How do you make dynamic WeChat pages?

Architecture

```
/
├── app.js
├── app.json
├── app.wxss
├── project.config.json
├── pages
│   ├── index
│   │   ├── index.js
│   │   ├── index.json
│   │   ├── index.wxml
│   │   └── index.wxss
│   └── logs
│       ├── logs.js
│       ├── logs.json
│       ├── logs.wxml
│       └── logs.wxss
└── utils
    └── util.js
```

Data Binding

Data binding uses mustache syntax to wrap variables.

```
<!-- .wxml -->
<view>{{text}}</view>
<view>{{array[0].msg}}</view>
```

```
// .js
Page({
  data: {
    text: 'init data',
    array: [{msg: 'I am message 1'}, {msg: 'I am message 2'}]
  }
}
```

Conditional Rendering

```
<!-- .wxml -->
<view wx:if="{{length > 5}}"> 1 </view>
<view wx:elif="{{length > 2}}"> 2 </view>
<view wx:else> 3 </view>
```

```
// .js
Page({
  data: {
    length: 10
  }
})
```

Event handlers

```
<!-- .wxml -->
<button type="primary" bindtap="add">
Incrementation: {{count}}
</button>

// .js
Page({
  data: {
    count: 1
  },
  add: function(e) {
    this.setData({
      count: this.data.count + 1
    })
  }
})
```

WXML

<button>

Create an animated button w/wo spinner

```
<!-- .wxml -->
<form bindsubmit="bindFormSubmit">
  <button type="primary" form-type="submit"
loading="{{loading}}">Send</button>
</form>
```

```
// .js
Page({
  data:{
    loading: false
  },
  bindFormSubmit: function (e) {
    this.setData({
      loading: !this.data.loading
  })
})
```

👉 List of attributes

Action 1.

Let’s make dynamic pages

小程序

How do you use all WeChat built-in features?

APIs

👉 open.wechat.com

APIs

The framework provides plenty of native WeChat APIs that can make it easy to call the capabilities provided by WeChat, such as getting user information, local storage and payment functions.

Description:

- The wx.on start API is an API interface that monitors the occurence of a certain event. It receives a CALLBACK function as a parameter. When this event is triggered, the CALLBACK function will be called.
- if there is no specific convention, other API interfaces all receive an OBJECT as a parameter.
- The OBJECT can specify `success`, `fail`, or `complete` when receiving interface call results.

Parameter name	Type	Required	Description
success	Function	No	Callback function for successful interface call
fail	Function	No	Callback function for failed interface call
complete	Function	No	Callback function for interface call results (will be executed if call succeeds or fails)

Get the current location

Use wx.getLocation

```
<!-- .wxml -->
<button type="primary" bindtap=“listenerBtnGetLocation">
Get location</button>
```

```
// .js
listenerBtnGetLocation: function () {
  wx.getLocation({
  type: 'wgs84',
  success: function(res) {
     console.log(res)
   }
  })
 }
```

Make HTTPs requests

Use wx.request

```
// .js
giveMeCats: function () {
    var that = this
    var endpoint = "https://api.giphy.com/v1/gifs/search?
q=funny+cat&api_key=dc6zaTOxFJmzC"
    wx.request({
      url: endpoint + key,
      header: {'content-type': 'application/json'},
      success: function (res) {
       console.log('success!' + res.statusCode);
       that.setData({catData: res.data.data})
      },
      fail: function (res) {
     },
      complete: function (res) {
     }
    })
  }
```

Action 2.

Let’s make a map

Your turn