

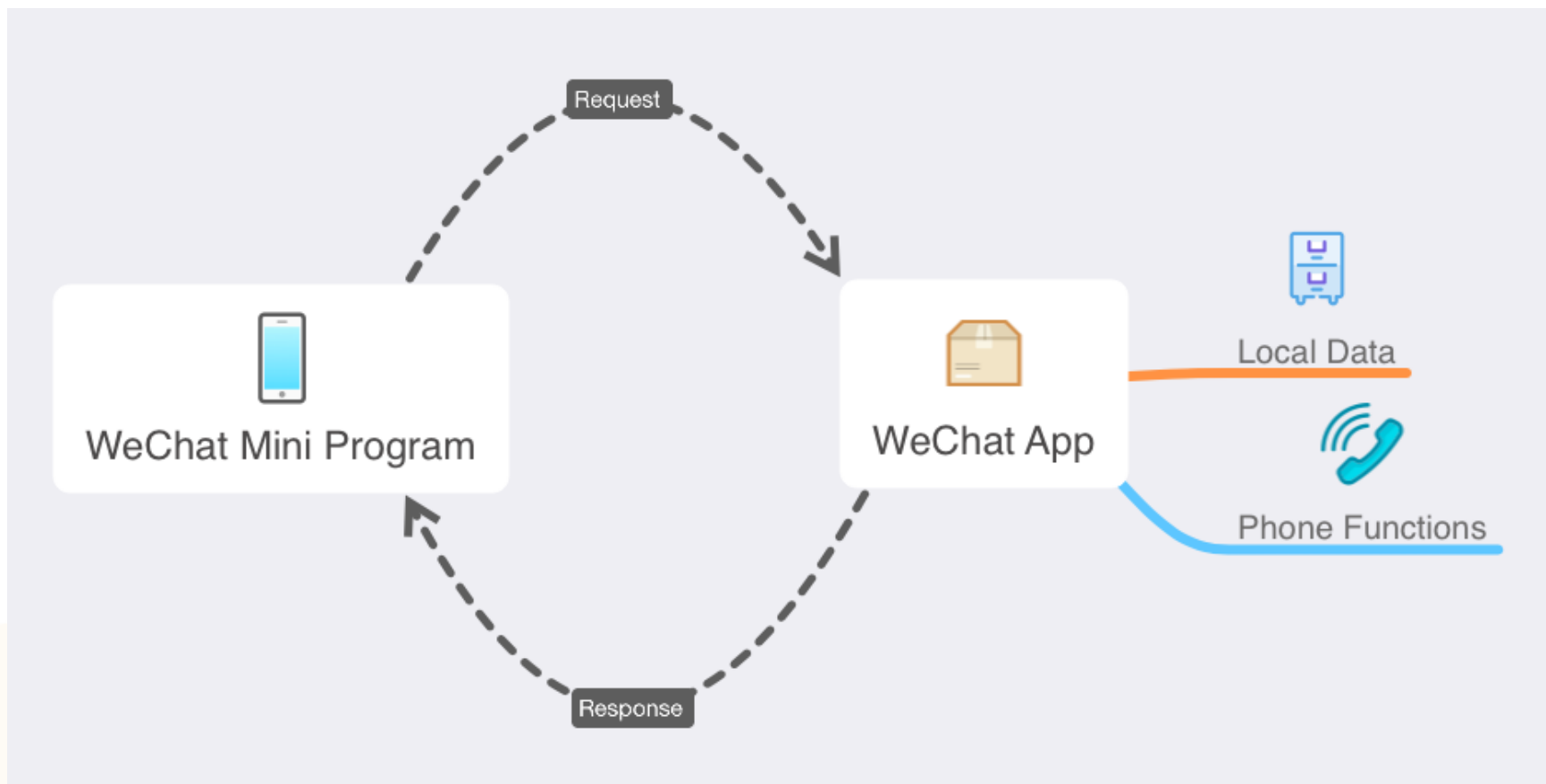
帶有API的小程序（Mini Programs）

帶有数据的微信小程序的API

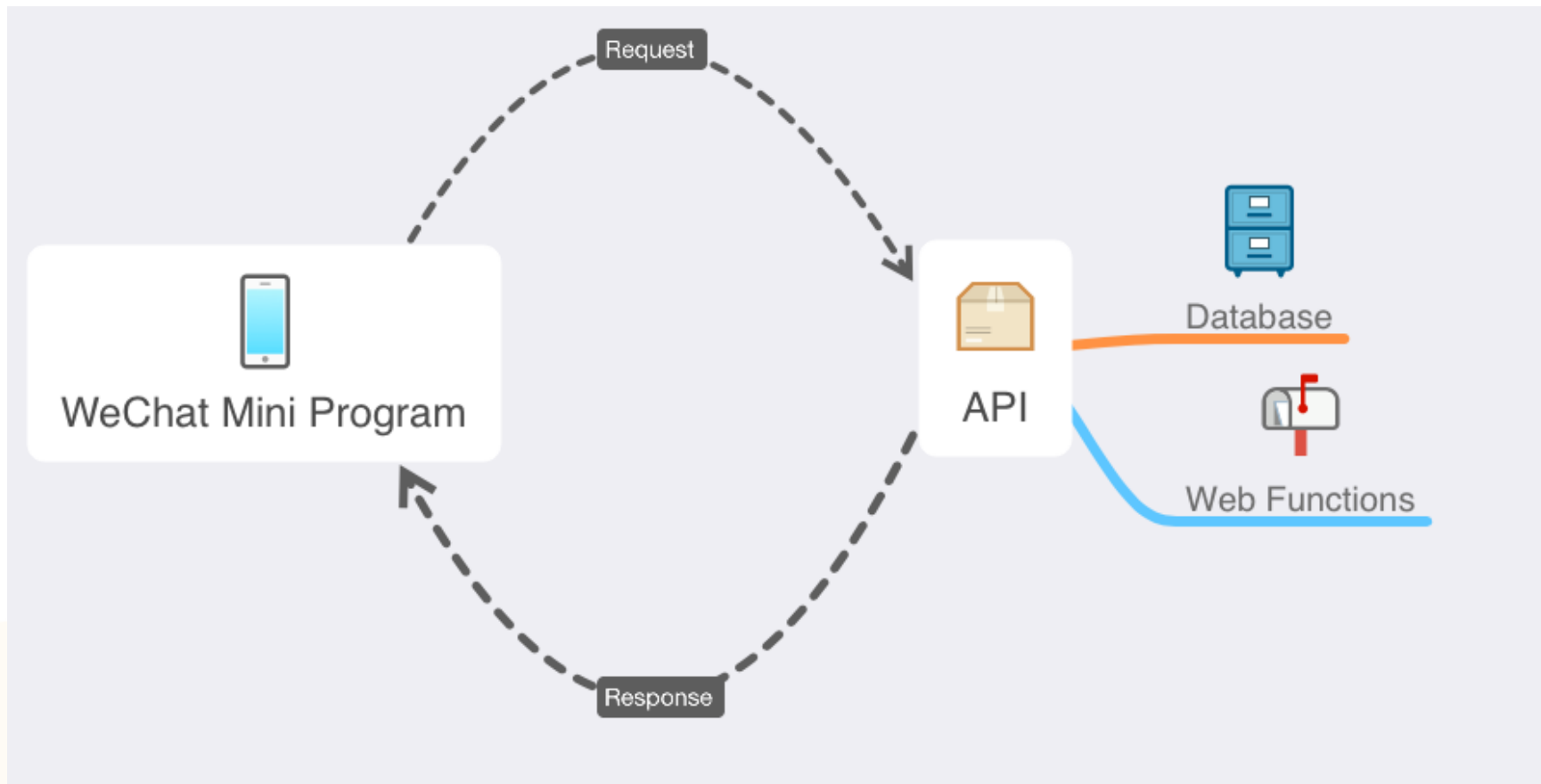


>_

之前的前端



今天，前端与API



API功能

- 存储数据在客户端本地 (e.g. app: web, 本机应用程序native, 微信小程序)
- 为客户提供服务的方法 (e.g. 短信, 付款)



NO BIG DEAL

只不过是一组为单元格内容返回**JSON**数据格式形式的新端点
(endpoints) ...



然后，JSON => 小程序

小程序发起**API请求（requests）**本地的**JSON数据**而不是静态数据
(e.g app.js文件中的全球对象globalData)



JSON复习

```
{  
  "header": {  
    "title": "The JSON example",  
    "descriptionText": "This is some title text."  
  },  
  "content": {  
    "title": "The content example text",  
    "elements": [  
      {  
        "title": "The first element",  
        "mainText": "First element main text",  
        "additionalText": "First element additional text"  
      },  
      {  
        "title": "The second element",  
        "mainText": "Second element main text",  
        "additionalText": "Second element additional text"  
      }  
    ]  
  }  
}
```



REST-FUL API

Purpose	Verb	URI Pattern	Table
all stories	GET	/stories	stories
create story	POST	/stories	stories
one story	GET	/stories/:id	stories
edit story	PUT	/stories/:id	stories
delete story	DELETE	/stories/:id	stories

请求方式: GET, POST, PUT, DELETE



使用API的6个步骤

1. 使用API key
2. 指定端点（endpoint）
3. 连接请求数据
4. 发送请求并等待响应
5. 从响应获取数据
6. 处理数据



INDEX：第一个端点

显示所有stories



1. 使用API provider提供的API token（或者key）

例如：

```
API_KEY: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
TOKEN: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```



像这样使用:

```
header: {'Authorization': 'Bearer xxxxxxxxxxxxxxxxxxxxxxxxxxxx'}
```

为参数 (params) 或表单数据

```
data: {'api_key': 'xxxxxxxxxxxxxxxxxxxxxxxxxxxx'}
```

今天的challenges不需要API token



2. 指定端点

Restful：请求方式（request）及路径（path）

```
GET /api/v1/stories
```



与主机 (host) 结合: `https://fml.shanghaiwogeng.com`

获取端点: `https://fml.shanghaiwogeng.com/api/v1/stories`



>_

```
// /pages/index/index.js
```

```
Page({
```

```
  //...
```

```
  onLoad: function (options) {
```

```
    // this代表的是当前page对象，只能在函数内部使用
```

```
    let page = this;
```

```
    ...
```

```
    const request = {
```

```
      url: `https://fml.shanghaiwogeng.com/api/v1/stories
```

```
      method: 'GET' // 如果无方法，默认GET请求方式
```

```
    }
```

```
  }
```

```
  //...
```



3. 连接请求数据

通过请求（作为json对象）获取数据

不需要index页面（我们将获取所有数据）



能够进行筛选:

```
// /pages/index/index.js
// in onLoad

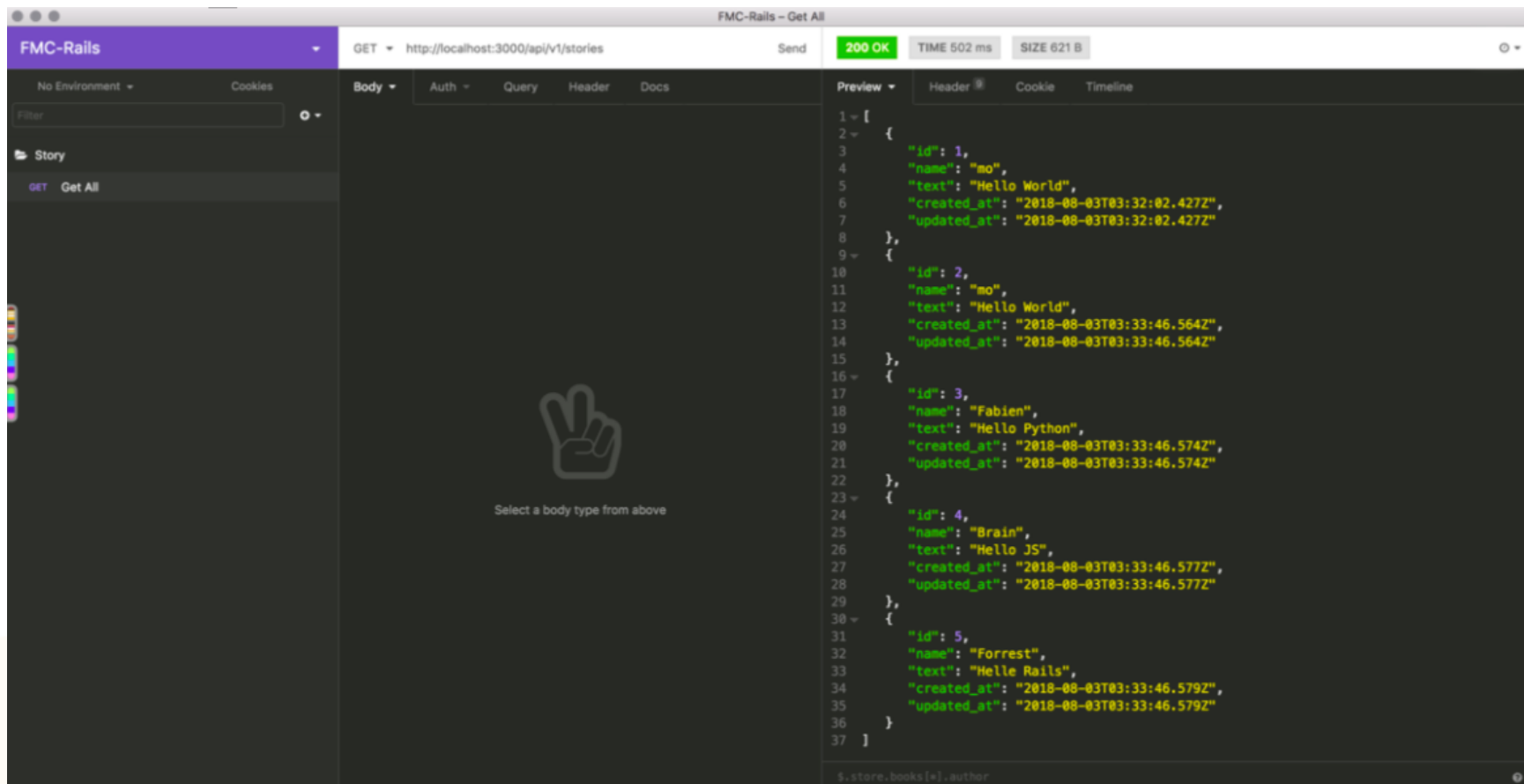
...
let filter = {
  include: 'My name',
}

const request = {
  url: `https://fml.shanghaiwogeng.com/api/v1/stories
  method: 'GET',
  data: filter // Not today, but later in course
}
```

index中没有任何数据



4. 发送请求并等待响应



The screenshot shows the FMC-Rails web interface. The top bar indicates the request method is GET and the URL is http://localhost:3000/api/v1/stories. The status is 200 OK, the response time is 502 ms, and the size is 621 B. The response body is displayed in the Preview tab, showing a JSON array of 5 story objects. The left sidebar shows the 'Story' resource and the 'GET' method selected. The main area displays a large hand icon with the text 'Select a body type from above'.

```
1 - [  
2 - {  
3   "id": 1,  
4   "name": "mo",  
5   "text": "Hello World",  
6   "created_at": "2018-08-03T03:32:02.427Z",  
7   "updated_at": "2018-08-03T03:32:02.427Z"  
8 },  
9 - {  
10  "id": 2,  
11  "name": "mo",  
12  "text": "Hello World",  
13  "created_at": "2018-08-03T03:33:46.564Z",  
14  "updated_at": "2018-08-03T03:33:46.564Z"  
15 },  
16 - {  
17  "id": 3,  
18  "name": "Fabien",  
19  "text": "Hello Python",  
20  "created_at": "2018-08-03T03:33:46.574Z",  
21  "updated_at": "2018-08-03T03:33:46.574Z"  
22 },  
23 - {  
24  "id": 4,  
25  "name": "Brain",  
26  "text": "Hello JS",  
27  "created_at": "2018-08-03T03:33:46.577Z",  
28  "updated_at": "2018-08-03T03:33:46.577Z"  
29 },  
30 - {  
31  "id": 5,  
32  "name": "Forrest",  
33  "text": "Hello Rails",  
34  "created_at": "2018-08-03T03:33:46.579Z",  
35  "updated_at": "2018-08-03T03:33:46.579Z"  
36 }  
37 ]
```



工具：使用Postman 或者 Insomnia

浏览器：`https://fml.shanghaiwogeng.com/api/v1/stories`



微信小程序：

```
// /pages/index/index.js

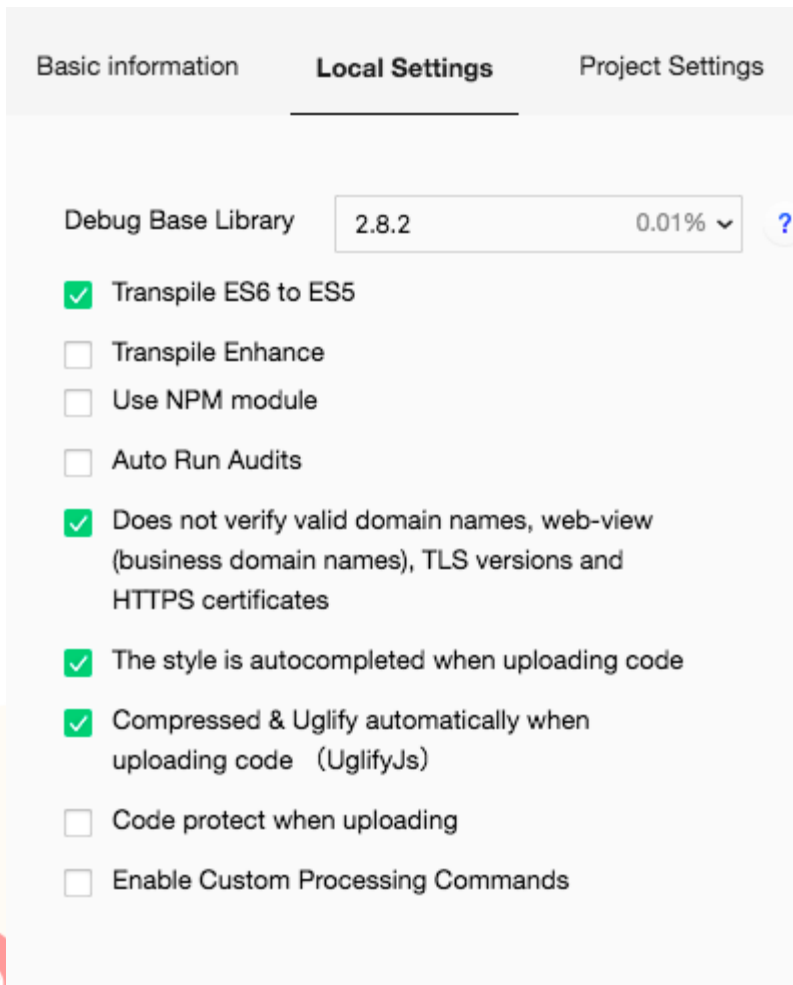
Page({
  //...
  onLoad: function (options) {
    // this代表的是当前page对象，只能在函数内部使用
    let page = this;
    ...

    const request = {
      url: `https://fml.shanghaiwogeng.com/api/v1/stories
      method: 'GET', // 如果无方法，默认GET请求方式
    }
    // 获取api数据
    wx.request(request); // 等待响应！
    // 当请求等待时，其余的代码将继续运行！
  }
  //...
});
```



Not Working? -> 需要获微信许可

Wechat IDE Menu: Settings -> Project Settings:



The screenshot shows the 'Local Settings' tab of the Wechat IDE settings. At the top, there are three tabs: 'Basic information', 'Local Settings' (which is selected and underlined), and 'Project Settings'. Below the tabs, the 'Debug Base Library' is set to '2.8.2' with a '0.01%' dropdown and a help icon. A list of settings follows, each with a checkbox and a description. The first four settings are checked: 'Transpile ES6 to ES5', 'Transpile Enhance', 'Use NPM module', and 'Auto Run Audits'. The next three are unchecked: 'Does not verify valid domain names, web-view (business domain names), TLS versions and HTTPS certificates', 'The style is autocompleted when uploading code', and 'Compressed & Uglify automatically when uploading code (UglifyJs)'. The last two are also unchecked: 'Code protect when uploading' and 'Enable Custom Processing Commands'.

Setting	Status
Debug Base Library	2.8.2 0.01% ?
<input checked="" type="checkbox"/> Transpile ES6 to ES5	Checked
<input type="checkbox"/> Transpile Enhance	Unchecked
<input type="checkbox"/> Use NPM module	Unchecked
<input type="checkbox"/> Auto Run Audits	Unchecked
<input checked="" type="checkbox"/> Does not verify valid domain names, web-view (business domain names), TLS versions and HTTPS certificates	Checked
<input checked="" type="checkbox"/> The style is autocompleted when uploading code	Checked
<input checked="" type="checkbox"/> Compressed & Uglify automatically when uploading code (UglifyJs)	Checked
<input type="checkbox"/> Code protect when uploading	Unchecked
<input type="checkbox"/> Enable Custom Processing Commands	Unchecked



5. 从响应获取数据

在index页面中添加一个新函数：`getRequestData`



```
// /pages/index/index.js
```

```
Page({  
  //...  
  getRequestData: function (res) {  
    console.log(res)  
  },  
  onLoad: function (options) {  
    //...  
  }  
})
```



>_

当请求响应 `success` 时调用该函数

```
// /pages/index/index.js

Page({
  //...
  onLoad: function (options) {
    // this代表的是当前page对象, 只能在函数内部使用
    let page = this;
    ...

    const request = {
      url: `https://fml.shanghaiwogeng.com/api/v1/stories`
      method: 'GET', // 如果无方法, 默认GET请求方式
      success: page.getRequestData
    }
    // Get api data
    wx.request(request);
  }
  //...
})
```



6. 处理数据

将响应中的数据传递给处理程序



```
// /pages/index/index.js
```

```
Page({  
  //...  
  getRequestData: function (res) {  
    console.log(res)  
  
    const data = res.data;  
    page.setStories(data);  
  },  
  onLoad: function (options) {  
    //...  
  }  
})
```



在index页中添加 `setStories` 的函数将处理数据

```
Page({
  //...
  setStories: function (data) {
    // this代表的是当前page对象, 只能在函数内部使用
    let page = this;

    // 获取stories的数据从
    const stories = data.stories;

    // 更新本地stories数据
    page.setData({
      stories: stories
    });
  }
})
```



第二个端点: **CREATE**



>_

1. 使用API token（或者key）

不需要开放API（e.g. anyone can create, 无须login）



2. 指定端点

Restful：请求方式（request）及路径（path）

```
POST /api/v1/stories
```



结合主机 (host) => 同样端点, 但POST 方法:

```
https://fml.shanghaiwogeng.com/api/v1/stories
```



```
// /pages/index/index.js
```

```
Page({
```

```
  //...
```

```
  onLoad: function (options) {
```

```
    // this代表的是当前page对象，只能在函数内部使用
```

```
    let page = this;
```

```
    ...
```

```
    const request = {
```

```
      url: `https://fml.shanghaiwogeng.com/api/v1/stories
```

```
      method: 'POST'
```

```
    }
```

```
  }
```

```
  //...
```



3. 连接请求数据

如以上的GET所示，通过请求（作为json对象）获取数据
For create，数据来自 `post` 页面的表单提交：

```
// pages/post/post.js

Page({
  // ...

  // 新的story提交
  bindSubmit: function (event) {
    console.log(event.detail.value.name)
    console.log(event.detail.value.content)

    let name = event.detail.value.name
    let text = event.detail.value.text
  }
  // ...
})
```



然后将story的表单数据转换为请求数据

```
// /pages/post/post.js
// in bindSubmit

...
let story = {
  name: name,
  text: text
}

const request = {
  url: `https://fml.shanghaiwogeng.com/api/v1/stories`,
  method: 'POST',
  data: story
}
```



4. 发送请求并等待响应



微信小程序：新的Story

如上，为api允许微信权限或跳过权限检查



>_

```
// in Page() pages/post/post.js 提交新的Story
bindSubmit: function (event) {
  console.log(event.detail.value.name)
  console.log(event.detail.value.content)

  let name = event.detail.value.name
  let text = event.detail.value.text

  let story = {
    name: name,
    text: text
  }

  const request = {
    url: `https://fml.shanghaiwogeng.com/api/v1/stories`,
    method: 'POST',
    data: story
  }

  // 发射数据
  wx.request(request); // Then wait for response!
}
```



5. 从响应获取数据

不需要响应数据，而是将重定向回index



6. 处理数据

重定向在 `success` 的函数中调用，不需要像 `index.js` 那样单独的页面函数

提示：JSON allows you to define functions inside to save you time



```
// /pages/post/post.js
// in bindSubmit

//...

const request = {
  url: `https://fml.shanghaiwogeng.com/api/v1/stories`,
  method: 'POST',
  data: story,
  success() {
    // redirect to index page when done
    wx.redirectTo({
      url: '/pages/index/index'
    });
  }
}

//...
```



小程序函数与API

深度阅读小程序api是如何构成的

- [Wechat Doc on Network Requests \(English\)](#)



API资源

我可以使用哪些api?

全球

rapidapi.com

programmableweb.com

apihound.com

apiforthat.com



国内

apistore.baidu.com

shenjian.io

juhe.cn



B2D - API经济

Revenue through api's

50% => [Salesforce.com](https://www.salesforce.com)

60% => [eBay.com](https://www.ebay.com)

90% => [Expedia.com](https://www.expedia.com)

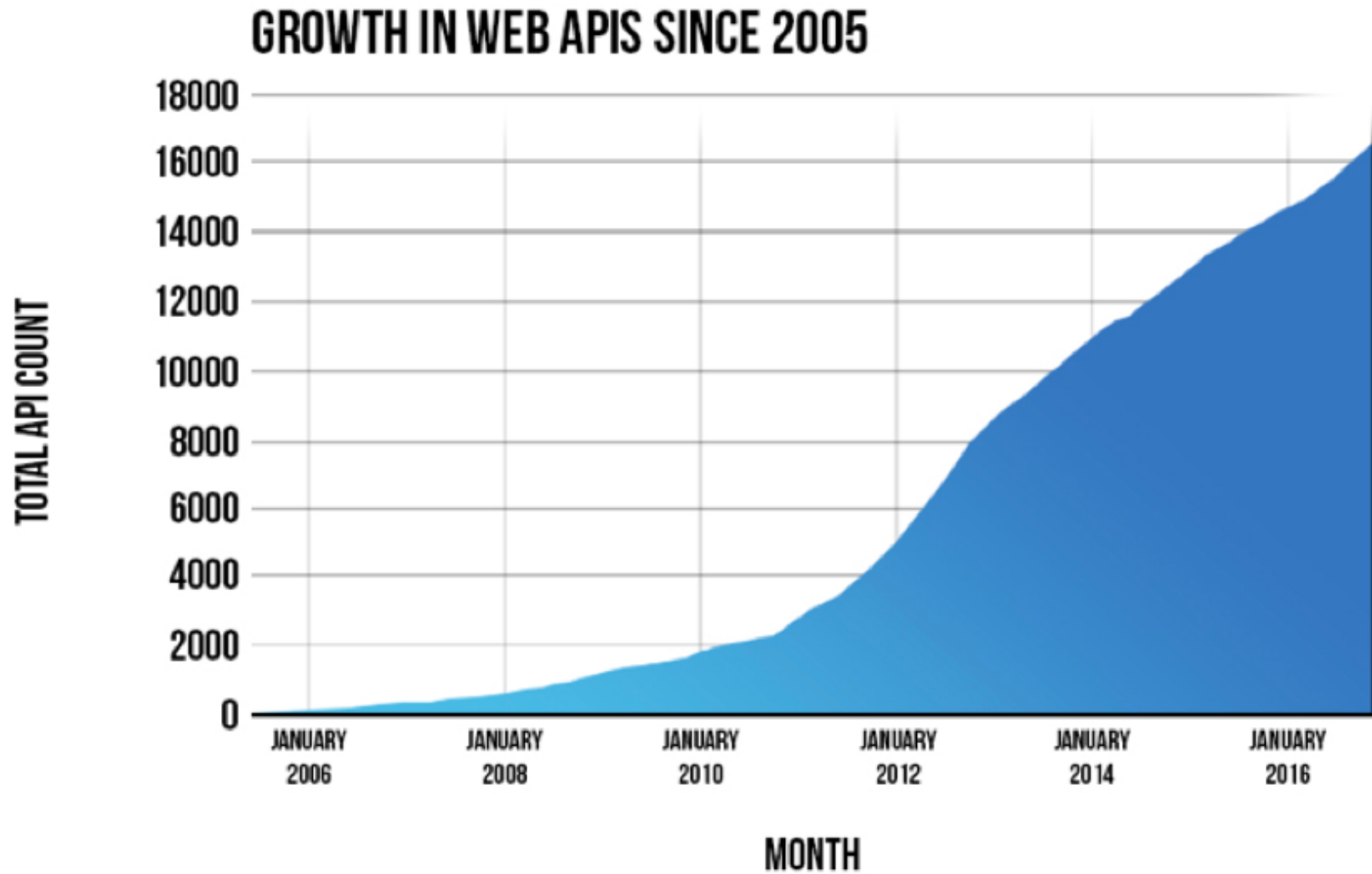


17,000个API

900万私有api开发者



Explosion of APIs growth



API策略

api对于原型设计是必不可少的——对创业者而言非常重要

关注业务的独特功能

快速且低成本地开发初始产品

api对数字转型至关重要

Use infrastructure so you don't build from scratch or reinvent the wheel



HAPPY API-ING!