# Development Tool Installation

1. Install Visual Code.

2. Install the Macle development plug-in.

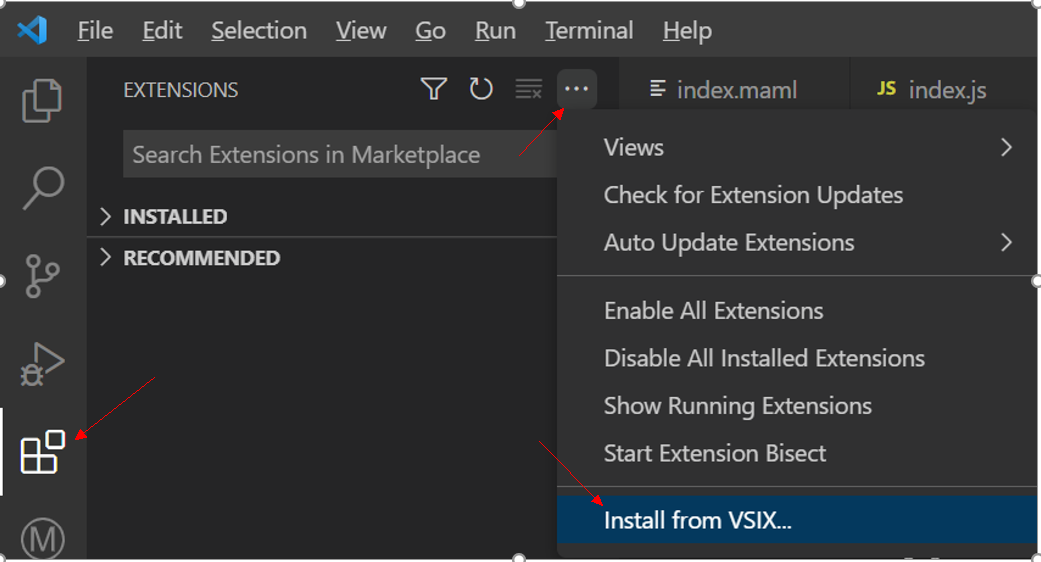
3. Install the Macle simulator.

## Installing the Visual Code

Download the Visual Code from the official website and install it, or decompress the VSCodeUserSetup-x64-1.71.2.zip file and install it.

## Installing the Macle Plug-in

After decompressing the Macle\_22.9.0\_DevSuite\_IDE.zip file, install the decompressed vsix file in Visual Code.



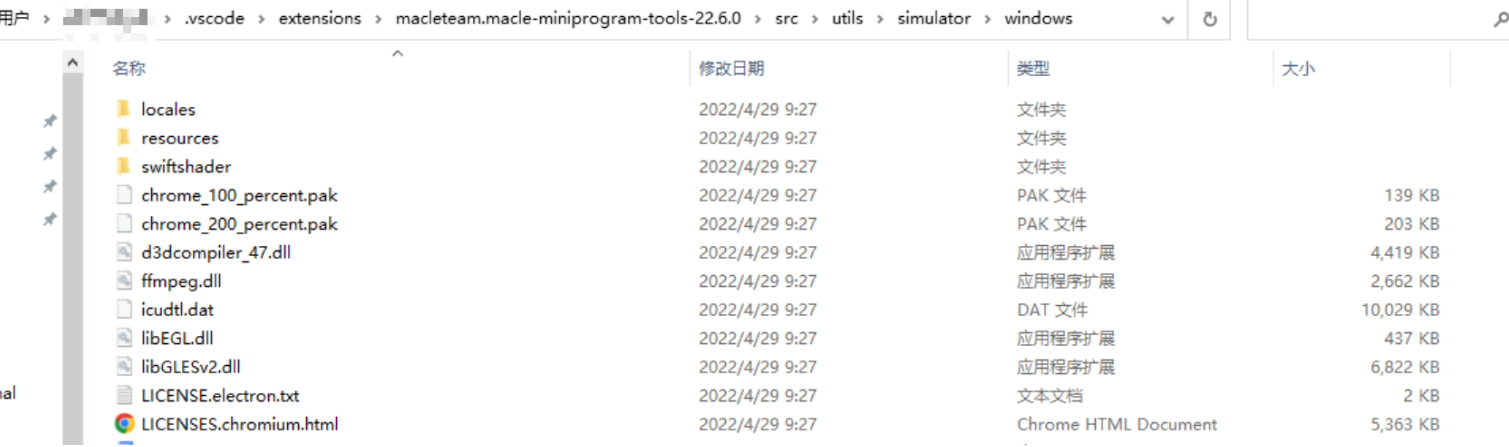
## Installing the Simulator

Window：

Extract the files from macle-simulator-win.zip to

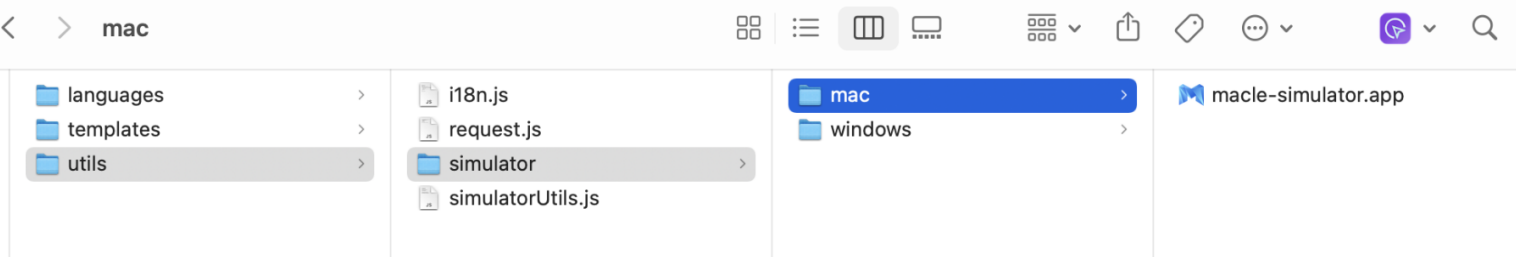
C:\Users\yourname \.vscode\extensions\ macleteam.macle-miniprogram-tools-22.9.0 \src\utils\simulator\windows Directory

Note: 1）22.9. 0 indicates the plug-in version. Replace it with the actual version.

2) If the corresponding directory cannot be found, create a simulator\windows directory.

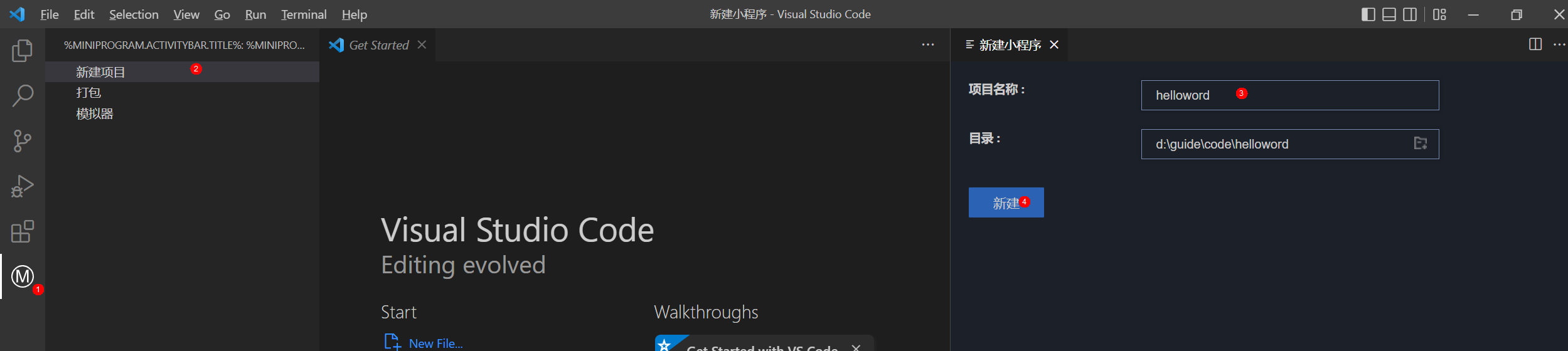
Mac：

Extract the files from macle-simulator-win.zip to ~/.vscode/extensions/macleteam.macle-miniprogram-tools-22.9.0/src/utils/simulator/mac Directory

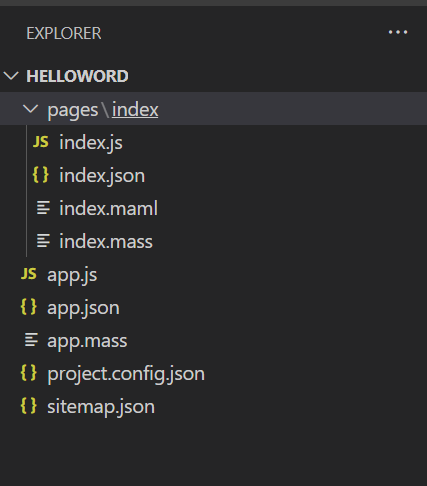


# Getting Started with Development

## Creating a Project

Follow these steps to create a helloword project

After the project is created, the project structure is as follows:



The entire project contains an index page, which contains the following files:

Index.json page configuration file (Refer to the help file, Applet -> Page Configuration)

index.maml home page label file

Index.js page script file

index.mass page style file

Open the index.maml file and use {{motto}} to display the contents of the motto variable

<!--index.maml-->

<view class="container">

    <text>{{motto}}</text>

</view>

Open the index.js file. The value of the motto variable in data on the page is hello world.

// index.js

Page({

  data: {

    motto: 'Hello World'

  },

  onLoad(options) {},

  onReady() {},

  onShow() {},

  onHide() {},

  onUnload() {},

  onShareAppMessage() {

    return {

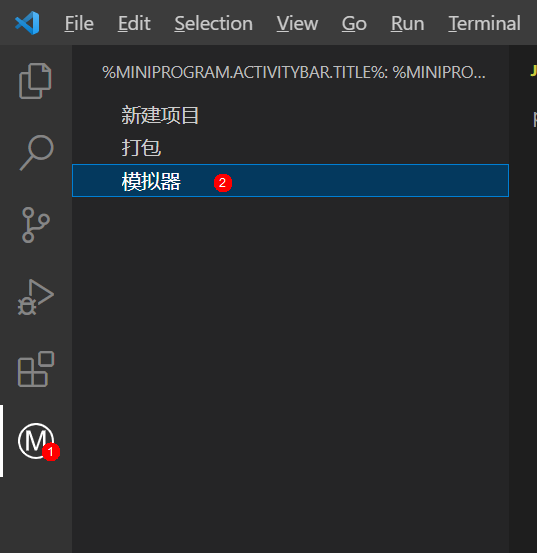
      title: ''

    };

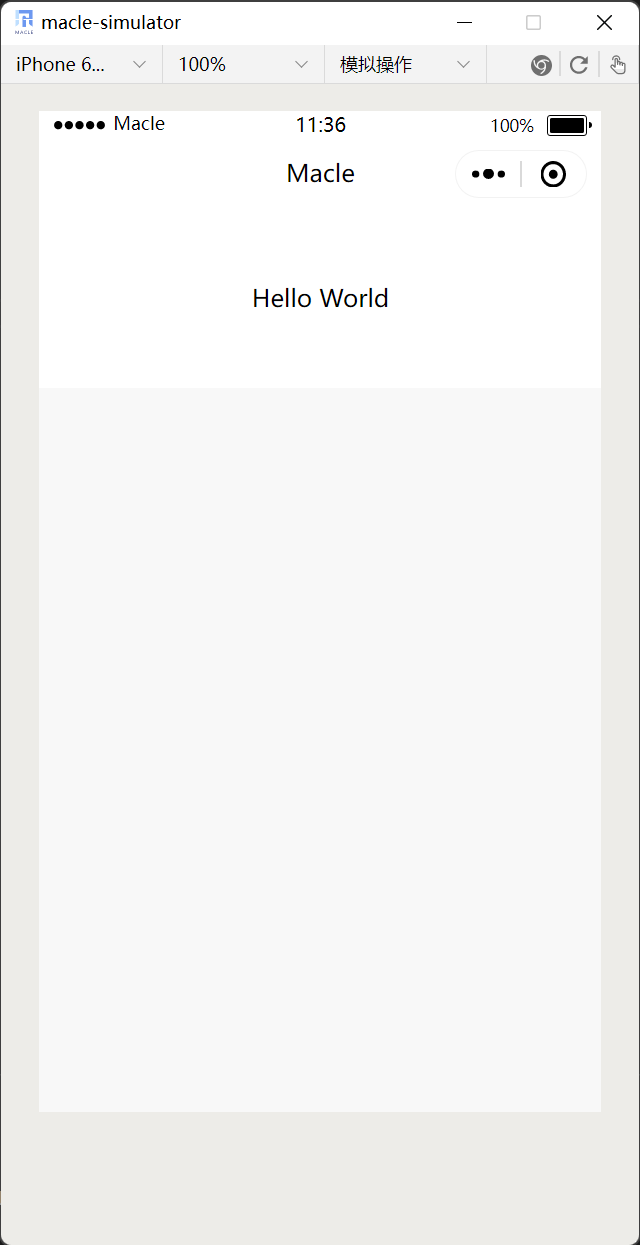
  }

});

To run the simulator to view the page effect, perform the following steps:

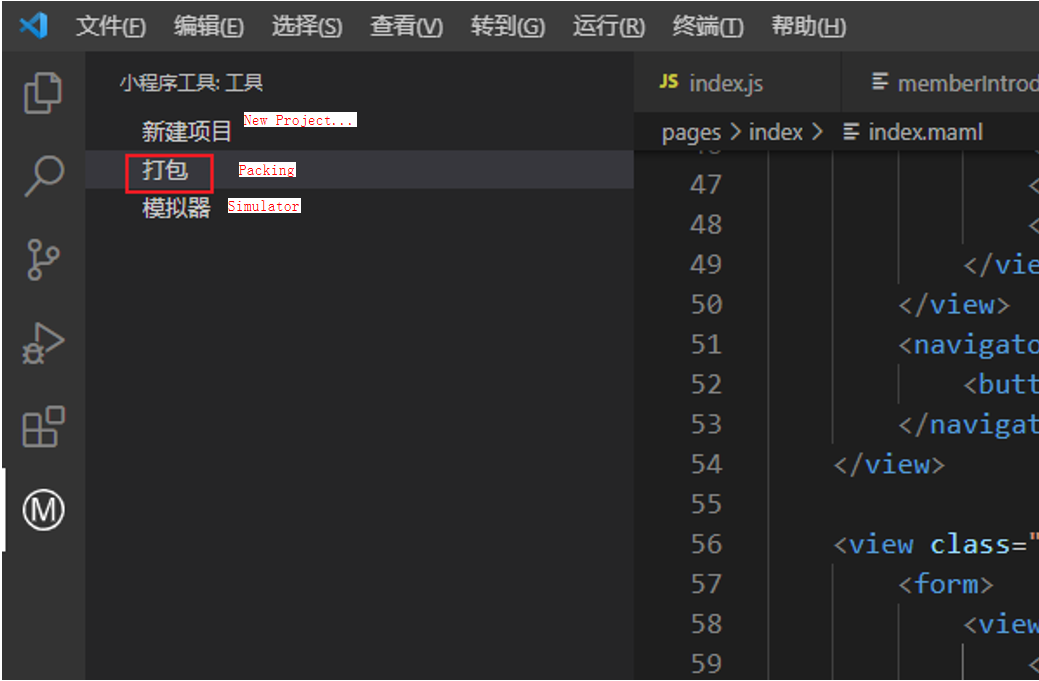


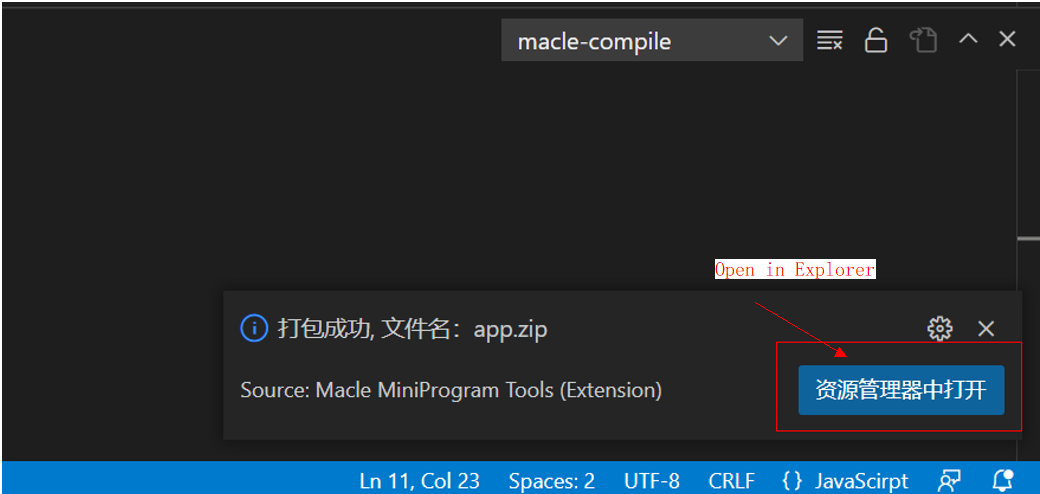
The simulator runs as follows:



## Publishing Applets

Packing Applets with Plug-in Tools



After the packaging is successful, the resource administrator is displayed in the lower right corner.

Click this button to open the app release directory：

C:\Users\yourname\.macle\target\compile\helloword\app

## Basic Functions of Applets

We add a button, click it, and change hello world to hello John!

We add a button to index.maml and use bindtap to bind a function called sayHello.

<!--index.maml-->

<view class="container">

    <view>

        <text>{{motto}}</text>

    </view>

    <view>

        <button bindtap='sayHello'>Click</button>

    </view>

</view>

Add the sayHello method to the index.js file. The method is to change the value of motto to Hello John.

// index.js

Page({

  data: {

    motto: 'Hello World'

  },

  onLoad(options) { },

  onReady() { },

  onShow() { },

  onHide() { },

  onUnload() { },

  onShareAppMessage() {

    return {

      title: ''

    };

  },

  sayHello() {

    console.log('Hello John');

    this.setData({

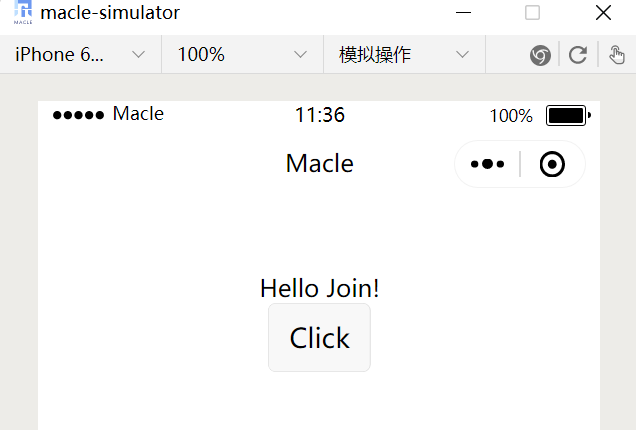
      motto: 'Hello John!'

    });

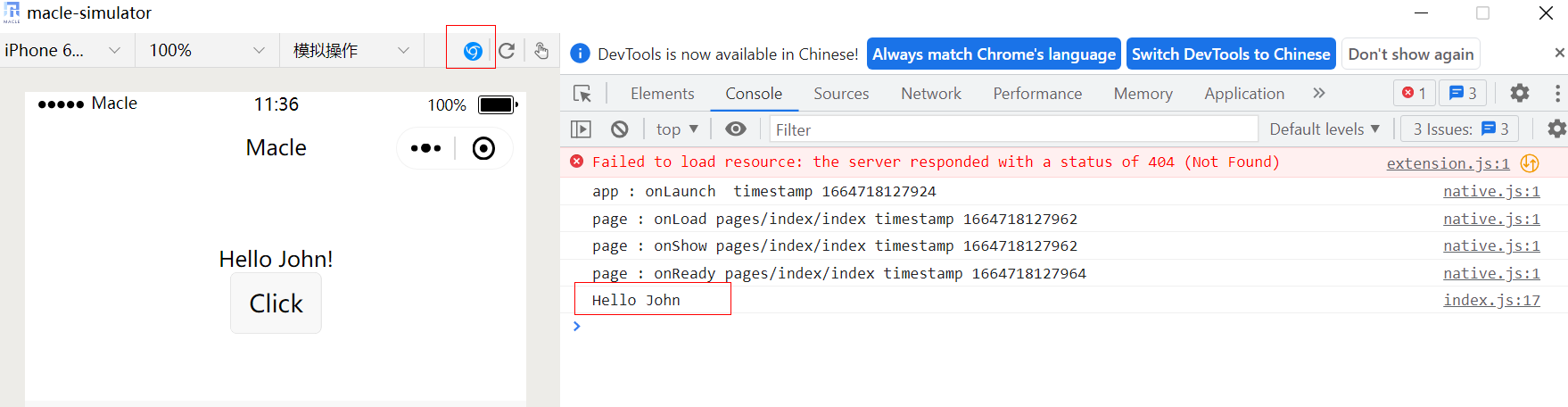
  }

});

Run the simulator and click Click. The text has changed：



The console.log method is added to the code. How to view the logs generated by the method for commissioning? Click the icon in the upper right corner of the simulator to open DevTools.



## Applet Advanced Functions

A game diamond to buy small procedures to demonstrate, introduced some advanced features of the demonstration.

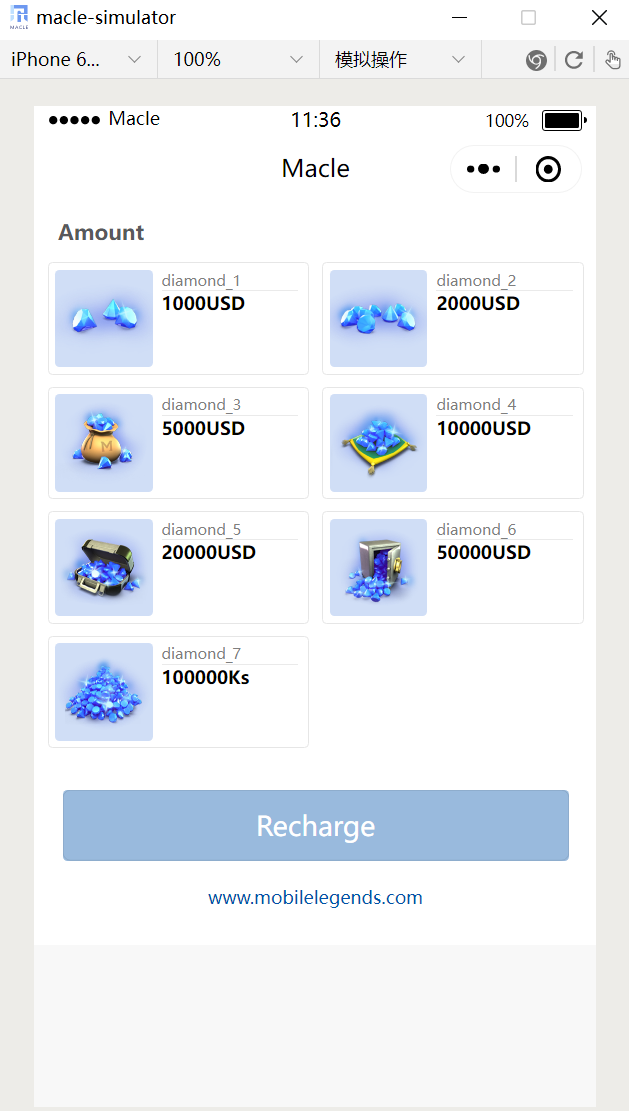
The following knowledge points are used:

1. List rendering (For details, see "MAML Syntax Reference\List Rendering" in the development help.)

2. Bind user-defined attributes and events. For details, see "Events" in the development help.

3. Conditional rendering (For details, see "MAML Syntax Reference\List Rendering" in the development help.)

4. Network request (For details, see "Applet API\Network" in the development help.)

The running UI of the applet is as follows. Users can select different prices of diamonds and purchase them.

index.maml

<!--index.maml-->

<view class="container">

    <view class="home">

        <view class="content">

            <p>Amount</p>

            <view class="amount">

                <view bind:tap="waresSelect" data-itemid="{{item.id}}" ma:for="{{waresList}}" ma:key="id" class="{{item.className}}">

                    <view class="tips">

                        <image ma:if="{{item.imageType == 1}}" src="../../res/images/diamonds\_1.png" />

                        <image ma:elif="{{item.imageType == 2}}" src="../../res/images/diamonds\_2.png" />

                        <image ma:elif="{{item.imageType == 3}}" src="../../res/images/diamonds\_3.png" />

                        <image ma:elif="{{item.imageType == 4}}" src="../../res/images/diamonds\_4.png" />

                        <image ma:elif="{{item.imageType == 5}}" src="../../res/images/diamonds\_5.png" />

                        <image ma:elif="{{item.imageType == 6}}" src="../../res/images/diamonds\_6.png" />

                        <image ma:elif="{{item.imageType == 7}}" src="../../res/images/diamonds\_7.png" />

                        <image ma:elif="{{item.imageType == 8}}" src="../../res/images/diamonds\_1.png" />

                    </view>

                    <view class="dscription">

                        <view class="bg1">{{ item.title }}</view>

                        <view class="bg2 fn2">{{ item.price }}{{ item.currency }}</view>

                    </view>

                </view>

            </view>

        </view>

        <view class="footer" id="foot">

            <button class="c" ma:if="{{selectedWaresInfo !== undefined}}" type="button" @click="createOrder">Recharge:{{ selectedWaresInfo.price }}{{ selectedWaresInfo.currency }}

            </button>

            <button class="b" ma:else type="button" id="buy">

                Recharge

            </button>

            <view class="p">www.mobilelegends.com</view>

        </view>

    </view>

</view>

index.js

// index.js

Page({

  data: {

    baseUrl: 'http://159.138.88.66:8080',

    selectedWaresInfo: undefined,

    waresList: [

      {

        id: 1,

        imageType: 1,

        price: 1000,

        currency: "USD",

        title: "diamond\_1",

        className: 'per perb'

      },

      {

        id: 2,

        imageType: 2,

        price: 2000,

        currency: "USD",

        title: "diamond\_2",

        selected: false,

        className: 'per perb'

      },

      {

        id: 3,

        imageType: 3,

        price: 5000,

        currency: "USD",

        title: "diamond\_3",

        selected: false,

        className: 'per perb'

      },

      {

        id: 4,

        imageType: 4,

        price: 10000,

        currency: "USD",

        title: "diamond\_4",

        selected: false,

        className: 'per perb'

      },

      {

        id: 5,

        imageType: 5,

        price: 20000,

        currency: "USD",

        title: "diamond\_5",

        selected: false,

        className: 'per perb'

      },

      {

        id: 6,

        imageType: 6,

        price: 50000,

        currency: "USD",

        title: "diamond\_6",

        selected: false,

        className: 'per perb'

      },

      {

        id: 7,

        imageType: 7,

        price: 100000,

        currency: "Ks",

        title: "diamond\_7",

        selected: false,

        className: 'per perb'

      },

    ],

  },

  onLoad(options) { },

  onReady() { },

  onShow() { },

  onHide() { },

  onUnload() { },

  onShareAppMessage() {

    return {

      title: ''

    };

  },

  waresSelect(e) {

    let itemid = e.currentTarget.dataset.itemid;

    let selectedItem;

    this.data.waresList.map((el) => {

      if (el.id == itemid) {

        selectedItem = el;

        el.className = 'per per-act';

      } else {

        el.className = 'per perb';

      }

    });

    this.setData({

      selectedWaresInfo: selectedItem,

      waresList: this.data.waresList

    });

  },

  buyGoods() {

    if(!this.data.selectedWaresInfo){

      ma.showToast({title:"please select a ware"});

      return;

    }

    ma.request({

      url: this.data.baseUrl + '/create/order',

      method: 'POST',

      data: {

        title: this.data.selectedWaresInfo.title,

        amount: this.data.selectedWaresInfo.price

      },

      success: (res) => {

        this.startPay(res.data);

      }

    });

  },

  startPay(rawRequest){

    ma.startPay({

      rawRequest: rawRequest,

      success: (res) => {

        ma.showToast({title:"res = " + res.resultCode});

      }

    });

  }

});

index.mass

Omitted. For details, see the code.

index.maml

The <view> tag is used throughout the page instead of the <div> tag of H5, and the syntax related to list rendering and conditional rendering is used.

### List Rendering

Before we talk about data binding, let's talk about list rendering.List rendering uses the ma:for syntax as follows:

<view bind:tap="waresSelect" data-itemid="{{item.id}}" ma:for="{{waresList}}" ma:key="id" class="{{item.className}}">

                    ……………

                </view>

Bind an array on a component using the ma:for control property to render the component repeatedly using the data for the items in the array. The default subscript variable name of the current item in the default array is index, and the default variable name of the current item in the array is item. Ma:key needs to be specified. Otherwise, an error will be reported during compilation.

### Custom Attribute and Event Bindings

With list rendering, we get a list of diamonds. We'll see that there are several other attributes in the view.

data-itemid="{{item.id}}"

We bind a custom attribute itemid to the item.id of the data.

We also found another code snippet

bind:tap="waresSelect"

Bind the tap event of the current view to the waresSelect method in index.js.

waresSelect(e) {

    let itemid = e.currentTarget.dataset.itemid;

    let selectedItem;

    this.data.waresList.map((el) => {

      if (el.id == itemid) {

        selectedItem = el;

        el.className = 'per per-act';

      } else {

        el.className = 'per perb';

      }

    });

    this.setData({

      selectedWaresInfo: selectedItem,

      waresList: this.data.waresList

    });

  },

In the wareSelect method, we can use the following statement to obtain the ID of the diamond product currently clicked, and then perform other operations.

let itemid = e.currentTarget.dataset.itemid;

### Conditional Rendering

Conditional renderingUse ma:if, ma:elif, ma:else to conditionally render

<image ma:if="{{item.imageType == 1}}" src="../../res/images/diamonds\_1.png" />

<image ma:elif="{{item.imageType == 2}}" src="../../res/images/diamonds\_2.png" />

The image tag is used to replace the img tag of H5. For details about other tags, see the help document.

### Network Requests

Network requests use the applet's ma.request API.

The network request order placement logic is implemented in the buyGoods file in the index.js file.

buyGoods() {

    if(!this.data.selectedWaresInfo){

      ma.showToast({title:"please select a ware"});

      return;

    }

    ma.request({

      url: this.data.baseUrl + '/create/order',

      method: 'POST',

      data: {

        title: this.data.selectedWaresInfo.title,

        amount: this.data.selectedWaresInfo.price

      },

      success: (res) => {

        this.startPay(res.data);

      }

    });

  },

  startPay(rawRequest){

    ma.startPay({

      rawRequest: rawRequest,

      success: (res) => {

        ma.showToast({title:"res = " + res.resultCode});

      }

    });

  }