

原 iOS-TextField知多少

分类: [iOS编程](#)

2012-10-28 20:44 46475人阅读 评论(21) 收藏 举报

//初始化textfield并设置位置及大小

```
UITextField *text = [[UITextField alloc] initWithFrame:CGRectMake(20, 20, 130, 30)];
```

//设置边框样式，只有设置了才会显示边框样式

```
text.borderStyle = UITextBorderStyleRoundedRect;
```

```
typedef enum {
```

```
    UITextBorderStyleNone,
```

```
    UITextBorderStyleLine,
```

```
    UITextBorderStyleBezel,
```

```
    UITextBorderStyleRoundedRect
```

```
} UITextBorderStyle;
```

//设置输入框的背景颜色，此时设置为白色 如果使用了自定义的背景图片边框会被忽略掉

```
text.backgroundColor = [UIColor whiteColor];
```

//设置背景

```
text.background = [UIImage imageNamed:@"dd.png"];
```

//设置背景

```
text.disabledBackground = [UIImage imageNamed:@"cc.png"];
```

//当输入框没有内容时，水印提示 提示内容为password

```
text.placeholder = @"password";
```

//设置输入框内容的字体样式和大小

```
text.font = [UIFont fontWithName:@"Arial" size:20.0f];
```

//设置字体颜色

```
text.textColor = [UIColor redColor];
```

```
//输入框中是否有个叉号，在什么时候显示，用于一次性删除输入框中的内容
```

```
text.clearButtonMode = UITextFieldViewModeAlways;
```

```
typedef enum {
```

```
    UITextFieldViewModeNever,    重不出现
```

```
    UITextFieldViewModeWhileEditing, 编辑时出现
```

```
    UITextFieldViewModeUnlessEditing, 除了编辑外都出现
```

```
    UITextFieldViewModeAlways    一直出现
```

```
} UITextFieldViewMode;
```

```
//输入框中一开始就有的文字
```

```
text.text = @"一开始就在输入框的文字";
```

```
//每输入一个字符就变成点 用语密码输入
```

```
text.secureTextEntry = YES;
```

```
//是否纠错
```

```
text.autocorrectionType = UITextAutocorrectionTypeNo;
```

```
typedef enum {
```

```
    UITextAutocorrectionTypeDefault, 默认
```

```
    UITextAutocorrectionTypeNo,    不自动纠错
```

```
    UITextAutocorrectionTypeYes,   自动纠错
```

```
} UITextAutocorrectionType;
```

```
//再次编辑就清空
```

```
text.clearsOnBeginEditing = YES;
```

```
//内容对齐方式
```

```
text.textAlignment = NSTextAlignmentLeft;
```

//内容的垂直对齐方式 UITextField继承自UIControl,此类中有一个属性contentVerticalAlignment

```
text.contentVerticalAlignment = UIControlContentVerticalAlignmentCenter;
```

//设置为YES时文本会自动缩小以适应文本窗口大小.默认是保持原来大小,而让长文本滚动

```
textFied.adjustsFontSizeToFitWidth = YES;
```

//设置自动缩小显示的最小字体大小

```
text.minimumFontSize = 20;
```

//设置键盘的样式

```
text.keyboardType = UIKeyboardTypeNumberPad;
```

typedef enum {

UIKeyboardTypeDefault, 默认键盘, 支持所有字符

UIKeyboardTypeASCIICapable, 支持ASCII的默认键盘

UIKeyboardTypeNumbersAndPunctuation, 标准电话键盘, 支持 + * # 字符

```
        UIKeyboardTypeURL,           URL键盘，支持.com按钮 只支持URL字符
UIKeyboardTypeNumberPad,           数字键盘
UIKeyboardTypePhonePad,           电话键盘
        UIKeyboardTypeNamePhonePad,   电话键盘，也支持输入人名
UIKeyboardTypeEmailAddress,       用于输入电子 邮件地址的键盘
UIKeyboardTypeDecimalPad,         数字键盘 有数字和小数点
        UIKeyboardTypeTwitter,        优化的键盘，方便输入@、#字符
        UIKeyboardTypeAlphabet = UIKeyboardTypeASCIICapable,
} UIKeyboardType;

//首字母是否大写
text.autocapitalizationType = UITextAutocapitalizationTypeNone;

typedef enum {
    UITextAutocapitalizationTypeNone, 不自动大写
    UITextAutocapitalizationTypeWords, 单词首字母大写
    UITextAutocapitalizationTypeSentences, 句子的首字母大写
```

```
        UITextAutocapitalizationTypeAllCharacters, 所有字母都大写  
    } UITextAutocapitalizationType;
```

```
//return键变成什么键
```

```
text.returnKeyType =UIReturnKeyDone;
```

```
typedef enum {
```

```
    UIReturnKeyDefault, 默认 灰色按钮, 标有Return
```

```
    UIReturnKeyGo,      标有Go的蓝色按钮
```

```
    UIReturnKeyGoogle, 标有Google的蓝色按钮, 用语搜索
```

```
    UIReturnKeyJoin, 标有Join的蓝色按钮
```

```
    UIReturnKeyNext, 标有Next的蓝色按钮
```

```
    UIReturnKeyRoute, 标有Route的蓝色按钮
```

```
    UIReturnKeySearch, 标有Search的蓝色按钮
```

```
    UIReturnKeySend, 标有Send的蓝色按钮
```

```
    UIReturnKeyYahoo, 标有Yahoo的蓝色按钮
```

```
        UIReturnKeyYahoo, 标有Yahoo的蓝色按钮
        UIReturnKeyEmergencyCall, 紧急呼叫按钮
    } UIReturnKeyType;

//键盘外观
textView.keyboardAppearance=UIKeyboardAppearanceDefault;
typedef enum {
    UIKeyboardAppearanceDefault, 默认外观, 浅灰色
    UIKeyboardAppearanceAlert, 深灰 石墨色
} UIReturnKeyType;

//设置代理 用于实现协议
    text.delegate = self;

//把textfield加到视图中
```



```
[self.window addSubview:text];
```

//最右侧加图片是以下代码 左侧类似

```
UIImageView *image=[[UIImageView alloc] initWithImage:[UIImage  
imageNamed:@"right.png"]];
```

```
text.rightView=image;
```

```
text.rightViewMode = UITextFieldViewModeAlways;
```

```
typedef enum {
```

```
    UITextFieldViewModeNever,
```

```
    UITextFieldViewModeWhileEditing,
```

```
    UITextFieldViewModeUnlessEditing,
```

```
    UITextFieldViewModeAlways
```

```
} UITextFieldViewMode;
```

//按return键键盘往下收 becomeFirstResponder

类要采用UITextFieldDelegate协议

`text.delegate = self;` 声明text的代理是我，我会去实现把键盘往下收的方法 这个方法在UITextFieldDelegate里所以我们要采用UITextFieldDelegate这个协议

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField
{
    [text resignFirstResponder];    //主要是[receiver resignFirstResponder]在哪调用就能把
receiver对应的键盘往下收
    return YES;
}
```

重写绘制行为

除了UITextField对象的风格选项，你还可以定制化UITextField对象，为他添加许多不同的重写方法，来改变文本字段的显示行为。这些方法都会返回一个CGRect结构，制定了文本字段每个部件的边界范围。以下方法都可以重写。

```
- textRectForBounds:      //重写来重置文字区域
- drawTextInRect:         //改变绘文字属性.重写时调用super可以按默认图形属性绘制,若自己完全重写绘制函数,就不用调用super了.
- placeholderRectForBounds:  //重写来重置占位符区域
- drawPlaceholderInRect:    //重写改变绘制占位符属性.重写时调用super可以按默认图形属性绘制,若自己完全重写绘制函数,就不用调用super了.
- borderRectForBounds:     //重写来重置边缘区域
- editingRectForBounds:    //重写来重置编辑区域
- clearButtonRectForBounds: //重写来重置clearButton位置,改变size可能导致button的图片失真
- leftViewRectForBounds:
- rightViewRectForBounds:
```

委托方法

```
- (BOOL)textFieldShouldBeginEditing:(UITextField *)textField{
```

```
//返回一个BOOL值, 指定是否循序文本字段开始编辑
```

```
        return YES;
    }

- (void)textFieldDidBeginEditing:(UITextField *)textField{

    //开始编辑时触发，文本字段将成为first responder
}

- (BOOL)textFieldShouldEndEditing:(UITextField *)textField{

    //返回BOOL值，指定是否允许文本字段结束编辑，当编辑结束，文本字段会让出first responder

    //要想在用户结束编辑时阻止文本字段消失，可以返回NO

    //这对一些文本字段必须始终保持活跃状态的程序很有用，比如即时消息

    return NO;
}
```

```
- (BOOL)textField:(UITextField*)textField shouldChangeCharactersInRange:
(NSRange)range replacementString:(NSString *)string{
```

//当用户使用自动更正功能，把输入的文字修改为推荐的文字时，就会调用这个方法。

//这对于想要加入撤销选项的应用程序特别有用

//可以跟踪字段内所做的最后一次修改，也可以对所有编辑做日志记录,用作审计用途。

//要防止文字被改变可以返回NO

//这个方法的参数中有一个NSRange对象，指明了被改变文字的位置，建议修改的文本也在其中

```
    return YES;
```

```
}
```

```
- (BOOL)textFieldShouldClear:(UITextField *)textField{
```

//返回一个BOOL值指明是否允许根据用户请求清除内容

//可以设置在特定条件下才允许清除内容

```
        return YES;
    }

    -(BOOL)textFieldShouldReturn:(UITextField *)textField{

        //返回一个BOOL值，指明是否允许在按下回车键时结束编辑

        //如果允许要调用resignFirstResponder 方法，这回导致结束编辑，而键盘会被收起
        [textField resignFirstResponder];

        //查一下resign这个单词的意思就明白这个方法了

        return YES;
    }
```

通知

UITextField派生自UIControl，所以UIControl类中的通知系统在文本字段中也可以使用。除了UIControl类的标准事件，你还可以使用下列UITextField类特有的事件

UITextFieldTextDidBeginEditingNotification

UITextFieldTextDidChangeNotification

UITextFieldTextDidEndEditingNotification

当文本字段退出编辑模式时触发。通知的object属性存储了最终文本。

因为文本字段要使用键盘输入文字，所以下面这些事件发生时，也会发送动作通知

UIKeyboardWillShowNotification //键盘显示之前发送

UIKeyboardDidShowNotification //键盘显示之后发送

UIKeyboardWillHideNotification //键盘隐藏之前发送

UIKeyboardDidHideNotification //键盘隐藏之后发送

1、Text ： 设置文本框的默认文本。

2、Placeholder ： 可以在文本框中显示灰色的字，用于提示用户应该在这个文本框输入什么内容。当这个文本框中输入

了数据时，用于提示的灰色的字将会自动消失。

3、Background：

4、Disabled：若选中此项，用户将不能更改文本框内容。

5、接下来是三个按钮，用来设置对齐方式。

6、Border Style：选择边界风格。

7、Clear Button：这是一个下拉菜单，你可以选择清除按钮什么时候出现，所谓清除按钮就是出一个现在文本框右边的小 X，你可以有以下选择：

7.1 Never appears：从不出现

7.2 Appears while editing：编辑时出现

7.3 Appears unless editing：

7.4 Is always visible：总是可见

8、Clear when editing begins：若选中此项，则当开始编辑这个文本框时，文本框中之前的内容会被清除掉。比如，你在这个文本框 A 中输入了 "What"，之后去编辑文本框 B，若再回来编辑文本框 A，则其中的 "What" 会被立即清除。

9、Text Color：设置文本框中文本的颜色。

10、Font：设置文本的字体与字号。

11、Min Font Size：设置文本框可以显示的最小字体（不过我感觉没什么用）

12、Adjust To Fit：指定当文本框尺寸减小时，文本框中的文本是否也要缩小。选择它，可以使得全部文本都可见，即使文本很长。但是这个选项要跟 Min Font Size 配合使用，文本再缩小，也不会小于设定的 Min Font Size。

接下来的部分用于设置键盘如何显示。

13、Captitalization : 设置大写。下拉菜单中有四个选项:

13.1 None : 不设置大写

13.2 Words : 每个单词首字母大写, 这里的单词指的是以空格分开的字符串

13.3 Sentances : 每个句子的第一个字母大写, 这里的句子是以句号加空格分开的字符串

13.4 All Characters : 所有字母大写

14、Correction : 检查拼写, 默认是 YES 。

15、Keyboard : 选择键盘类型, 比如全数字、字母和数字等。

16、Appearance:

17、Return Key : 选择返回键, 可以选择 Search 、 Return 、 Done 等。

18、Auto-enable Return Key : 如选择此项, 则只有至少在文本框输入一个字符后键盘的返回键才有效。

19、Secure : 当你的文本框用作密码输入框时, 可以选择这个选项, 此时, 字符显示为星号。

1.Alignment Horizontal 水平对齐方式

2.Alignment Vertical 垂直对齐方式

3.用于返回一个BOOL值 输入框是否 Selected(选中) Enabled(可用) Highlighted(高亮)

限制只能输入特定的字符

```
(BOOL)textField:(UITextField *)textField shouldChangeCharactersInRange:(NSRange)range
replacementString:(NSString *)string{

    NSMutableCharacterSet *cs;

    cs = [[NSMutableCharacterSet characterSetWithCharactersInString:NUMBERS]invertedSet];

    NSString *filtered = [[string
componentsSeparatedByCharactersInSet:cs]componentsJoinedByString:@""]; //按cs分离出数组,数组
按@" "分离出字符串

    BOOL canChange = [string isEqualToString:filtered];

    return canChange;
}
```

上面那个NUMBERS是一个宏，可以在文件顶部定义：

`#define NUMBERS @"0123456789\n"`（这个代表可以输入数字和换行，请注意这个\n，如果不写这个，Done按键将不会触发，如果用在SearchBar中，将会不触发Search事件，因为你自己限制不让输入\n，好惨，我在项目中才发现的。）

所以，如果你要限制输入英文和数字的话，就可以把这个定义为：

`#define kAlphaNum @"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"`。

当然，你还可以在以上方法return之前，做一提示的，比如提示用户只能输入数字之类的。如果你觉得有需要的话。

限制只能输入一定长度的字符

```
– (BOOL)textField:(UITextField *)textField shouldChangeCharactersInRange:(NSRange)range  
replacementString:(NSString *)string;
```

{ //string就是此时输入的那个字符 textField就是此时正在输入的那个输入框 返回YES就是可以改变输入框的值 NO相反

```
if ([string isEqualToString:@"\n"]) //按回车可以改变  
{  
    return YES;  
}
```

```
NSString * toBeString = [textField.text stringByReplacingCharactersInRange:range
withString:string]; //得到输入框的内容

if (self.myTextField == textField) //判断是否是我们想要限定的那个输入框
{
    if ([toBeString length] > 20) { //如果输入框内容大于20则弹出警告
        textField.text = [toBeString substringToIndex:20];
        UIAlertView *alert = [[[UIAlertView alloc] initWithTitle:nil message:@"超过最大字
数不能输入了" delegate:nil cancelButtonTitle:@"Ok" otherButtonTitles:nil, nil] autorelease];
        [alert show];
        return NO;
    }
}

return YES;
}
```

▼ 下一篇

Windows、Unix、Mac不同操作系统的数据问题-讨论回车符\r和换行符\n