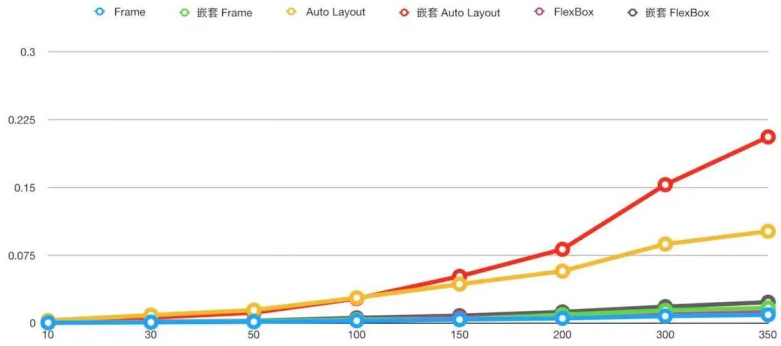
值得学习的理由

ASDK 使用的是 Flex 布局, 且面向对象

偷一张图

iOS 布局耗时测试

	Frame	嵌套 Frame	Auto Layout	嵌套 Auto Layout	FlexBox	嵌套 FlexBox
10	0.000306	0.000393	0.002951	0.001807	0.000397	0.000518
30	0.000820	0.001147	0.008795	0.006023	0.001170	0.001513
50	0.001379	0.002141	0.014375	0.011551	0.001929	0.002527
100	0.002556	0.003760	0.027948	0.027057	0.004102	0.005750
150	0.003861	0.003760	0.043013	0.051693	0.005785	0.008185
200	0.005210	0.009242	0.057437	0.081550	0.008314	0.012315
300	0.007708	0.014013	0.087354	0.153033	0.01178	0.018155
350	0.009187	0.016747	0.101465	0.205872	0.013384	0.023091



image

具体对比: iOS 上的 FlexBox 布局

简单来说, 缺点只有一个, 就是学习曲线相对 Frame AutoLayout 更陡峭, 而优点是 性能与 Frame 相当, 上手后比 AutoLayout 还简单, 如果你已经开始尝试, 请坚持下去

不同的方式和思想

AutoLayout

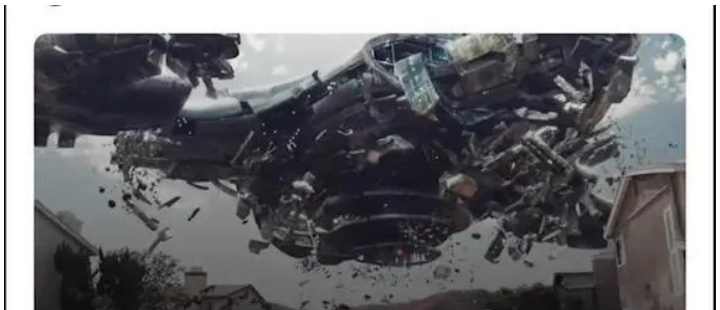
使用 AutoLayout 时我心里想的无外乎:

- 我要把你放在左上角: Left Top
- 把你放在它右边: LeftTo(它)
- 放中间: Center
- 至少/至多离它多远: less / greater

缺点是视图之间的依赖性太强, 可读性维护性较差(更差的是 Frame), 例如排列数个距离不等控件, 就会很厌烦, 然后 cv 重复代码; 处理多个多行文本垂直排列时很恶心, 想要处理好最终需要去计算文字行高, 外加入自定行间距; ...

Flex

例如 Demo 中的



image

我做的事情是:

- 声明大图的比例: Ratio(9.0/16.0); 那么这个声明存储为 `postImageRatioSpec`
- 声明大标题的内边距: Inset(8, 8, 8, 8); `titleInsetSpec`
- 声明 `titleInsetSpec` 的位置是垂直方向下的最尾部; `titleRelativeSpec`
- 声明 `titleRelativeSpec` 是覆盖到 `postImageRatioSpec` 上: `titleOverlaySpec`

此时大图和文字布局完成

image

接下来是用户栏:

- 声明用户头像和名称水平排列, 水平方向从左也就是从头部开始, 距离 4, 对齐方式为居中, 此时的居中为垂直方向; `leftStackSpec`
- 同理, 声明两个图标为水平, 尾部起始, 距离4, 居中; `rightStackSpec`
- 接下来, 声明 `leftStackSpec` 和 `rightStackSpec` 水平排列, 等间距排列填充(实际是为 left 和 rightStack 进行填充), 距离 8, 对齐方式填充(无实际作用, 由于子视图同为 Stack 且都是水平方向); `userStackSpec`

image

最后:

- 声明 `titleOverlaySpec` 和 `userStackSpec` 垂直排列, 自上而下, 对齐方式填充(同理于 `userStackSpec`, 此处影响的是 `userStackSpec`); `videoStackSpec`
- 声明 `videoStackSpec` 的内边距: Inset(16, 16, 0, 16); `videoInsetSpec`

特别注意 `userStackSpec` 和 `videoStackSpec`, StackSpec 多层叠加后, 父子间是存在影响的, 我在使用中也感觉比较奇怪, 具体需要自行尝试体会..

具体实现代码: VideoCellNode.m , UserNode.m

相关链接 (不分先后)

文章

iOS性能优化探讨

AsyncDisplayKit 系列教程 —— 为什么要使用 AsyncDisplayKit

新大陆 : AsyncDisplayKit

iOS 保持界面流畅的技巧

Texture-Resources (EN 推荐)

Getting-Started 入门教程之一

Texture 布局篇

Via: Conver

使用替换UIImageView -> ASImageNode/ASNetworkImageNode
APP性能的优化，一直都是任重而道远，对于如今需要承载更多信息的APP来说更是突出，值得庆幸的苹果在这方面做得至少比安卓让开发者省心。UIKit 控件虽然在大多数情况下都能满足用户对于流畅性的需求，但有时候还是难以达到理想效果。

AsyncDisplayKit(以下简称ASDK) 的出现至少又给了开发者一个不错的选择。毕竟Paper(虽然 Facebook 已经关闭了这个应用)当年有着炫酷的效果的同时依然保持较好的流畅性也得益于 ASDK 的加入。在Paper发布的几个月后 Facebook 就干脆从中剥离出来成为一个独立的库，就在前两天 ASDK 刚好发布了 2.0 版本。

目前据我所知国内比较知名有 轻芒阅读(豌豆荚一览) 、即刻 和 Yep 在用ASDK。拿 即刻 来说包括 `消息盒子` 、`主题的详情页` 、`动态通知` 、`我的喜欢` 、`评论页` 、`最近热门` 、`即刻小报` 、`他关注的人` 、`关注他的人` 以及 `搜索页` 都用到了 ADSK。

目前 AsyncDisplayKit 已经从 facebook 迁移至 TextureGroup 新的项目地址是 Texture

控件

Texture 几乎涵盖了常用的控件，下面是 `Texture` 和 `UIKit` 的对应关系，有些封装可以说非常良心。

Nodes:

Texture	UIKit
ASDisplayNode	UIView
ASCellNode	UITableViewCell/UICollectionViewCell
ASTextNode	UILabel
ASImageNode	UIImageView
ASNetworkImageNode	UIImageView
ASVideoNode	AVPlayerLayer
ASControlNode	UIControl
ASScrollNode	UIScrollView
ASControlNode	UIControl
ASEditableTextNode	UITextView
ASMultiplexImageNode	UIImageView

Node Containers

Texture	UIKit
ASViewController	UIViewController
ASTableNode	UITableView
ASCollectionNode	UICollectionView
ASPagerNode	UICollectionView

子父类关系:

- ASDisplayNode
  - ASCellNode
    - ASTextCellNode
  - ASCollectionNode

- ASPagerNode

- ASControlNode
  - ASButtonNode
  - ASImageNode
    - ASMapNode
    - ASMultiplexImageNode
    - ASNetworkImageNode
      - ASVideoNode
  - ASTextNode
  - ASTextNode2

- ASEditableTextNode
- ASScrollNode
- ASTableNode
- ASVideoPlayerNode

ASDisplayNode:

作用同等于 `UIView`，是所有 Node 的父类，需要注意的是 `ASDisplayNode` 其实拥有一个 `view` 属性，所以 `ASDisplayNode` 及其子类都可以通过这个 `view` 来添加 `UIKit` 控件，这样一来 `Texture` 和 `UIKit` 混用是完全没问题的。

`ASDisplayNode` 中添加 `UIKit`

```
UIView *otherView = [[UIView alloc] init];
otherView.frame = ...;
[self.view addSubview:otherView];

```

或

```
ASDisplayNode *gradientNode = [[ASDisplayNode alloc] initWithViewBlock:^UIView * _Nonnull
    UIView *view = [[UIView alloc] init];
    return view;
}];

```

第二种的初始化最终生成的就是 block 返回的 `UIKit` 对象，但外部表现出来的是 `ASDisplayNode`。这样子的好处在于布局，关于布局，后面会讲到。

`UIKit` 中添加 `ASDisplayNode`

```
ASImageNode *imageNode = [[ASImageNode alloc] init];
imageNode.image = [UIImage imageNamed:@"iconShowMore"];
imageNode.frame = ...;
[self addSubnode:imageNode];
self.imageNode = imageNode;

```

ASCellNode:

作用同等于 `UITableViewCell` 或 `UICollectionViewCell`，自带 `indexPath` 属性，有些时候很有用。

ASTextNode

作用同等于 `UILabel`，和 `UILabel` 不同的是 `ASTextNode` 必须通过 `attributedText` 添加文字。

ASTextNode2

在 ASTextNode 基础修复了一些 Bug

ASImageNode

作用同等于 `UIImageView`，但是只能设置静态图片，如果需要使用网络图片，请使用 `ASNetworkImageNode`。

ASNetworkImageNode

作用同等于 `UIImageView`，如果使用网络图片请使用此类，`Texture` 用的是第三方的图片加载库 PINRemoteImage，`ASNetworkImageNode` 其实并不支持 gif，如果需要显示 gif 推荐使用FLAnimatedImage。

ASButtonNode

作用同等于 `UIButton`，需要注意的是下面这个两个属性

```
@property (nonatomic, assign) CGFloat contentSpacing;// 设置图片和文字的间距
@property (nonatomic, assign) ASButtonNodeImageAlignment imageAlignment;// 图片和文字的排列

```

简直要抱头痛哭一下😭，`imageAlignment` 可以设置两个值：

```
ASButtonNodeImageAlignmentBeginning, // 图片在前，文字在后
ASButtonNodeImageAlignmentEnd// 文字在前，图片在后

```

ASTableNode

作用同等于 `UITableView`，但是实现上并没有采用 `UITableView` 的重用机制，而是通过用户滚动对需要显示的视图进行 `add` 和 不需要的进行 `remove` 的操作（我猜的）。另外重要的一点：`ASTableNode` 并没有像 `UITableView` 一样提供一个 `-tableView:heightForRowAtIndexPath:` 协议方法来决定每个 Cell 的高度，而是由 `ASCellNode` 本身决定。这样带来的另外一个好处是，动态高度的实现可谓是易如反掌，具体可以看官方 Demo 中的 Kittens。

如何正确的使用

对于现有的项目中出现的并不严重的性能问题，我的建议是用对应的 Texture 控件代替即可。

比如把 `UIImageView -> ASImageNode/ASNetworkImageNode`，`UILabel -> ASTextNode` 之类的，而不是把原有的 `UITableView -> ASTableNode`，`UICollectionView -> ASCollectionNode`。

在 Cell 中替换 UIImageView

```
ASImageNode *imageNode = [[ASImageNode alloc] init];
imageNode.image = [UIImage imageNamed:@"iconShowMore"];
imageNode.frame = ...;
[self.contentView addSubnode:imageNode];
self.imageNode = imageNode;

```

原因有以下几点：

1. `ASCellNode` 内部的布局会用到 Texture 本身有一套布局方案，然而这套布局学习成本略高。
2. 包括 `ASTableNode` 和 `ASCollectionNode` 和原生的 `UITableView` 和 `UICollectionView` 有较大的 API 改变，侵略性较大，不太利于后期维护。
3. 第三方的支持问题，例如 DZNEmptyDataSet 对于 `ASTableNode` 和 `ASCollectionNode` 的支持还是有点问题。

所以当你还没有做好应付上面三个问题的准备，简单的 `UIKit -> Texture` 替换才是正确选择。

布局

阅读 Texture 布局篇

其他

刷新列表

无论是 `ASTableNode` 还是 `ASCollectionNode` 当列表中已经有数据显示了，调用 `reloadData` 你会发现列表会闪一下。最常见的案例是上拉加载更多获取到新数据后调用 `reloadData` 刷新列表用户体验会比较差，事实上官方文档在 [Batch Fetching API] 给出了解决办法：

```
- (void)tableNode:(ASTableNode *)tableNode willBeginBatchFetchWithContext:(ASBatchContext
{
  // Fetch data most of the time asynchronoulsy from an API or local database
  NSArray *newPhotos = [SomeSource getNewPhotos];

  // Insert data into table or collection node
  [self insertNewRowsInTableNode:newPhotos];

  // Decide if it's still necessary to trigger more batch fetches in the future
  _stillDataToFetch = ...;

  // Properly finish the batch fetch
  [context completeBatchFetching:YES];
}
```

获取新数据后直接插入到列表中，而不是刷新整个列表，比如：

```
- (void)insertSections:(NSIndexSet *)sections withRowAnimation:(UITableViewRowAnimation)a

```

和

```
- (void)insertRowsAtIndexPaths:(NSArray<NSIndexPath *> *)indexPaths withRowAnimation:(UIT

```

加载数据

细心的同学可能发现了前面提到内容中的就有相关的方法：

```
- (void)tableNode:(ASTableNode *)tableNode willBeginBatchFetchWithContext:(ASBatchContext

```

现在绝大多数APP加载更多数据的方法都是通过下拉到列表底部再去请求数据然后添加到列表中，但是 Texture 提供了另外一种更“合理”的方式，原文是这样描述的：

> By default, as a user is scrolling, when they approach the point in the table or collection where they are 2 “screens” away from the end of the current content, the table will try to fetch more data.

当列表滚到到距离底部还有两个屏幕高度请求新的数据，这个阈值是可以调整的。一旦距离底部达到两个屏幕的高度的时候，就会调用前面提到的方法。所以用起来大概是这样的：

```
- (void)tableNode:(ASTableNode *)tableNode willBeginBatchFetchWithContext:(ASBatchContext
    [context beginBatchFetching];
    [listApi startWithBlockSuccess:^(HQHomeListApi *request) {
        @strongify(self);
        NSArray *array = [request responseJSONObject];
        [self.dataSourceArray addObjectsFromArray:array];
        [self.tableNode insertSections:[NSIndexSet indexSetWithIndexesInRange:rang] withR
        [self updateHavMore:array];
        [context completeBatchFetching:YES];
    } failure:NULL];
}

- (BOOL)shouldBatchFetchForTableNode:(ASTableNode *)tableNode{
    return self.haveMore;
}
```

`shouldBatchFetchForTableNode` 用来控制是否需要获取更多数据。这种方式优点在于在网络状况好的情况下用户都不会感受到已经加载了其他数据并显示，缺点在于网络状况不好的情况下用于即使列表已经下拉到底部也没有任何提示

全部评论 2

只看作者

按时间倒序 按时间正序

d0u
3楼 2021.08.16 10:12
这个不能用masonry吗?
赞 回复

613808bc19af
2楼 2019.11.27 14:46
谢谢带入门
赞 回复

被以下专题收入，发现更多相似内容


iOS开发知识点


iOS


iOS应用点


Texture


iOS精选


iOS开发之常...


UI的实现

推荐阅读

更多精彩内容 >

iOS工具和框架1

1、通过CocoaPods安装项目名称项目信息 AFNetworking网络请求组件 FMDB本地数据库组件 SD...


爱运动爱学习 阅读 15,270 评论 3 赞 114

iOS 面试宝典 没有比这更全的了(持续更新)

1.ios高性能编程 (1).内层 最小的内层平均值和峰值(2).耗电量 高效的算法和数据结构(3).初始化时...

欧辰_OSR 阅读 25,743 评论 8 赞 249

【文】出门

走出房子，下了楼，吱呀一声，一把推开大铁门，陡然一股冷风袭来。不，准确点说，是寒气袭来。两腿感觉最是明显，丝丝寒...


加果石头 阅读 82 评论 0 赞 3

【武侠】飞鹤传情(61)

第五十九章 王爷的名声 “王爷。”刘一首不知道什么时候走了进来，“我听说有刺客，怎么样了，您没有伤到吧！”“哦...

沐弘晨

巴伐利亚的玛丽·安娜公主，萨克森王后1805–1877

玛丽·安娜公主和上文的索菲公主是双胞胎，巴登家的卡洛琳公主能一下子生两胎双胞胎真是太神奇了。与索菲顺利嫁到奥地利...

徐三余

天啦，原来福报会这么点点滴滴折掉的！

很多人都希望能修福，希望自己的生活过得好一些，过得自在一些，富足一些。因此，会去做许多的善事，诸如布施、供养、放生...

弥陀弟子耀清