```
  }
});
```

Parameters

The input parameters are displayed in the following table:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | The callback function for a successful API call. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function for a completed API call (Regardless of whether the call is successful or not). |

Success Callback Function

The input parameters are displayed in the following table:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | devices | Array | A list of all the devices that are discovered. |

Device Object

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | name | String | Name of the bluetooth device.(For some devices, there's no name.) | | deviceName(Compatibal with initial version) | String | Name of the bluetooth device. | | localName | String | Name of the local device. | | deviceId | String | Device ID | | RSSI | Number | Received Signal Strength Indicator | | advertisData | Hex String | Advertisement data of the device | | manufacturerData | Hex String | Manufacturer data of the device |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_bluetooth_bluetooth_getbluetoothdevices

---

my.getClipboard {#mygetclipboard}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getClipboard

2022-07-03 18:44

Get the clipboard data.

Sample Code

copy

```
Page({
  data: {
    text: '3.1415926',
    copy: '',
  },

  handlePaste() {
    my.getClipboard({
      success: ({ text }) => {
        this.setData({ copy: text });
      },
    });
  },
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | text | String | Clipboard data. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_Clipboard_getClipboard

---

my.getClipboard {#mygetclipboard}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getClipboard

2021-05-09 18:43

Get the clipboard data.

Sample Code

copy

```
Page({
  data: {
    text: '3.1415926',
    copy: '',
  },

  handlePaste() {
    my.getClipboard({
      success: ({ text }) => {
        this.setData({ copy: text });
      },
    });
  },
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | text | String | Clipboard data. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_clipboard_getclipboard

---

my.getConnectedBluetoothDevices {#mygetconnectedbluetoothdevices}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getConnectedBluetoothDevices

2022-07-03 18:44

Use this API to get the bluetooth devices that are connected.

Instructions:

- If you have searched for a Bluetooth device in the mini programbefore, you can directly pass in the deviceId obtained by the previous search to connect to the device.
- If the specified bluetooth device is connected, you'll be returned with a success response if the connection is repeated.

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
```

```
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
       <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
       <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
       <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
       <button type="primary"
```

```
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>

    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
```

```
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {
            // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];
              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
```

```
  my.stopBluetoothDevicesDiscovery({
    success: res => {
      my.offBluetoothDeviceFound();
      my.alert({ content: 'Succeeded!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain the connected device
getConnectedBluetoothDevices() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connecting devices!' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain all searched devices
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
```

```
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Disconnect the device
  disconnectBLEDevice() {
    my.disconnectBLEDevice({
      deviceId: this.data.devid,
      success: () => {
        my.alert({ content: 'Succeeded to disconnect!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the services of the connected device
  getBLEDeviceServices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Obtain the char ID of the connected device, read and write
characteristics are respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
```

```
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            //See the related document for more information of the
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.readBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          //1  Android reading service
          // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
          success: res => {
            my.alert({ content: JSON.stringify(res) });
          },
          fail: error => {
            my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
          },
        });
```

```
    },
  });
},
writeBLECharacteristicValue() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.writeBLECharacteristicValue({
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.charid,
        //Android writing service
        //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
        //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
        value: 'ABCD',
        success: res => {
          my.alert({ content: 'Succeeded to write data!' });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },
  });
},
notifyBLECharacteristicValueChange() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.notifyBLECharacteristicValueChange({
        state: true,
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.notifyId,
        success: () => {
          //Listens to characteristic change events
          my.onBLECharacteristicValueChange({
            success: res => {
              //  my.alert({content: 'Changes of
```

```
characteristics  '+JSON.stringify(res)});
                my.alert({ content: 'Obtain the response data = ' +
res.value });
              },
            });
            my.alert({ content: 'Succeeded to listen' });
          },
          fail: error => {
            my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
```

```
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChan
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Parameters

The input parameters are displayed in the following table:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | deviceId | String | Yes | Device ID of the bluetooth. | | success | Function | No | The callback function for a successful API call. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function for a completed API call (Regardless of whether the call is successful or not). |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_Bluetooth_Bluetooth_getConnectedBluetoothDevices

---

my.getConnectedBluetoothDevices {#mygetconnectedbluetoothdevices}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getConnectedBluetoothDevices

2021-05-09 18:43

Use this API to get the bluetooth devices that are connected.

Instructions:

- If you have searched for a Bluetooth device in the mini programbefore, you can directly pass in the deviceId obtained by the previous search to connect to the device.

- If the specified bluetooth device is connected, you'll be returned with a success response if the connection is repeated.

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
```

```
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
       <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
       <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
       <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
       <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>

    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
```

```
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {
            // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];
              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
```

```
//Obtain the connected device
getConnectedBluetoothDevices() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connecting devices!' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain all searched devices
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Disconnect the device
disconnectBLEDevice() {
  my.disconnectBLEDevice({
```

```
      deviceId: this.data.devid,
      success: () => {
        my.alert({ content: 'Succeeded to disconnect!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the services of the connected device
  getBLEDeviceServices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Obtain the char ID of the connected device, read and write
characteristics are respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
```

```
            //See the related document for more information of the
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.readBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          //1  Android reading service
          // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
          success: res => {
            my.alert({ content: JSON.stringify(res) });
          },
          fail: error => {
            my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  writeBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
```

```
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.writeBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.charid,
            //Android writing service
            //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
            value: 'ABCD',
            success: res => {
              my.alert({ content: 'Succeeded to write data!' });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },
    notifyBLECharacteristicValueChange() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.notifyBLECharacteristicValueChange({
            state: true,
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.notifyId,
            success: () => {
              //Listens to characteristic change events
              my.onBLECharacteristicValueChange({
                success: res => {
                  //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                  my.alert({ content: 'Obtain the response data = ' +
res.value });
                },
              });
              my.alert({ content: 'Succeeded to listen' });
            },
            fail: error => {
              my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
```

```
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChang
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
```

```
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Parameters

The input parameters are displayed in the following table:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | deviceId | String | Yes | Device ID of the bluetooth. | | success | Function | No | The callback function for a successful API call. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function for a completed API call (Regardless of whether the call is successful or not). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_bluetooth_getconnectedbluetoothdevices

---

my.getFileInfo {#mygetfileinfo}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getFileInfo

2022-07-03 18:44

Get file information.

Sample Code

copy

```
my.getFileInfo({
    apFilePath:
'https://resource/apml953bb093ebd2834530196f50a4413a87.video',
    digestAlgorithm: 'sha1',
    success: (res)=> {
        console.log(JSON.stringify(res))
    }
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | apFilePath | String | Yes | File path. | | digestAlgorithm | String | No | Digest algorithm, supporting `md5` and `sha1`, `md5` by default. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | size | Number | File size. | | digest | String | Digest result. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_File_getFileInfo

---

my.getFileInfo {#mygetfileinfo}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getFileInfo

2021-05-09 18:43

Get file information.

Sample Code

copy

```
my.getFileInfo({
    apFilePath:
'https://resource/apml953bb093ebd2834530196f50a4413a87.video',
    digestAlgorithm: 'sha1',
    success: (res)=> {
        console.log(JSON.stringify(res))
    }
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | apFilePath | String | Yes | File path. | | digestAlgorithm | String | No | Digest algorithm, supporting `md5` and `sha1`, `md5` by default. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | size | Number | File size. | | digest | String | Digest result. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_file_getfileinfo

---

my.getImageInfo {#mygetimageinfo}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getImageInfo

2021-05-09 18:43

Get picture information.

Sample Code

copy

```
<!-- .axml -->
<view class="page">
  <view class="page-description">Get picture info API</view>
  <view class="page-section">
    <view class="page-section-title">my.getImageInfo</view>
    <view class="page-section-demo">
      <image src="{{src}}" onError="imageError" onLoad="imageLoad" />
      <button type="primary" onTap="getImageInfo">Get picture
info</button>
    </view>
```

```
  </view>
</view>
```

copy

```
//.js
// Network picture path
my.getImageInfo({
  src:'https://img.example.com/example.jpg',
  success:(res)=>{
    console.log(JSON.stringify(res))
  }
})

//apFilePath
my.chooseImage({
  success: (res) => {
    my.getImageInfo({
      src:res.apFilePaths[0],
      success:(res)=>{
        console.log(JSON.stringify(res))
      }
    })
  },
})

//Relative path
my.getImageInfo({
  src:'image/api.png',
  success:(res)=>{
    console.log(JSON.stringify(res))
  }
})
```

Parameters

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | src | String | No | Picture path, supporting network picture path, apFilePath path and relative path. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | width | Number | Picture width (in px). | | height | Number | Picture height (in px). | | path | String | Local path of picture. | | orientation | String | Return picture orientation. Effective values are listed below. | | type |

String | Return picture format. |

Orientation Parameter Description

| | | --- | --- | | **Enumerator** | **Description** | | up | Default. | | down | 180-Degree rotation. | | left | Rotate by 90 degree counterclockwise. | | right | Rotate by 90 degree clockwise. | | up-mirrored | Same as up except for flipping horizontally. | | down-mirrored | Same as down except for flipping horizontally. | | left-mirrored | Same as left except for flipping vertically. | | right-mirrored | Same as right except for flipping vertically. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_media_image_getimageinfo

---

my.getImageInfo {#mygetimageinfo}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getImageInfo

2022-07-03 18:44

Get picture information.

Sample Code

copy

```
<!-- .axml -->
<view class="page">
  <view class="page-description">Get picture info API</view>
  <view class="page-section">
    <view class="page-section-title">my.getImageInfo</view>
    <view class="page-section-demo">
      <image src="{{src}}" onError="imageError" onLoad="imageLoad" />
      <button type="primary" onTap="getImageInfo">Get picture
info</button>
    </view>
  </view>
</view>
```

copy

```
//.js
// Network picture path
```

```
my.getImageInfo({
  src:'https://img.example.com/example.jpg',
  success:(res)=>{
    console.log(JSON.stringify(res))
  }
})

//apFilePath
my.chooseImage({
  success: (res) => {
    my.getImageInfo({
      src:res.apFilePaths[0],
      success:(res)=>{
        console.log(JSON.stringify(res))
      }
    })
  },
})

//Relative path
my.getImageInfo({
  src:'image/api.png',
  success:(res)=>{
    console.log(JSON.stringify(res))
  }
})
```

Parameters

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | src | String | No | Picture path, supporting network picture path, apFilePath path and relative path. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | width | Number | Picture width (in px). | | height | Number | Picture height (in px). | | path | String | Local path of picture. | | orientation | String | Return picture orientation. Effective values are listed below. | | type | String | Return picture format. |

Orientation Parameter Description

| | | --- | --- | | **Enumerator** | **Description** | | up | Default. | | down | 180-Degree rotation. | | left | Rotate by 90 degree counterclockwise. | | right | Rotate by 90 degree clockwise. | | up-mirrored | Same as up except for flipping horizontally. | | down-mirrored | Same as down except for flipping horizontally. | | left-mirrored | Same as left except for flipping vertically. | | right-mirrored | Same as right except for flipping vertically. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_media_image_getimageinfo

---

my.getLocation {#mygetlocation}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getLocation

2022-07-03 18:44

Get the current geographical location of the user.

Sample Code

copy

```
my.getLocation({
  success(res) {
    my.hideLoading();
    console.log(res)
    that.setData({
      hasLocation: true,
      location: formatLocation(res.longitude, res.latitude)
    })
  },
  fail() {
    my.hideLoading();
    my.alert({ title: 'location failed' });
  },
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | cacheTimeout | Number | No | longitude and latitude location cache expiry time in seconds. Default is 30s. Use of cache can speed up location process. Re-location is done upon cache expiry. | | type | Number | No | 0: default, get the longitude and latitude. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | longitude | String | Longitude. | | latitude | String | Latitude. | | accuracy | String | Accuracy, in m. |

Error Code

| | | | | --- | --- | --- | | **Error** | **Description** | **Solution** | | 11 | Make sure the location related right has been enabled. | Prompt the user to enable location permission. | | 12 | Network abnormity, try again later. | Prompt the user to check the current network. | | 13 | Location failure, try again later. | - | | 14 | Service location timeout. | Prompt the user to try again. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Location_getLocation

---

my.getLocation {#mygetlocation}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getLocation

2021-05-09 18:43

Get the current geographical location of the user.

Sample Code

copy

```
my.getLocation({
  success(res) {
    my.hideLoading();
    console.log(res)
    that.setData({
      hasLocation: true,
      location: formatLocation(res.longitude, res.latitude)
    })
  },
  fail() {
    my.hideLoading();
    my.alert({ title: 'location failed' });
  },
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | cacheTimeout | Number | No | longitude and latitude location cache expiry time in seconds. Default is 30s. Use of cache can speed up location process. Re-location is done upon cache expiry. | | type | Number | No | 0: default, get the longitude and latitude. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | longitude | String | Longitude. | | latitude | String | Latitude. | | accuracy | String | Accuracy, in m. |

Error Code

| | | | --- | --- | --- | | **Error** | **Description** | **Solution** | | 11 | Make sure the location related right has been enabled. | Prompt the user to enable location permission. | | 12 | Network abnormity, try again later. | Prompt the user to check the current network. | | 13 | Location failure, try again later. | - | | 14 | Service location timeout. | Prompt the user to try again. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_location_getlocation

---

my.getNetworkType {#mygetnetworktype}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getNetworkType

2022-07-03 18:44

Get the current network status.

Sample Code

copy

```
Page({
  data: {
    hasNetworkType: false
  },
  getNetworkType() {
    my.getNetworkType({
      success: (res) => {
        this.setData({
          hasNetworkType: true,
          networkType: res.networkType
        })
      }
    })
  },
  clear() {
    this.setData({
      hasNetworkType: false,
      networkType: ''
    })
  },
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | networkAvailable | Boolean | If the network is available. | | networkType | String | Network type, UNKNOWN / NOTREACHABLE / WIFI / 3G / 2G / 4G / WWAN. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_network_getnetworktype

---

my.getNetworkType {#mygetnetworktype}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getNetworkType

2021-05-09 18:43

Get the current network status.

Sample Code

copy

```
Page({
  data: {
    hasNetworkType: false
  },
  getNetworkType() {
    my.getNetworkType({
      success: (res) => {
        this.setData({
          hasNetworkType: true,
          networkType: res.networkType
        })
      }
    })
  },
  clear() {
    this.setData({
      hasNetworkType: false,
      networkType: ''
    })
  },
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | networkAvailable | Boolean | If the network is available. | | networkType | String | Network type, UNKNOWN / NOTREACHABLE / WIFI / 3G / 2G / 4G / WWAN. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_network_getnetworktype

---

my.getOpenUserInfo {#mygetopenuserinfo}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getOpenUserInfo

2022-07-03 18:44

Get the basic information about a user. This feature requires the user to deliberately trigger to activate the function. This function is not directly called by the API but rather waits for when the user has activated it by clicking a `<button>` component. If the Mini Program wants to get userId, please call `my.getAuthCode`.

For more information, please refer to the obtain basic member information.

Use Attention

You need to set the value of the `<button>` component `open-type` to `getAuthorize` and set the value of the `scope` to `userInfo` . After the user clicks the authorization button, the Mini Program can get the user information returned by the `my.getOpenUserInfo` JSAPI.

`my.getOpenUserInfo` will send a network request to the server to obtain user information, so it may be take some time before the callback function invoked.

Parameters

| | | | | --- | --- | --- | --- | | **Name** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Name** | **Type** | **Required** | **Description** | | code | String | NO | The result code. | | msg | String | NO | The result message. | | avatar | String | NO | User avatar image url. | | nickName | String | NO | User nickName. | | gender | String | NO | User gender. "m" is male， "f" is female. | | countryCode | String | NO | The code of the country where user is located. it should follow ISO 3166-1 alpha-2 code standard, such as 'US', 'SG'. | | province | String | NO | The province where user is located. | | city | String | NO | The city where user is located. |

These fields are returned every time, but it will be an empty string if the app does not return the related information.

The maximum length of these fields are 128 bytes except avatar， the maximum length of avatar is 2048 bytes.

Sample Code

copy

```
<!-- .axml -->
<button
    a:if="{{canIUseAuthButton}}"
    open-type="getAuthorize"
    onGetAuthorize="onGetAuthorize"
    onError="onAuthError"
    scope='userInfo'>
</button>
```

Button Property Description

| | | | --- | --- | | **Name** | **Description** | | open-type | `getAuthorize`(Must be this value). | | scope | `userInfo`(Must be this value). | | onGetAuthorize | Authorization success callback (The Mini Program can call my.getOpenUserInfo to get information in this callback). | | onError | Authorization failure callback (Including user rejection and system exceptions). |

Get User Basic Information

After the user clicks the consent, the user basic information can be obtained through my.getOpenUserInfo().

copy

```
// .js
onGetAuthorize(res) {
    my.getOpenUserInfo({
        fail: (res) => {
        },
        success: (res) => {
            let userInfo = JSON.parse(res.response).response
        }
    });
}
```

Return Res Object In the Success Callback

- An example of a successfully res object returned is as follows:

copy

```
{
    "response": "{\"response\": {\"code\": \"10000\",\"msg\":
\"Success\",\"countryCode\": \"code\",\"gender\": \"f\",\"nickName\":
\"XXX\",\"avatar\": \"https://cdn/images/partner/XXXXXXXX\",\"city\":
\"city\",\"province\": \"province\"}}"
}
```

- If the function package of "Get Basic User Information" is not connected, the format example of returned res message is as follows:

copy

```
{
    "response":"{\"response\":
{\"code\":\"40006\",\"msg\":\"Insufficient
Permissions\",\"subCode\":\"isv.insufficient-isv-
permissions\",\"subMsg\": \"Insufficient permissions\"}}"
}
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_OpenAPI_getOpenUserInfo

---

my.getOpenUserInfo {#mygetopenuserinfo}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getOpenUserInfo

2021-05-09 18:43

Get the basic information about a user. This feature requires the user to deliberately trigger to activate the function. This function is not directly called by the API but rather waits for when the user has activated it by clicking a `<button>` component. If the Mini Program wants to get userId, please call `my.getAuthCode`.

For more information, please refer to the obtain basic member information.

Use Attention

You need to set the value of the `<button>` component `open-type` to `getAuthorize` and set the value of the `scope` to `userInfo` . After the user clicks the authorization button, the Mini Program can get the user information returned by the `my.getOpenUserInfo` JSAPI.

`my.getOpenUserInfo` will send a network request to the server to obtain user information, so it may be take some time before the callback function invoked.

Parameters

| | | | | --- | --- | --- | --- | | **Name** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Name** | **Type** | **Required** | **Description** | | code | String | NO | The result code. | | msg | String | NO | The result message. | | avatar | String | NO | User avatar image url. | | nickName | String | NO | User nickName. | | gender | String | NO | User gender. "m" is male， "f" is female. | | countryCode | String | NO | The code of the country where user is located. it should follow ISO 3166-1 alpha-2 code standard, such as 'US', 'SG'. | | province | String | NO | The province where user is located. | | city | String | NO | The city where user is located. |

These fields are returned every time, but it will be an empty string if the app does not return the related information.

The maximum length of these fields are 128 bytes except avatar， the maximum length of avatar is 2048 bytes.

Sample Code

copy

```
<!-- .axml -->
<button
    a:if="{{canIUseAuthButton}}"
    open-type="getAuthorize"
    onGetAuthorize="onGetAuthorize"
    onError="onAuthError"
    scope='userInfo'>
</button>
```

Button Property Description

| | | --- | --- | | **Name** | **Description** | | open-type | `getAuthorize`(Must be this value). | | scope | `userInfo`(Must be this value). | | onGetAuthorize | Authorization success callback (The Mini Program can call my.getOpenUserInfo to get information in this callback). | | onError | Authorization failure callback (Including user rejection and system exceptions). |

Get User Basic Information

After the user clicks the consent, the user basic information can be obtained through my.getOpenUserInfo().

copy

```
// .js
onGetAuthorize(res) {
    my.getOpenUserInfo({
        fail: (res) => {
        },
        success: (res) => {
            let userInfo = JSON.parse(res.response).response
        }
    });
}
```

Return Res Object In the Success Callback

- An example of a successfully res object returned is as follows:

copy

```
{
    "response": "{\"response\": {\"code\": \"10000\",\"msg\":
\"Success\",\"countryCode\": \"code\",\"gender\": \"f\",\"nickName\":
\"XXX\",\"avatar\": \"https://cdn/images/partner/XXXXXXXX\",\"city\":
\"city\",\"province\": \"province\"}}"
}
```

- If the function package of "Get Basic User Information" is not connected, the format example of returned res message is as follows:

copy

```
{
    "response":"{\"response\":
{\"code\":\"40006\",\"msg\":\"Insufficient
Permissions\",\"subCode\":\"isv.insufficient-isv-
permissions\",\"subMsg\": \"Insufficient permissions\"}}"
}
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_openapi_getopenuserinfo

---

my.getRunScene {#mygetrunscene}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getRunScene

2021-05-09 18:43

Use this API to obtain the running version of the current Mini Program.

Sample Code

copy

```
my.getRunScene({
  success(result) {
     my.alert({
      title: 'Mini Program version',
      content:`${result.envVersion}`
    });
  },
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | The callback function for a successful API call. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function used

when the API call is completed. This function is always executed no matter the call succeeds or fails. |

Success Callback Function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | envVersion | String | The current running version of the Mini Program. Valid values are:
- `develop`: development version
- `release`: release version |

Fail Callback Function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | error | String | The error code. | | errorMessage | String | The error message. |

Error Code

| | | | --- | --- | | **Error Code** | **Description** | | 3 | An unknown error has occurred. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_basic_getrunscene

---

my.getRunScene {#mygetrunscene}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getRunScene

2022-07-03 18:44

Use this API to obtain the running version of the current Mini Program.

Sample Code

copy

```
my.getRunScene({
  success(result) {
     my.alert({
      title: 'Mini Program version',
      content:`${result.envVersion}`
    });
```

```
  },
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | The callback function for a successful API call. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function used when the API call is completed. This function is always executed no matter the call succeeds or fails. |

Success Callback Function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | envVersion | String | The current running version of the Mini Program. Valid values are:
- `develop`: development version
- `release`: release version |

Fail Callback Function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | error | String | The error code. | | errorMessage | String | The error message. |

Error Code

| | | | --- | --- | | **Error Code** | **Description** | | 3 | An unknown error has occurred. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_basic_getrunscene

---

my.getSavedFileInfo {#mygetsavedfileinfo}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getSavedFileInfo

2021-05-09 18:43

Get saved file information.

Sample Code

The my.saveFile saved address is required to use my.getSavedFileInfo

copy

```
var that = this;
my.chooseImage({
  success: (res) => {
    console.log(res.apFilePaths[0], 1212)
    my.saveFile({
      apFilePath: res.apFilePaths[0],
      success: (result) => {
        console.log(result, 1212)
        my.getSavedFileInfo({
          apFilePath: result.apFilePath,
          success: (resu) => {
            console.log(JSON.stringify(resu))
            that.filePath = resu
          }
        })
      },
    });
  },
});
```

Parameters

Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | apFilePath | String | Yes | File path. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | size | Number | File size. | | createTime | Number | Timestamp for the created time. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_file_getsavedfileinfo

---

my.getSavedFileInfo {#mygetsavedfileinfo}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getSavedFileInfo

2022-07-03 18:44

Get saved file information.

Sample Code

The my.saveFile saved address is required to use my.getSavedFileInfo

copy

```
var that = this;
my.chooseImage({
  success: (res) => {
    console.log(res.apFilePaths[0], 1212)
    my.saveFile({
      apFilePath: res.apFilePaths[0],
      success: (result) => {
        console.log(result, 1212)
        my.getSavedFileInfo({
          apFilePath: result.apFilePath,
          success: (resu) => {
            console.log(JSON.stringify(resu))
            that.filePath = resu
          }
        })
      },
    });
  },
});
```

Parameters

Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | apFilePath | String | Yes | File path. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | size | Number | File size. | | createTime | Number | Timestamp for the created time. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_File_getSavedFileInfo

---

my.getSavedFileList {#mygetsavedfilelist}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getSavedFileList

2021-05-09 18:43

Get information of all saved files.

Sample Code

copy

```
my.getSavedFileList({
  success:(res)=>{
    console.log(JSON.stringfy(res))
  }
});
```

Parameters

Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | fileList | List | File list. |

File Object Attribute

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | size | Number | File size. | | createTime | Number | Created time. | | apFilePath | String | File path. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_file_getsavedfilelist

---

my.getSavedFileList {#mygetsavedfilelist}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getSavedFileList

2022-07-03 18:44

Get information of all saved files.

Sample Code

copy

```
my.getSavedFileList({
  success:(res)=>{
    console.log(JSON.stringify(res))
  }
});
```

Parameters

Object type with the following attributes:

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | fileList | List | File list. |

File Object Attribute

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | size | Number | File size. | | createTime | Number | Created time. | | apFilePath | String | File path. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_File_getSavedFileList

---

my.getScreenBrightness {#mygetscreenbrightness}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getScreenBrightness

2021-05-09 18:43

Get screen brightness

Sample Code

copy

```
<!-- API-DEMO page/API/screen/screen.axml-->
<view class="page">
  <view class="page-description">Screen brightness API</view>
  <view class="page-section">
    <view class="page-section-title">Set whether to keep screen
on</view>
    <view class="page-section-demo">
      <switch checked="{{status}}" onChange="switchKeepScreenOn"/>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">Set screen brightness</view>
    <view class="page-section-demo">
      <slider value="{{brightness}}" max="1" min="0"
onChange="sliderChange" step="0.02"/>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">Get screen brightness</view>
    <view class="page-section-demo">
      <button type="primary" onTap="getBrightness">Get screen
```

```
brightness</button>
    </view>
  </view>
</view>
```

copy

```
// API-DEMO page/API/screen/screen.js
Page({
  data: {
    status: false,
    brightness: 1,
  },
  onLoad() {
    my.getScreenBrightness({
      success: res => {
        this.setData({
          brightness: res.brightness
        })
      },
    })
  },
  sliderChange(e) {
    my.setScreenBrightness({
      brightness: e.detail.value,
      success: (res) => {
        this.setData({
          brightness: e.detail.value,
        })
      }
    })
  },
  switchKeepScreenOn(e) {
    my.setKeepScreenOn({
      keepScreenOn: e.detail.value,
      success: (res) => {
        this.setData({
          status: e.detail.value,
        })
      }
    })
  },
  getBrightness() {
    my.getScreenBrightness({
      success: res => {
        my.alert({
          content: `Current screen brightness: ${res.brightness}`
        });
      }
    })
  }
});
```

Parameters

Object type with the following attributes:

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_screen_getscreenbrightness

---

my.getScreenBrightness {#mygetscreenbrightness}

Last updated: 2022-07-04

Path: miniprogram_gcash

my.getScreenBrightness

2022-07-04 03:44

Get screen brightness

Sample Code

copy

```
<!-- API-DEMO page/API/screen/screen.axml-->
<view class="page">
  <view class="page-description">Screen brightness API</view>
  <view class="page-section">
    <view class="page-section-title">Set whether to keep screen
on</view>
    <view class="page-section-demo">
      <switch checked="{{status}}" onChange="switchKeepScreenOn"/>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">Set screen brightness</view>
    <view class="page-section-demo">
      <slider value="{{brightness}}" max="1" min="0"
onChange="sliderChange" step="0.02"/>
    </view>
```

```
  </view>
  <view class="page-section">
    <view class="page-section-title">Get screen brightness</view>
    <view class="page-section-demo">
      <button type="primary" onTap="getBrightness">Get screen
brightness</button>
    </view>
  </view>
</view>
```

copy

```
// API-DEMO page/API/screen/screen.js
Page({
  data: {
    status: false,
    brightness: 1,
  },
  onLoad() {
    my.getScreenBrightness({
      success: res => {
        this.setData({
          brightness: res.brightness
        })
      },
    })
  },
  sliderChange(e) {
    my.setScreenBrightness({
      brightness: e.detail.value,
      success: (res) => {
        this.setData({
          brightness: e.detail.value,
        })
      }
    })
  },
  switchKeepScreenOn(e) {
    my.setKeepScreenOn({
      keepScreenOn: e.detail.value,
      success: (res) => {
        this.setData({
          status: e.detail.value,
        })
      }
    })
  },
  getBrightness() {
    my.getScreenBrightness({
      success: res => {
        my.alert({
          content: `Current screen brightness: ${res.brightness}`
```

```
        });
      }
    })
  }
});
```

Parameters

Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_screen_getscreenbrightness

---

my.getScreenOrientation {#mygetscreenorientation}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getScreenOrientation

2022-07-03 18:44

Call this API to get screen orientation.

Sample code

copy

```
my.getScreenOrientation({
  success: (res) => {
    my.alert({
      title: 'success',
      content: JSON.stringify(res)
    })
  },
  fail: (res) => {
    my.alert({
      title: 'fail',
```

```
      content: JSON.stringify(res)
    })
  }
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success | | fail | Function | No | Callback function upon call failure | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure) |

Success callback function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | success | Boolean | Specifies whether the call is successful. When the value is `true`, the call is successful. | | orientation | String | Indicates the orientation of the screen, `portrait` or `landscape`. |

Fail callback function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | error | Number | The error code for the failure | | errorMessage | String | The error message |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_screen_getscreenorientation

---

my.getServerTime {#mygetservertime}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getServerTime

2021-05-09 18:43

Get current server time in milliseconds

Sample Code

copy

```
<!-- API-DEMO page/API/get-server-time/get-server-time.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-btns">
      <view onTap="getServerTime">Get server time </view>
    </view>
  </view>
</view>
```

copy

```
// API-DEMO page/API/get-server-time/get-server-time.js
Page({
  getServerTime(){
    my.getServerTime({
      success: (res) => {
        my.alert({
          content: res.time,
        });
      },
    });
  }
})
```

Parameters

Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | time | Number | Get current server time. A numerical value is returned, indicating the milliseconds since 0:0:0 January 1 1970 (UTC). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_server_getservertime

---

my.getServerTime {#mygetservertime}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getServerTime

2022-07-03 18:44

Get current server time in milliseconds

Sample Code

copy

```
<!-- API-DEMO page/API/get-server-time/get-server-time.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-btns">
      <view onTap="getServerTime">Get server time </view>
    </view>
  </view>
</view>
```

copy

```
// API-DEMO page/API/get-server-time/get-server-time.js
Page({
  getServerTime(){
    my.getServerTime({
      success: (res) => {
        my.alert({
          content: res.time,
        });
      },
    });
  }
})
```

Parameters

Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | time | Number | Get current server time. A numerical value is returned, indicating the milliseconds since 0:0:0 January 1 1970 (UTC). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_Server_getServerTime

---

my.getSetting {#mygetsetting}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getSetting

2021-05-09 18:43

Use this API to obtain the user's current settings. Only the permissions that have been requested by the Mini Program from the user are returned.

Sample Code

copy

```
my.getSetting({
  success: (res) => {
    /*
     * res.authSetting = {
     *   "location": true,
     *   "audioRecord": true,
     *   ...
     * }
     */
  }
})
```

Parameters

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | The callback function for a successful API call. See Sample Return Value for details. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function used when the API call is completed. This function is always executed no matter the call succeeds or fails. |

Success callback function

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | authSetting | Object | Results of user authorization. Keys are the values of `scopes` and values are boolean types, which shows whether the user gives the permission or not. See Scopes for details. |

Return Value Sample

copy

```
{
    "authSetting": {
        "camera": true,
        "location": true,
        "album": true,
        "userInfo": true,
        "phoneNumber": true
    }
}
```

Scopes

| | | | --- | --- | --- | | **Scope** | **API** | **Description** | | location | my.getLocation | This field specifies whether to authorize access to geographic location. | | album | my.chooseImage、 my.saveImage | This field specifies whether to authorize to save images to the albums. | | camera | my.scan | This field specifies whether to authorize access to camera. | | phoneNumber | my.getPhoneNumber | This field specifies whether to authorize access to phone number. | | userInfo | my.getOpenUserInfo | This field specifies whether to authorize access to user information. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_setting_getsetting

---

my.getSetting {#mygetsetting}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getSetting

2022-07-03 18:44

Use this API to obtain the user's current settings. Only the permissions that have been requested by the Mini Program from the user are returned.

Sample Code

copy

```
my.getSetting({
  success: (res) => {
    /*
     * res.authSetting = {
     *   "location": true,
     *   "audioRecord": true,
     *   ...
     * }
     */
  }
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | The callback function for a successful API call. See Sample Return Value for details. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function used when the API call is completed. This function is always executed no matter the call succeeds or fails. |

Success callback function

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | authSetting | Object | Results of user authorization. Keys are the values of `scopes` and values are boolean types, which shows whether the user gives the permission or not. See Scopes for details. |

Return Value Sample

copy

```
{
    "authSetting": {
        "camera": true,
        "location": true,
        "album": true,
        "userInfo": true,
        "phoneNumber": true
    }
}
```

Scopes

| | | | --- | --- | --- | | **Scope** | **API** | **Description** | | location | my.getLocation | This field specifies whether to authorize access to geographic location. | | album | my.chooseImage、 my.saveImage | This field specifies whether to authorize to save images to the albums. | | camera | my.scan | This field specifies whether to authorize access to camera. | | phoneNumber | my.getPhoneNumber | This field specifies whether to authorize access to phone number. | | userInfo | my.getOpenUserInfo | This field specifies whether to authorize access to user information. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_Setting_getSetting

---

my.getSiteInfo {#mygetsiteinfo}

Last updated: 2022-07-25

Path: miniprogram_gcash

my.getSiteInfo

2022-07-25 00:03

Use this API to obtain the site information.

Note:

Please make sure you use the Appx with 1.24.6 or higher versions in order to use this API.

Sample code

copy

```
my.getSiteInfo({
  success: (res) => {
    my.alert({
      content: JSON.stringify(res),
    });
  },
  fail: (res) => {
    my.alert({
      content: JSON.stringify(res),
    });
```

```
  }
});
```

Input Parameters

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | siteName | String | Following values are supported:
GCASH |

An example of a successfully returned message is as follows:

copy

```
{
  "siteName":"GCASH"
}
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_alipay-connect_getsiteinfo

---

my.getSiteInfo {#mygetsiteinfo}

Last updated: 2023-01-29

Path: miniprogram_gcash

my.getSiteInfo

2023-01-29 20:55

Use this API to obtain the site information.

Note:

Please make sure you use the Appx with 1.24.6 or higher versions in order to use this API.

Sample code

copy

```
my.getSiteInfo({
  success: (res) => {
    my.alert({
      content: JSON.stringify(res),
    });
  },
  fail: (res) => {
    my.alert({
      content: JSON.stringify(res),
    });
  }
});
```

Input Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | siteName | String | Following values are supported:
GCASH |

An example of a successfully returned message is as follows:

copy

```
{
  "siteName":"GCASH"
}
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_alipay-connect_getsiteinfo

---

my.getStorage {#mygetstorage}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getStorage

2021-05-09 18:43

Get cached data.

> This is an asynchronous interface.
>
> support the isolation between embedded webview cache and Mini Program cache. Getting the cache of the specified key of embedded webview will not return the cached data of the same key of the Mini Program.

Sample Code

copy

```
my.getStorage({
  key: 'currentCity',
  success: function(res) {
    my.alert({content: 'Success' + res.data.cityName});
  },
  fail: function(res){
    my.alert({content: res.errorMessage});
  }
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | key | String | Yes | Cache data key. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | data | Object/String | Corresponding content of the key. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_storage_getstorage

---

my.getStorage {#mygetstorage}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getStorage

2022-07-03 18:44

Get cached data.

> This is an asynchronous interface.
>
> support the isolation between embedded webview cache and Mini Program cache. Getting the cache of the specified key of embedded webview will not return the cached data of the same key of the Mini Program.

Sample Code

copy

```
my.getStorage({
  key: 'currentCity',
  success: function(res) {
    my.alert({content: 'Success' + res.data.cityName});
  },
  fail: function(res){
    my.alert({content: res.errorMessage});
  }
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | key | String | Yes | Cache data key. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | data | Object/String | Corresponding content of the key. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Storage_getStorage

---

my.getStorageSync {#mygetstoragesync}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.getStorageSync

2021-05-09 18:43

Get cached data synchronously.

> This is a synchronous interface.

Sample Code

copy

```
let res = my.getStorageSync({ key: 'currentCity' });
 my.alert({
    content: JSON.stringify(res.data),
 });
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | key | String | Yes | Cache data key. |

Return Value

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | data | Object/String | Corresponding content of the key. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_storage_getstoragesync

---

my.getStorageSync {#mygetstoragesync}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getStorageSync

2022-07-03 18:44

Get cached data synchronously.

> This is a synchronous interface.

Sample Code

copy

```
let res = my.getStorageSync({ key: 'currentCity' });
 my.alert({
    content: JSON.stringify(res.data),
 });
```

Parameters

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | key | String | Yes | Cache data key. |

Return Value

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | data | Object/String | Corresponding content of the key. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Storage_getStorageSync

---

my.getSystemInfo {#mygetsysteminfo}

Last updated: 2023-01-29

Path: miniprogram_gcash

my.getSystemInfo

2023-01-29 20:55

Get system information.

Sample Code

copy

```
Page({
  data: {
    systemInfo: {}
  },
  getSystemInfoPage() {
    my.getSystemInfo({
      success: (res) => {
        this.setData({
          systemInfo: res
        })
      }
    })
  },
})
```

Parameters

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success | | fail | Function | No | Callback function upon call failure | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure.) |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | model | String | Cellphone model | | pixelRatio | Number | Device pixel ratio | | windowWidth | Number | Window width | | windowHeight | Number | Window height | | language | String | The language set by the user in the app. If the app does not support the language setting, return the system language. | | version | String | App version number | | storage | String | Device disk capacity | | currentBattery | String | Current battery percentage | | system | String | System version | | platform | String | System name: `Android`, iOS | | titleBarHeight | Number | Title bar height | | statusBarHeight | Number | Status bar height | | screenWidth | Number | Screen width | | screenHeight | Number | Screen height | | brand | String | Cellphone brand | | fontSizeSetting | Number | User setting font size | | app | String | Current running client. The app value can refer to the following table. |

App Value Reference Table

| | | | --- | --- | | **App** | **Value** | | GCash | gcash |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_System_getSystemInfo

---

my.getSystemInfo {#mygetsysteminfo}

Last updated: 2022-07-24

Path: miniprogram_gcash

my.getSystemInfo

2022-07-24 23:36

Get system information

Sample Code

copy

```
Page({
  data: {
    systemInfo: {}
  },
  getSystemInfoPage() {
    my.getSystemInfo({
      success: (res) => {
        this.setData({
          systemInfo: res
        })
      }
    })
  },
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | model | String | Cellphone model. | | pixelRatio | Number | Device pixel ratio. | | windowWidth | Number | Window width. | | windowHeight | Number | Window height. | | language | String | The language set by the user in the app. If the app does not support the language setting, return the system language. | | version | String | App version number. | | storage | String | Device disk

capacity. | | currentBattery | String | Current battery percentage. | | system | String | System version. | | platform | String | System name: `Android`, `iOS`. | | titleBarHeight | Number | Title bar height. | | statusBarHeight | Number | Status bar height. | | screenWidth | Number | Screen width. | | screenHeight | Number | Screen height. | | brand | String | Cellphone brand. | | fontSizeSetting | Number | User setting font size. | | app | String | Current running client. The app value can refer to the following table. |

App Value Reference Table

| | | | --- | --- | | **App** | **Value** | | GCash | gcash. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_system_getsysteminfo

---

my.getUpdateManager {#mygetupdatemanager}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.getUpdateManager

2022-07-03 18:44

Call this API to create an UpdateManager object. The UpdateManager is a globally unique manager of the version update, which is used to manage the mini program updates.

Return value

The return value is UpdateManager.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_update_getupdatemanager

---

my.hideBackHome {#myhidebackhome}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.hideBackHome

2021-05-09 18:43

Use this API to hide the home button in the top navigation bar, and the return-home option in the tab bar in the upper right corner.

Notes:

- By default, the home button is displayed if the page where an user enters on starting the Mini Program is not the homepage.
- If the tab bar is configured to redirect to pages/index/index in the app.json, the return-home option is not displayed.

Sample Code

copy

```
//.js
Page({
  onReady() {
    if (my.canIUse('hideBackHome')) {
      my.hideBackHome();
    }
  },
});
```

copy

```
//.js
onLoad(){
    my.reLaunch({
    url:'../swiper/swiper'// An added page other than the homepage
  })

  setTimeout(() => {
    //Hide the home button after 5 seconds
    my.hideBackHome()
  }, 5000)
}
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_navigationbar_hidebackhome

---

my.hideBackHome {#myhidebackhome}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.hideBackHome

2022-07-03 18:44

Use this API to hide the home button in the top navigation bar, and the return-home option in the tab bar in the upper right corner.

Notes:

- By default, the home button is displayed if the page where an user enters on starting the Mini Program is not the homepage.
- If the tab bar is configured to redirect to pages/index/index in the app.json, the return-home option is not displayed.

Sample Code

copy

```
//.js
Page({
  onReady() {
    if (my.canIUse('hideBackHome')) {
      my.hideBackHome();
    }
  },
});
```

copy

```
//.js
onLoad(){
    my.reLaunch({
    url:'../swiper/swiper'// An added page other than the homepage
  })

  setTimeout(() => {
    //Hide the home button after 5 seconds
    my.hideBackHome()
  }, 5000)
}
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_UI_NavigationBar_hideBackHome

---

my.hideKeyboard {#myhidekeyboard}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.hideKeyboard

2021-05-09 18:43

Hide the keyboard.

Sample Code

copy

```
my.hideKeyboard();
```

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_keyboard_hidekeyboard

---

my.hideKeyboard {#myhidekeyboard}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.hideKeyboard

2022-07-03 18:44

Hide the keyboard.

Sample Code

copy

```
my.hideKeyboard();
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_UI_Keyboard_hideKeyboard

---

my.hideLoading {#myhideloading}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.hideLoading

2021-05-09 18:43

Hide the loading dialog.

Sample Code

copy

```
my.hideLoading();

Page({
  onLoad() {
    my.showLoading();
    const that = this;
    setTimeout(() => {
      my.hideLoading({
        page: that,  // Prevents switching to other pages when
execution, page pointing is not accurate
      });
    }, 4000);
  }
})
```

Parameters

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | page | Object | No | Specifically it means the current page instance. In some scenarios, it is required to specify the exact page for hideLoading. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_feedback_hideloading

---

my.hideLoading {#myhideloading}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.hideLoading

2022-07-03 18:44

Hide the loading dialog.

Sample Code

copy

```
my.hideLoading();

Page({
  onLoad() {
    my.showLoading();
    const that = this;
    setTimeout(() => {
      my.hideLoading({
        page: that,  // Prevents switching to other pages when
execution, page pointing is not accurate
      });
    }, 4000);
  }
})
```

Parameters

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | page | Object | No | Specifically it means the current page instance. In some scenarios, it is required to specify the exact page for hideLoading. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_UI_Feedback_hideLoading

---

my.hideNavigationBarLoading {#myhidenavigationbarloading}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.hideNavigationBarLoading

2021-05-09 18:43

Hide the navigation bar loading.

Sample Code

copy

```
my.hideNavigationBarLoading();
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_navigationbar_hidenavigationbarloading

---

my.hideTabBar {#myhidetabbar}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.hideTabBar

2022-07-03 18:44

Hide tab bar.

Sample Code

copy

```
my.hideTabBar({
    animation: true
})
```

Parameters

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | animation | Boolean | No | Need animation effect or not, none by default. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_UI_TabBar_hideTabBar

---

my.hideTabBar {#myhidetabbar}

Last updated: 2021-05-10

Path: miniprogram_gcash

my.hideTabBar

2021-05-10 03:43

Hide tab bar.

Sample Code

copy

```
my.hideTabBar({
    animation: true
})
```

Parameters

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | animation | Boolean | No | Need animation effect or not, none by default. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_tabbar_hidetabbar

my.hideToast {#myhidetoast}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.hideToast

2022-07-03 18:44

Hide the toast dialog.

Sample Code

copy

```
my.hideToast()
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | The callback function for a successful API call. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function used when the API call is completed. This function is always executed no matter the call succeeds or fails. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_UI_Feedback_hideToast

---

my.hideToast {#myhidetoast}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.hideToast

2021-05-09 18:43

Hide the toast dialog.

Sample Code

copy

```
my.hideToast()
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | success | Function | No | The callback function for a successful API call. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function used when the API call is completed. This function is always executed no matter the call succeeds or fails. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_feedback_hidetoast

---

my.makePhoneCall {#mymakephonecall}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.makePhoneCall

2021-05-09 18:43

Make a phone call.

Note: Mini Program Studio simulator does not support simulation temporarily. Please use the real machine to debug.

Sample Code

copy

```
// API-DEMO page/API/make-phone-call/make-phone-call.json
{
    "defaultTitle": "Make a phone call"
}
```

copy

```
// API-DEMO page/API/make-phone-call/make-phone-call.axml
<view class="page">
  <view class="page-section">
```

```
    <view class="page-section-title">my.makePhoneCall</view>
    <view class="page-section-btns">
      <view onTap="makePhoneCall">Make a phone call</view>
    </view>
  </view>
</view>
```

copy

```
// API-DEMO page/API/make-phone-call/make-phone-call.js
Page({
  makePhoneCall() {
    my.makePhoneCall({ number: '00000' });
  },
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | number | String | Yes | Phone number. |

FAQ

'Is not a function' error after calling my.makePhoneCall?

The Mini Program Studio simulator does not support simulation temporarily. Please use the real machine to debug.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_call_makephonecall

---

my.makePhoneCall {#mymakephonecall}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.makePhoneCall

2022-07-03 18:44

Make a phone call.

Note: Mini Program Studio simulator does not support simulation temporarily. Please use the real machine to debug.

Sample Code

copy

```
// API-DEMO page/API/make-phone-call/make-phone-call.json
{
    "defaultTitle": "Make a phone call"
}
```

copy

```
// API-DEMO page/API/make-phone-call/make-phone-call.axml
<view class="page">
  <view class="page-section">
    <view class="page-section-title">my.makePhoneCall</view>
    <view class="page-section-btns">
      <view onTap="makePhoneCall">Make a phone call</view>
    </view>
  </view>
</view>
```

copy

```
// API-DEMO page/API/make-phone-call/make-phone-call.js
Page({
  makePhoneCall() {
    my.makePhoneCall({ number: '00000' });
  },
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | number | String | Yes | Phone number. |

FAQ

'Is not a function' error after calling my.makePhoneCall?

The Mini Program Studio simulator does not support simulation temporarily. Please use the real machine to debug.

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_Call_makePhoneCall

---

my.multiLevelSelect {#mymultilevelselect}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.multiLevelSelect

2022-07-03 18:44

Cascade selection function, mainly used for selecting several levels of associated data, such as province, city and district.

Sample Code

copy

```
//.json
{
     "defaultTitle": "Cascade selector"
}
```

copy

```
<!-- .axml -->
<view class="page">
  <view class="page-description">Cascade selector API</view>
  <view class="page-section">
    <view class="page-section-title">my.multiLevelSelect</view>
    <view class="page-section-demo">
      <button type="primary" onTap="openMultiLevelSelect">Cascade
selector</button>
    </view>
  </view>
</view>
```

copy

```
//.js
Page({
  openMultiLevelSelect() {
    my.multiLevelSelect({
        title: 'Cascade selector',//Cascade selector title
        list: [\
        {\
            name: "City",//entry name\
            subList: [\
                {\
```

```
                        name: "District A",\
                        subList: [\
                            {\
                                name: "Street A"\
                            },\
                            {\
                                name: "Street B"\
                            }\
                        ]\
                    },\
                    {\
                        name: "District B",\
                        subList: [\
                            {\
                                name: "Street C"\
                            },\
                            {\
                                name: "Street D"\
                            }\
                        ]\
                    }\
                ]// cascade sub-data list\
            }],// Cascade data list
            success:(res)=>{
                my.alert({title:JSON.stringify(res)})
            }
        });
    }
})
```

Parameters

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | title | String | No | Title. | | list | JsonArray | Yes | Selection data list. | | name | String | Yes | Entry name. | | subList | JsonArray | No | Sub-entry list. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | success | Boolean | Selection completed or not, returning false for cancellation. | | result | JsonArray | Selection result, such as [{"name":"City"},{"name":"District A"},{"name":"Street A"}]. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_UI_Multi-Level-Select_multiLevelSelect

my.multiLevelSelect {#mymultilevelselect}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.multiLevelSelect

2021-05-09 18:43

Cascade selection function, mainly used for selecting several levels of associated data, such as province, city and district.

Sample Code

copy

```
//.json
{
     "defaultTitle": "Cascade selector"
}
```

copy

```
<!-- .axml -->
<view class="page">
  <view class="page-description">Cascade selector API</view>
  <view class="page-section">
    <view class="page-section-title">my.multiLevelSelect</view>
    <view class="page-section-demo">
      <button type="primary" onTap="openMultiLevelSelect">Cascade
selector</button>
    </view>
  </view>
</view>
```

copy

```
//.js
Page({
  openMultiLevelSelect() {
    my.multiLevelSelect({
        title: 'Cascade selector',//Cascade selector title
        list: [\
        {\
            name: "City",//entry name\
            subList: [\
```

```
                    {\
                        name: "District A",\
                        subList: [\
                            {\
                                name: "Street A"\
                            },\
                            {\
                                name: "Street B"\
                            }\
                        ]\
                    },\
                    {\
                        name: "District B",\
                        subList: [\
                            {\
                                name: "Street C"\
                            },\
                            {\
                                name: "Street D"\
                            }\
                        ]\
                    }\
                ]// cascade sub-data list\
            }],// Cascade data list
            success:(res)=>{
                my.alert({title:JSON.stringify(res)})
            }
        });
    }
})
```

Parameters

The incoming parameter is of the Object type with the following attributes:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | title | String | No | Title. | | list | JsonArray | Yes | Selection data list. | | name | String | Yes | Entry name. | | subList | JsonArray | No | Sub-entry list. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Success Callback Function

The incoming parameter is of the Object type with the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | success | Boolean | Selection completed or not, returning false for cancellation. | | result | JsonArray | Selection result, such as [{"name":"City"},{"name":"District A"},{"name":"Street A"}]. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_multi-level-select_multilevelselect

---

my.navigateBack {#mynavigateback}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.navigateBack

2022-07-03 18:44

Close the current page and return to the previous one or more pages. It is possible to use `getCurrentPages` to get the current page stack information and decide how many levels to return.

Sample Code

copy

```
// Note: When calling navigateTo API, the page that called the method will be added to the stack.

// This is the page one
my.navigateTo({
  url: 'two?pageId=10000'
})

// This is the page two
my.navigateTo({
  url: 'one?pageId=99999'
})

// navigateBack in page three , will return page one
my.navigateBack({
  delta: 2
})
```

Parameters

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Default** | **Description** | | delta | Number | 1 | Number of pages to return. If delta is greater than the number of open pages, it returns to the home page. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_ui_route_navigateback

---

my.navigateBack {#mynavigateback}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.navigateBack

2021-05-09 18:43

Close the current page and return to the previous one or more pages. It is possible to use `getCurrentPages` to get the current page stack information and decide how many levels to return.

Sample Code

copy

```
// Note: When calling navigateTo API, the page that called the method will be added to the stack.

// This is the page one
my.navigateTo({
  url: 'two?pageId=10000'
})

// This is the page two
my.navigateTo({
  url: 'one?pageId=99999'
})

// navigateBack in page three , will return page one
my.navigateBack({
  delta: 2
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Default** | **Description** | | delta | Number | 1 | Number of pages to return. If delta is greater than the number of open pages, it returns to the home page. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_route_navigateback

---

my.navigateBackMiniProgram {#mynavigatebackminiprogram}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.navigateBackMiniProgram

2022-07-03 18:44

Return to the previous Mini Program. Only used for when another Mini Program jumps back to the foregrounded Mini Program.

Sample Code

copy

```
my.navigateBackMiniProgram({
    extraData:{
    "data1":"test"
    },
    success: (res) => {
    console.log(JSON.stringify(res))
    },
    fail: (res) => {
    console.log(JSON.stringify(res))
    }
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | extraData | Object | No | The extra data that needs to be returned to the target Mini Program, and the target Mini Program can get it in `App.onLaunch()` or `App.onShow()`. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_openapi_navigatebackminiprogram

my.navigateBackMiniProgram {#mynavigatebackminiprogram}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.navigateBackMiniProgram

2021-05-09 18:43

Return to the previous Mini Program. Only used for when another Mini Program jumps back to the foregrounded Mini Program.

Sample Code

copy

```
my.navigateBackMiniProgram({
    extraData:{
    "data1":"test"
    },
    success: (res) => {
    console.log(JSON.stringify(res))
    },
    fail: (res) => {
    console.log(JSON.stringify(res))
    }
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | extraData | Object | No | The extra data that needs to be returned to the target Mini Program, and the target Mini Program can get it in `App.onLaunch()` or `App.onShow()`. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_openapi_navigatebackminiprogram

my.navigateTo {#mynavigateto}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.navigateTo

2021-05-09 18:43

Maintain the current page and jump to the specified page within the application. Use `my.navigateBack` to return to the original page.

Note: The maximum page depth is 10. In other words, the navigateTo can be called 10 times at most.

Sample Code

copy

```
my.navigateTo({
  url: 'new_page?count=100'
})

Page({
  onLoad(query){
    my.alert({
      content: JSON.stringify(query),
    });
  }
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | url | String | Yes | The application for thejumping does not include the destination page path of the tabBar. The path can be followed by parameters. Rules for the parameters: The path and parameter are separated with `?`, the parameter key and the parameter value are connected with `=`, and different parameters must be separated with `&`, such as `path?key1=value1&key2=value2`. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_route_navigateto

---

my.navigateTo {#mynavigateto}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.navigateTo

2022-07-03 18:44

Maintain the current page and jump to the specified page within the application. Use `my.navigateBack` to return to the original page.

Note: The maximum page depth is 10. In other words, the navigateTo can be called 10 times at most.

Sample Code

copy

```
my.navigateTo({
  url: 'new_page?count=100'
})

Page({
  onLoad(query){
    my.alert({
      content: JSON.stringify(query),
    });
  }
})
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | url | String | Yes | The application for thejumping does not include the destination page path of the tabBar. The path can be followed by parameters. Rules for the parameters: The path and parameter are separated with `?`, the parameter key and the parameter value are connected with `=`, and different parameters must be separated with `&`, such as `path?key1=value1&key2=value2`. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_ui_route_navigateto

---

my.navigateToMiniProgram {#mynavigatetominiprogram}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.navigateToMiniProgram

2021-05-09 18:43

Jump to another Mini Program.

Sample Code

copy

```
my.navigateToMiniProgram({
    appId: 'xxxx',
    extraData:{
    "data1":"test"
    },
    success: (res) => {
    console.log(JSON.stringify(res))
    },
    fail: (res) => {
    console.log(JSON.stringify(res))
    }
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | appId | String | Yes | The appId of the target Mini Program to jump to. | | path | String | No | The path of the target Mini Program to jump to, open the homepage if it is empty. | | extraData | Object | No | The extra data that needs to be passed to the target Mini Program, and the target Mini Program can get it in `App.onLaunch()` or `App.onShow()`. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_openapi_navigatetominiprogram

---

my.navigateToMiniProgram {#mynavigatetominiprogram}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.navigateToMiniProgram

2022-07-03 18:44

Jump to another Mini Program.

Sample Code

copy

```
my.navigateToMiniProgram({
    appId: 'xxxx',
    extraData:{
    "data1":"test"
    },
    success: (res) => {
    console.log(JSON.stringify(res))
    },
    fail: (res) => {
    console.log(JSON.stringify(res))
    }
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | appId | String | Yes | The appId of the target Mini Program to jump to. | | path | String | No | The path of the target Mini Program to jump to, open the homepage if it is empty. | | extraData | Object | No | The extra data that needs to be passed to the target Mini Program, and the target Mini Program can get it in `App.onLaunch()` or `App.onShow()`. | | success | Function | No | Callback function upon call success. | | fail | Function | No | Callback function upon call failure. | | complete | Function | No | Callback function upon call completion (to be executed upon either call success or failure). |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_openapi_navigatetominiprogram

---

my.notifyBLECharacteristicValueChange {#mynotifyblecharacteristicvaluechange}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.notifyBLECharacteristicValueChange

2022-07-03 18:44

Use this API enable notification on change of Bluetooth Low Energy (BLE) device characteristics.

Instructions:

- The device characteristics must support `notify` or `indicate` to use this API. See properties in my.getBLEDeviceCharacteristics for details.
- You must enable this API first before you can use my.onBLECharacteristicValueChange.
- After a successful subscription, the device must actively update the value of the characteristic to trigger my.onBLECharacteristicValueChange.
- Subscription is more efficient and is recommended over the read method.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
my.notifyBLECharacteristicValueChange({
  deviceId: deviceId,
  serviceId: serviceId,
  characteristicId: characteristicId,
  success: (res) => {
    console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | deviceId | String | Yes | The Bluetooth device ID. | | serviceId | String | Yes | The UUID of the service corresponding to a Bluetooth characteristic. | | characteristicId | String | Yes | The Bluetooth characteristic UUID. | | descriptorId | String | No | Descriptor UUID of the notification. This is Android-specific, the default value is 00002902-0000-10008000-00805F9b34fb. | | state | Boolean | No | Whether `notify` or `indicate` is enabled. | | success | Function | No | The callback function for a successful API call. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function used when the API call is completed. This function is always executed no matter the call succeeds or fails. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_bluetooth_ble_notifyblecharacteristicvaluechange

---

my.notifyBLECharacteristicValueChange {#mynotifyblecharacteristicvaluechange}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.notifyBLECharacteristicValueChange

2021-05-09 18:43

Use this API enable notification on change of Bluetooth Low Energy (BLE) device characteristics.

Instructions:

- The device characteristics must support `notify` or `indicate` to use this API. See properties in my.getBLEDeviceCharacteristics for details.
- You must enable this API first before you can use my.onBLECharacteristicValueChange.
- After a successful subscription, the device must actively update the value of the characteristic to trigger my.onBLECharacteristicValueChange.
- Subscription is more efficient and is recommended over the read method.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
my.notifyBLECharacteristicValueChange({
  deviceId: deviceId,
  serviceId: serviceId,
  characteristicId: characteristicId,
  success: (res) => {
    console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Parameters

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | deviceId | String | Yes | The Bluetooth device ID. | | serviceId | String | Yes | The UUID of the service corresponding to a Bluetooth characteristic. | | characteristicId | String | Yes | The Bluetooth characteristic UUID. | | descriptorId | String | No | Descriptor UUID of the notification. This is Android-specific, the default value is 00002902-0000-10008000-00805F9b34fb. | | state | Boolean | No | Whether `notify` or `indicate` is enabled. | | success | Function | No | The callback function for a successful API call. | | fail | Function | No | The callback function for a failed API call. | | complete | Function | No | The callback function used when the API call is completed. This function is always executed no matter the call succeeds or fails. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_ble_notifyblecharacteristicvaluechange

---

my.offAccelerometerChange {#myoffaccelerometerchange}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offAccelerometerChange

2021-05-09 18:43

Use this API to stop listening to acceleration data event.

Sample Code

copy

```
my.offAccelerometerChange();
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offAccelerometerChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offAccelerometerChange(this.callback);
```

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_accelerometer_offaccelerometerchange

---

my.offAccelerometerChange {#myoffaccelerometerchange}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offAccelerometerChange

2022-07-03 18:44

Use this API to stop listening to acceleration data event.

Sample Code

copy

```
my.offAccelerometerChange();
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offAccelerometerChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offAccelerometerChange(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_Accelerometer_offAccelerometerChange

---

my.offAppHide {#myoffapphide}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offAppHide

2022-07-03 18:44

Unlisten for the event that the mini program is switched to background from foreground.

Sample code

.axml

copy

```
<!-- .axml-->
<button size="default" onTap="offAppHideHanlder"
type="primary">Unlisten for the event that the mini program is
switched to background</button>
```

.js

copy

```
//.js
onLoad() {
    my.onAppHide(this.onAppHideHandler)
},
// The method of listening for the event that the mini program is
switched to background from foreground
onAppHideHandler() {
    console.log('The method of listening for the event that the mini
program is switched to background from foreground')
},
// The method of unlistening for the event that the mini program is
switched to background from foreground
offAppHideHanlder() {
    my.offAppHide(this.onAppHideHandler)
},
```

Parameters

The parameter is in object type and has the following property:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | callback | Function | The callback function for the event that the mini program is switched to background. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_event_offapphide

---

my.offAppShow {#myoffappshow}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offAppShow

2022-07-03 18:44

Unlisten for the event that the mini program is switched to foreground from background.

Sample code

.axml

copy

```
<!-- .axml-->
<button size="default" onTap="offAppShowHanlder"
type="primary">Unlisten for the event that the mini program is
switched to foreground from background</button>
```

.js

copy

```
//.js
onLoad() {
    my.onAppShow(this.onAppShowHandler)
},
//The method of listening for the event that the mini program is
switched to foreground from background
onAppShowHandler() {
    console.log('The mini program is switched to foreground from
background')
},
//The method of unlistening for the event that the mini program is
switched to foreground from background
offAppShowHanlder() {
    my.offAppShow(this.onAppShowHandler)
},
apphide() {
    console.log('The mini program is switched to background from
foreground')
}
```

Parameters

The parameter is in object type and has the following property:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | callback | Function | The callback function for the event that the mini program is switched to foreground. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_event_offappshow

---

my.offBLECharacteristicValueChange {#myoffblecharacteristicvaluechange}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offBLECharacteristicValueChange

2021-05-09 18:43

Use this API to unlisten to the BLE device characteristic change event.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onBLECharacteristicValueChange(this.callback);
  },
  onUnload() {
    my.offBLECharacteristicValueChange(this.callback);
  },
  callback(res) {
    console.log(res);
  },
})
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | deviceId | String | The Bluetooth device ID. | | serviceId | String | The UUID of the service corresponding to a Bluetooth characteristic. | | characteristicId | String | The Bluetooth device characteristic UUID. | | value | Hex String | The latest hexadecimal value of the characteristic. |

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offBLECharacteristicValueChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offBLECharacteristicValueChange(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_ble_offblecharacteristicvaluechange

---

my.offBLECharacteristicValueChange {#myoffblecharacteristicvaluechange}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offBLECharacteristicValueChange

2022-07-03 18:44

Use this API to unlisten to the BLE device characteristic change event.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onBLECharacteristicValueChange(this.callback);
  },
  onUnload() {
    my.offBLECharacteristicValueChange(this.callback);
  },
  callback(res) {
    console.log(res);
  },
})
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | deviceId | String | The Bluetooth device ID. | | serviceId | String | The UUID of the service corresponding to a Bluetooth characteristic. | | characteristicId | String | The Bluetooth device characteristic UUID. | | value | Hex String | The latest hexadecimal value of the characteristic. |

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offBLECharacteristicValueChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offBLECharacteristicValueChange(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_Bluetooth_BLE_offBLECharacteristicValueChange

---

my.offBLEConnectionStateChanged {#myoffbleconnectionstatechanged}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offBLEConnectionStateChanged

2022-07-03 18:44

Use this API to unlisten to the Bluetooth Low Energy (BLE) connection status change event.

Instruction:

It is recommended that you call the `off` method and close event listening before you call the `on` method to listen events to prevent the situation where multiple listening event cause multiple callbacks of an event.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
       <button type="primary"
```

```
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
       <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
       <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
       <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>
    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
```

```
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {
            // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];
              //Pair the target device with the device name or
```

```
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the connected device
  getConnectedBluetoothDevices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connecting devices!' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
        devid = res.devices[0].deviceId;
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
```

```
  });
},

//Obtain all searched devices
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Disconnect the device
disconnectBLEDevice() {
  my.disconnectBLEDevice({
    deviceId: this.data.devid,
    success: () => {
      my.alert({ content: 'Succeeded to disconnect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain the services of the connected device
getBLEDeviceServices() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
```

```
            my.alert({ content: 'No connected devices' });
            return;
          }
          my.getBLEDeviceServices({
            deviceId: this.data.devid,
            success: res => {
              my.alert({ content: JSON.stringify(res) });
              this.setData({
                serid: res.services[0].serviceId,
              });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },

    //Obtain the char ID of the connected device, read and write
  characteristics are respectively screened out.
    getBLEDeviceCharacteristics() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.getBLEDeviceCharacteristics({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            success: res => {
              my.alert({ content: JSON.stringify(res) });
              //See the related document for more information of the
  properties of the characteristics. Pair the characteristics according
  to the properties and record the value for later use.
              this.setData({
                charid: res.characteristics[0].characteristicId,
              });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },
```

```
  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.readBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          //1  Android reading service
          // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
          success: res => {
            my.alert({ content: JSON.stringify(res) });
          },
          fail: error => {
            my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  writeBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.writeBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.charid,
          //Android writing service
          //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
          value: 'ABCD',
          success: res => {
            my.alert({ content: 'Succeeded to write data!' });
          },
          fail: error => {
```

```
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },
  notifyBLECharacteristicValueChange() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.notifyBLECharacteristicValueChange({
          state: true,
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          success: () => {
            //Listens to characteristic change events
            my.onBLECharacteristicValueChange({
              success: res => {
                //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                my.alert({ content: 'Obtain the response data = ' +
res.value });
              },
            });
            my.alert({ content: 'Succeeded to listen' });
          },
          fail: error => {
            my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
```

```
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChang
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offBLEConnectionStateChanged();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offBLEConnectionStateChanged(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_bluetooth_ble_offbleconnectionstatechanged

---

my.offBLEConnectionStateChanged {#myoffbleconnectionstatechanged}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offBLEConnectionStateChanged

2021-05-09 18:43

Use this API to unlisten to the Bluetooth Low Energy (BLE) connection status change event.

Instruction:

It is recommended that you call the `off` method and close event listening before you call the `on` method to listen events to prevent the situation where multiple listening event cause multiple callbacks of an event.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
```

```
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
       <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
```

```
      <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
      <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
      <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
      <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
      <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
      <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
      <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
      <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>
    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
```

```
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {
            // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];
              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
```

```
                  break;
                }
              }
            },
            fail: error => {
              my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
            },
          });
        },
        fail: error => {
          my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
        },
      });
    },

    //Stop scanning
    stopBluetoothDevicesDiscovery() {
      my.stopBluetoothDevicesDiscovery({
        success: res => {
          my.offBluetoothDeviceFound();
          my.alert({ content: 'Succeeded!' });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },

    //Obtain the connected device
    getConnectedBluetoothDevices() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connecting devices!' });
            return;
          }
          my.alert({ content: JSON.stringify(res) });
          devid = res.devices[0].deviceId;
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },

    //Obtain all searched devices
    getBluetoothDevices() {
      my.getBluetoothDevices({
        success: res => {
```

```
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Disconnect the device
disconnectBLEDevice() {
  my.disconnectBLEDevice({
    deviceId: this.data.devid,
    success: () => {
      my.alert({ content: 'Succeeded to disconnect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain the services of the connected device
getBLEDeviceServices() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      my.getBLEDeviceServices({
        deviceId: this.data.devid,
        success: res => {
          my.alert({ content: JSON.stringify(res) });
```

```
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Obtain the char ID of the connected device, read and write
characteristics are respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            //See the related document for more information of the
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
```

```
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.readBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.notifyId,
            //1  Android reading service
            // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
            success: res => {
              my.alert({ content: JSON.stringify(res) });
            },
            fail: error => {
              my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
            },
          });
        },
      });
    },
    writeBLECharacteristicValue() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.writeBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.charid,
            //Android writing service
            //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
            value: 'ABCD',
            success: res => {
              my.alert({ content: 'Succeeded to write data!' });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },
    notifyBLECharacteristicValueChange() {
```

```
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.notifyBLECharacteristicValueChange({
          state: true,
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          success: () => {
            //Listens to characteristic change events
            my.onBLECharacteristicValueChange({
              success: res => {
                //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                my.alert({ content: 'Obtain the response data = ' +
res.value });
              },
            });
            my.alert({ content: 'Succeeded to listen' });
          },
          fail: error => {
            my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState(
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
```

```
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChang
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offBLEConnectionStateChanged();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offBLEConnectionStateChanged(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_ble_offbleconnectionstatechanged

---

my.offBluetoothAdapterStateChange {#myoffbluetoothadapterstatechange}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offBluetoothAdapterStateChange

2021-05-09 18:43

Use this API to remove the bluetooth adapter with a state change.

In order to prevent multiple callbacks of an event, which are resulted from multiple registered event listeners, it is recommended to call off method to listen for an event and close the previous event listener, before you call on method.

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
my.offBluetoothAdapterStateChange();
```

Transmitting Callback Values

- If you don't transmit the callback value, all the event listener callbacks will be removed. See the below code sample for more information:

copy

```
my.offBluetoothAdapterStateChange();
```

- If you transmit the callback value, the corresponding callbacks will be removed. See the below code sample for more information:

copy

```
my.offBluetoothAdapterStateChange(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_bluetooth_offbluetoothadapterstatechange

---

my.offBluetoothAdapterStateChange {#myoffbluetoothadapterstatechange}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offBluetoothAdapterStateChange

2022-07-03 18:44

Use this API to remove the bluetooth adapter with a state change.

In order to prevent multiple callbacks of an event, which are resulted from multiple registered event listeners, it is recommended to call off method to listen for an event and close the previous event listener, before you call on method.

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
my.offBluetoothAdapterStateChange();
```

Transmitting Callback Values

- If you don't transmit the callback value, all the event listener callbacks will be removed. See the below code sample for more information:

copy

```
my.offBluetoothAdapterStateChange();
```

- If you transmit the callback value, the corresponding callbacks will be removed. See the below code sample for more information:

copy

```
my.offBluetoothAdapterStateChange(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_Bluetooth_Bluetooth_offBluetoothAdapterStateChange

---

my.offBluetoothDeviceFound {#myoffbluetoothdevicefound}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offBluetoothDeviceFound

2022-07-03 18:44

Use this API to remove the bluetooth devices that are found.

In order to prevent multiple callbacks of an event, which are resulted from multiple registered event listeners, it is recommended to call off method to listen for an event and close the previous event listener, before you call on method.

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
my.offBluetoothDeviceFound();
```

Transmitting Callback Values

- If you don't transmit the callback value, all the event listener callbacks will be removed. See the below code sample for more information:

copy

```
my.offBluetoothDeviceFoun();
```

- If you transmit the callback value, the corresponding callbacks will be removed. See the below code sample for more information:

copy

```
my.offBluetoothDeviceFoun(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_bluetooth_bluetooth_offbluetoothdevicefound

---

my.offBluetoothDeviceFound {#myoffbluetoothdevicefound}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offBluetoothDeviceFound

2021-05-09 18:43

Use this API to remove the bluetooth devices that are found.

In order to prevent multiple callbacks of an event, which are resulted from multiple registered event listeners, it is recommended to call off method to listen for an event and close the previous event listener, before you call on method.

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
my.offBluetoothDeviceFound();
```

Transmitting Callback Values

- If you don't transmit the callback value, all the event listener callbacks will be removed. See the below code sample for more information:

copy

```
my.offBluetoothDeviceFoun();
```

- If you transmit the callback value, the corresponding callbacks will be removed. See the below code sample for more information:

copy

```
my.offBluetoothDeviceFoun(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_bluetooth_offbluetoothdevicefound

---

my.offCompassChange {#myoffcompasschange}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offCompassChange

2021-05-09 18:43

Use this API to unlisten to the compass data.

Sample Code

copy

```
my.offCompassChange();
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offCompassChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offCompassChange(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_compass_offcompasschange

---

my.offCompassChange {#myoffcompasschange}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offCompassChange

2022-07-03 18:44

Use this API to unlisten to the compass data.

Sample Code

copy

```
my.offCompassChange();
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offCompassChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offCompassChange(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_compass_offcompasschange

---

my.offError {#myofferror}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offError

2022-07-03 18:44

Unlisten for the event that JS errors occur in the mini program.

Sample code

copy

```
// .js
App({
  onShow() {
    this.handleError = error => {
      // Errors occur when running the mini program.
      console.log(error);
    }
    // The type of error is String.
    my.onError(this.handleError);
  },
  onHide() {
    // Unlisten for the event that JS errors occur in the mini
program.
    my.offError(this.handleError);
  }
})
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | callback | Function | The callback function for the event that JS errors occur in the mini program. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_event_offerror

---

my.offMemoryWarning {#myoffmemorywarning}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offMemoryWarning

2022-07-03 18:44

Use this API to unlisten to the insufficient memory alarm event. Ensure that the parameter (callback) is the same object as the one in onMemoryWarning.

Sample Code

copy

```
// API-DEMO page/API/memory-warning/memory-warning.json
{
   "defaultTitle": "OnMemoryWarning"
}
```

copy

```
<!-- API-DEMO page/API/memory-warning/memory-warning.axml-->
<view class="page">

  <button type="primary" onTap="onMemoryWarning">
    Listen to Insufficient Memory Alarm Event
  </button>

</view>
```

copy

```
// API-DEMO page/API/memory-warning/memory-warning.js
Page({
  onLoad() {
    this.callback = (res) => {
        var levelString = 'iOS device, No alarm level exists.';
        switch (res.level) {
          case 10:
            levelString = 'Android device, level =
TRIM_MEMORY_RUNNING_LOW';
            break;
          case 15:
            levelString = 'Android device, level =
TRIM_MEMORY_RUNNING_CRITICAL';
            break;
        }
        my.alert({
          title: 'Received insufficient memory alarm',
          content: levelString
        });
    };
    this.isApiAvailable = my.canIUse('onMemoryWarning');
  },
  onMemoryWarning() {
    if (this.isApiAvailable) {
      my.onMemoryWarning(this.callback);
    } else {
```

```
      my.alert({
        title: 'Client version is too low',
        content: 'my.onMemoryWarning() and my.offMemoryWarning() need
10.1.35 or higher versions'
      });
    }
  },
  onUnload() {
    if (this.isApiAvailable) {
      my.offMemoryWarning(this.callback);
    }
  }
});
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offMemoryWarning();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offMemoryWarning(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_memory-warning_offmemorywarning

---

my.offMemoryWarning {#myoffmemorywarning}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offMemoryWarning

2021-05-09 18:43

Use this API to unlisten to the insufficient memory alarm event. Ensure that the parameter (callback) is the same object as the one in onMemoryWarning.

Sample Code

copy

```
// API-DEMO page/API/memory-warning/memory-warning.json
{
   "defaultTitle": "OnMemoryWarning"
}
```

copy

```
<!-- API-DEMO page/API/memory-warning/memory-warning.axml-->
<view class="page">

  <button type="primary" onTap="onMemoryWarning">
    Listen to Insufficient Memory Alarm Event
  </button>

</view>
```

copy

```
// API-DEMO page/API/memory-warning/memory-warning.js
Page({
  onLoad() {
    this.callback = (res) => {
        var levelString = 'iOS device, No alarm level exists.';
        switch (res.level) {
          case 10:
            levelString = 'Android device, level =
TRIM_MEMORY_RUNNING_LOW';
            break;
          case 15:
            levelString = 'Android device, level =
TRIM_MEMORY_RUNNING_CRITICAL';
            break;
        }
        my.alert({
          title: 'Received insufficient memory alarm',
          content: levelString
        });
    };
    this.isApiAvailable = my.canIUse('onMemoryWarning');
  },
  onMemoryWarning() {
    if (this.isApiAvailable) {
      my.onMemoryWarning(this.callback);
    } else {
      my.alert({
        title: 'Client version is too low',
        content: 'my.onMemoryWarning() and my.offMemoryWarning() need
10.1.35 or higher versions'
```

```
      });
    }
  },
  onUnload() {
    if (this.isApiAvailable) {
      my.offMemoryWarning(this.callback);
    }
  }
});
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offMemoryWarning();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offMemoryWarning(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_memory-warning_offmemorywarning

---

my.offSocketClose {#myoffsocketclose}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offSocketClose

2022-07-03 18:44

Use this API to unlisten to the event of disabling the WebSocket connection.

Sample Code

copy

```
Page({
  onLoad() {
  my.onSocketClose(this.callback);
  },
  onUnload() {
    my.offSocketClose(this.callback);
    //    my.offSocketClose();
  },
  callback(res) {
  my.alert({content: 'The connection is disabled!'});
      this.setData({
        sendMessageAbility: false,
        closeLinkAbility: false,
      });
  },
})
```

Note: The case is only for reference. Please use your own URL to test.

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offSocketClose();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offSocketClose(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Network_offSocketClose

---

my.offSocketClose {#myoffsocketclose}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offSocketClose

2021-05-09 18:43

Use this API to unlisten to the event of disabling the WebSocket connection.

Sample Code

copy

```
Page({
  onLoad() {
  my.onSocketClose(this.callback);
  },
  onUnload() {
    my.offSocketClose(this.callback);
    //    my.offSocketClose();
  },
  callback(res) {
  my.alert({content: 'The connection is disabled!'});
      this.setData({
        sendMessageAbility: false,
        closeLinkAbility: false,
      });
  },
})
```

Note: The case is only for reference. Please use your own URL to test.

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offSocketClose();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offSocketClose(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_network_offsocketclose

---

my.offSocketError {#myoffsocketerror}

Last updated: 2022-07-04

Path: miniprogram_gcash

my.offSocketError

2022-07-04 03:44

Use this API to unlisten to WebSocket error events.

Sample Code

copy

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onSocketError(this.callback);
  },
  onUnload() {
    my.offSocketError(this.callback);
  },
  callback(res) {
  },
})
```

Note: The case is only for reference. Please use your own URL to test.

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offSocketError();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offSocketError(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Network_offSocketError

---

my.offSocketError {#myoffsocketerror}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offSocketError

2021-05-09 18:43

Use this API to unlisten to WebSocket error events.

Sample Code

copy

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onSocketError(this.callback);
  },
  onUnload() {
    my.offSocketError(this.callback);
  },
  callback(res) {
  },
})
```

Note: The case is only for reference. Please use your own URL to test.

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offSocketError();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offSocketError(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_network_offsocketerror

---

my.offSocketMessage {#myoffsocketmessage}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offSocketMessage

2022-07-03 18:44

Use this API to unlisten to the event of receiving server messages by WebSocket.

Sample Code

copy

```
my.connectSocket({
  url: 'Server URL'
})

my.onSocketMessage(function(res) {
  console.log('Server content received  ' + res.data)
})

my.offSocketMessage();
```

Note: The case is only for reference. Please use your own URL to test.

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offSocketMessage();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offSocketMessage(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Network_offSocketMessage

---

my.offSocketMessage {#myoffsocketmessage}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offSocketMessage

2021-05-09 18:43

Use this API to unlisten to the event of receiving server messages by WebSocket.

Sample Code

copy

```
my.connectSocket({
  url: 'Server URL'
})

my.onSocketMessage(function(res) {
  console.log('Server content received  ' + res.data)
})

my.offSocketMessage();
```

Note: The case is only for reference. Please use your own URL to test.

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offSocketMessage();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offSocketMessage(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_network_offsocketmessage

---

my.offSocketOpen {#myoffsocketopen}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offSocketOpen

2021-05-09 18:43

Use this API to unlisten to the event of enabling the WebSocket connection.

Sample Code

copy

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onSocketOpen(this.callback);
  },
  onUnload() {
    my.offSocketOpen(this.callback);
  },
  callback(res) {
  },
})
```

Note: The case is only for reference. Please use your own URL to test.

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offSocketOpen();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offSocketOpen(this.callback);
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_network_offsocketopen

---

my.offSocketOpen {#myoffsocketopen}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offSocketOpen

2022-07-03 18:44

Use this API to unlisten to the event of enabling the WebSocket connection.

Sample Code

copy

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onSocketOpen(this.callback);
  },
  onUnload() {
    my.offSocketOpen(this.callback);
  },
  callback(res) {
  },
})
```

Note: The case is only for reference. Please use your own URL to test.

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

copy

```
my.offSocketOpen();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

copy

```
my.offSocketOpen(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Network_offSocketOpen

---

my.offUnhandledRejection {#myoffunhandledrejection}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offUnhandledRejection

2022-07-03 18:44

Unlisten for the *unhandledrejection* event.

Sample code

copy

```
//.js
App({
  onShow(options) {
    const handleRejection = (res) => {
      console.log(res.reason);
      console.log(res.promise);
    }
    my.onUnhandledRejection(handleRejection);
    my.offUnhandledRejection(handleRejection);
  }
})
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | callback | Function | The *unhandledrejection* event is triggered when a JavaScript `Promise` that has no rejection handler is rejected. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_event_offunhandledrejection

---

my.offUserCaptureScreen {#myoffusercapturescreen}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.offUserCaptureScreen

2022-07-03 18:44

Cancel screen capture listener event. This is usually paired with `my.onUserCaptureScreen`.

Sample Code

copy

```
<!-- API-DEMO page/API/user-capture-screen/user-capture-screen.axml-->
<view class="page">
  <view class="page-description">User screen capture event API</view>
  <view class="page-section">
    <view class="page-section-title">my.onUserCaptureScreen</view>
    <view class="page-section-demo">
      <view>Current status: {{ condition ? "listening on" : 'Listening
off' }}</view>
      <view a:if="{{condition}}">
        <button type="primary" onTap="offUserCaptureScreen">Cancel
screen capture listening event</button>
      </view>
      <view a:else>
        <button type="primary" onTap="onUserCaptureScreen">Turn on
screen capture listening event</button>
      </view>
    </view>
  </view>
</view>
```

copy

```
// API-DEMO page/API/user-capture-screen/user-capture-screen.js
Page({
  data: {
    condition: false,
  },
  onReady() {
    my.onUserCaptureScreen(() => {
      my.alert({
        content: 'Received user screen capture',
      });
    });
  },
  offUserCaptureScreen() {
    my.offUserCaptureScreen();
    this.setData({
      condition: false,
    });
  },
  onUserCaptureScreen() {
    my.onUserCaptureScreen(() => {
      my.alert({
        content: 'Received user screen capture'
      });
    });
    this.setData({
      condition: true,
    });
  },
});
```

Dismissing Callback

• If you need to remove all event listener callback. Sample code:

copy

```
my.offUserCaptureScreen();
```

• If you need to remove a specific callback event. Sample code:

copy

```
my.offUserCaptureScreen(this.callback);
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_capture_offusercapturescreen

---

my.offUserCaptureScreen {#myoffusercapturescreen}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.offUserCaptureScreen

2021-05-09 18:43

Cancel screen capture listener event. This is usually paired with `my.onUserCaptureScreen`.

Sample Code

copy

```
<!-- API-DEMO page/API/user-capture-screen/user-capture-screen.axml-->
<view class="page">
  <view class="page-description">User screen capture event API</view>
  <view class="page-section">
    <view class="page-section-title">my.onUserCaptureScreen</view>
    <view class="page-section-demo">
      <view>Current status: {{ condition ? "listening on" : 'Listening
off' }}</view>
      <view a:if="{{condition}}">
        <button type="primary" onTap="offUserCaptureScreen">Cancel
screen capture listening event</button>
      </view>
      <view a:else>
        <button type="primary" onTap="onUserCaptureScreen">Turn on
screen capture listening event</button>
      </view>
    </view>
  </view>
</view>
```

copy

```
// API-DEMO page/API/user-capture-screen/user-capture-screen.js
Page({
  data: {
    condition: false,
  },
  onReady() {
    my.onUserCaptureScreen(() => {
      my.alert({
        content: 'Received user screen capture',
```

```
      });
    });
  },
  offUserCaptureScreen() {
    my.offUserCaptureScreen();
    this.setData({
      condition: false,
    });
  },
  onUserCaptureScreen() {
    my.onUserCaptureScreen(() => {
      my.alert({
        content: 'Received user screen capture'
      });
    });
    this.setData({
      condition: true,
    });
  },
});
```

Dismissing Callback

• If you need to remove all event listener callback. Sample code:

copy

```
my.offUserCaptureScreen();
```

• If you need to remove a specific callback event. Sample code:

copy

```
my.offUserCaptureScreen(this.callback);
```

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_capture_offusercapturescreen

---

my.onAccelerometerChange {#myonaccelerometerchange}

Last updated: 2022-07-04

Path: miniprogram_gcash

my.onAccelerometerChange

2022-07-04 03:44

Use this API to listen to the acceleration data event. The callback interval is 500ms. After the interface is called, the listening is automatically started. You can use my.offAccelerometerChange to stop listening.**

Sample Code

copy

```
my.onAccelerometerChange(function(res) {
  console.log(res.x);
  console.log(res.y);
  console.log(res.z);
})
```

Parameters

The property is a callback function which uses object properties with the following property:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | x | Number | x-axis | | y | Number | y-axis | | z | Number | z-axis |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_accelerometer_onaccelerometerchange

---

my.onAccelerometerChange {#myonaccelerometerchange}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.onAccelerometerChange

2021-05-09 18:43

Use this API to listen to the acceleration data event. The callback interval is 500ms. After the interface is called, the listening is automatically started. You can use my.offAccelerometerChange to stop listening.**

Sample Code

copy

```
my.onAccelerometerChange(function(res) {
  console.log(res.x);
  console.log(res.y);
  console.log(res.z);
})
```

Parameters

The property is a callback function which uses object properties with the following property:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | x | Number | x-axis | | y | Number | y-axis | | z | Number | z-axis |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_accelerometer_onaccelerometerchange

---

my.onAppHide {#myonapphide}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.onAppHide

2022-07-03 18:44

Listen for the event that the mini program is switched to background from foreground. The triggered time of the API is the same with that of the `onHide()` method. To unlisten for the event that the mini program is switched to background from foreground, see my.offAppHide.

Sample code

.axml

copy

```
<!-- .axml-->
<button size="default" onTap="offAppHideHanlder"
```

```
type="primary">Unlisten for the event tha the mini program is switched
to background</button>
```

.js

copy

```
//.js
onLoad() {
    my.onAppHide(this.onAppHideHandler)
},
// The method of listening for the event that the mini program is
switched to background
onAppHideHandler() {
    console.log('The method of listening for the event that the mini
program is switched to background')
},
// The method of unlistening for the event that the mini program is
switched to background
offAppHideHanlder() {
    my.offAppHide(this.onAppHideHandler)
},
```

Parameters

The parameter is in object type and has the following property:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | callback | Function | The callback function for the event that the mini program is switched to background. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_event_onapphide

---

my.onBLECharacteristicValueChange {#myonblecharacteristicvaluechange}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.onBLECharacteristicValueChange

2022-07-03 18:44

Use this API to listen to the Bluetooth Low Energy (BLE) device characteristic change event.

Instruction:

It is recommended that you call the `off` method and close event listening before you call the `on` method to listen events to prevent the situation where multiple listening event cause multiple callbacks of an event.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
```

```
      <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
      <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
      <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
      <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
      <button type="primary" onTap="getBLEDeviceServices">Get device
services</button>
      <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
      <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
      <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
      <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
      <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
      <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
      <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
      <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
      <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
      <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>

    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
```

```
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {

            // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];

              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
```

```
  });
},

//Obtain the connected device
getConnectedBluetoothDevices() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connecting devices!' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain all searched devices
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
```

```
  //Disconnect the device
  disconnectBLEDevice() {
    my.disconnectBLEDevice({
      deviceId: this.data.devid,
      success: () => {
        my.alert({ content: 'Succeeded to disconnect!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the services of the connected device
  getBLEDeviceServices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Obtain the char ID of the connected device, read and write
characteristics are respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
```

```
            serviceId: this.data.serid,
            success: res => {
              my.alert({ content: JSON.stringify(res) });

              //See the related document for more information of the
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
              this.setData({
              this.setData({
              this.setData({
                charid: res.characteristics[0].characteristicId,
              });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
  },

  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.readBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,

          //1  Android reading service
          // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
          success: res => {
            my.alert({ content: JSON.stringify(res) });
          },
          fail: error => {
            my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
```

```
writeBLECharacteristicValue() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });

      my.writeBLECharacteristicValue({
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.charid,

        //Android writing service
        //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
        //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
        value: 'ABCD',
        success: res => {
          my.alert({ content: 'Succeeded to write data!' });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },
  });
},
notifyBLECharacteristicValueChange() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });

      my.notifyBLECharacteristicValueChange({
        state: true,
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.notifyId,
        success: () => {

          //Listens on characteristic change events
          my.onBLECharacteristicValueChange({
```

```
                success: res => {

                  //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                  my.alert({ content: 'Obtain the response data = ' +
res.value });
                },
              });
              my.alert({ content: 'Succeeded to listen' });
            },
            fail: error => {
              my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
            },
          });
        },
      });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
```

```
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChan
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | deviceId | String | The Bluetooth device ID. | | connected | Boolean | The current state of the connection. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_bluetooth_ble_onblecharacteristicvaluechange

---

my.onBLECharacteristicValueChange {#myonblecharacteristicvaluechange}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.onBLECharacteristicValueChange

2021-05-09 18:43

Use this API to listen to the Bluetooth Low Energy (BLE) device characteristic change event.

Instruction:

It is recommended that you call the `off` method and close event listening before you call the `on` method to listen events to prevent the situation where multiple listening event cause multiple callbacks of an event.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
```

```
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Get device
services</button>
       <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
       <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
       <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
       <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>

    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
```

```
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
```

```
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {

            // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];

              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the connected device
  getConnectedBluetoothDevices() {
```

```
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connecting devices!' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain all searched devices
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Disconnect the device
disconnectBLEDevice() {
  my.disconnectBLEDevice({
    deviceId: this.data.devid,
    success: () => {
```

```
          my.alert({ content: 'Succeeded to disconnect!' });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },

    //Obtain the services of the connected device
    getBLEDeviceServices() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          my.getBLEDeviceServices({
            deviceId: this.data.devid,
            success: res => {
              my.alert({ content: JSON.stringify(res) });
              this.setData({
                serid: res.services[0].serviceId,
              });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },

    //Obtain the char ID of the connected device, read and write
  characteristics are respectively screened out.
    getBLEDeviceCharacteristics() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.getBLEDeviceCharacteristics({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            success: res => {
              my.alert({ content: JSON.stringify(res) });

              //See the related document for more information of the
```

```
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
            this.setData({
            this.setData({
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.readBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,

          //1  Android reading service
          // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
          success: res => {
            my.alert({ content: JSON.stringify(res) });
          },
          fail: error => {
            my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
          },
        });
      },
    });
  },

  writeBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
```

```
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });

          my.writeBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.charid,

            //Android writing service
            //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
            value: 'ABCD',
            success: res => {
              my.alert({ content: 'Succeeded to write data!' });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },
    notifyBLECharacteristicValueChange() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });

          my.notifyBLECharacteristicValueChange({
            state: true,
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.notifyId,
            success: () => {

              //Listens on characteristic change events
              my.onBLECharacteristicValueChange({
                success: res => {

                  //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                  my.alert({ content: 'Obtain the response data = ' +
```

```
res.value });
              },
            });
            my.alert({ content: 'Succeeded to listen' });
          },
          fail: error => {
            my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
```

```
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChan
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | deviceId | String | The Bluetooth device ID. | | connected | Boolean | The current state of the connection. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_ble_onblecharacteristicvaluechange

---

my.onBLEConnectionStateChanged {#myonbleconnectionstatechanged}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.onBLEConnectionStateChanged

2022-07-03 18:44

Use this API to listen to the Bluetooth Low Energy (BLE) connection error event, including device loss and unusual disconnections.

Instruction:

It is recommended that you call the `off` method and close event listening before you call the `on` method to listen events to prevent the situation where multiple listening event cause multiple callbacks of an event.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}

.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>

    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>

    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
```

```
      <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
      <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
      <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
      <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
      <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>

     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
      <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
      <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
      <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
      <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>

     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
      <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
      <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
      <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
      <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>
    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
```

```
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
```

```
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {
            // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];
              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the connected device
  getConnectedBluetoothDevices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
```

```
          my.alert({ content: 'No connecting devices!' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
        devid = res.devices[0].deviceId;
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain all searched devices
  getBluetoothDevices() {
    my.getBluetoothDevices({
      success: res => {
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  bindKeyInput(e) {
    this.setData({
      devid: e.detail.value,
    });
  },

  //Connect the device
  connectBLEDevice() {
    my.connectBLEDevice({
      deviceId: this.data.devid,
      success: res => {
        my.alert({ content: 'Succeeded to connect!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Disconnect the device
  disconnectBLEDevice() {
    my.disconnectBLEDevice({
      deviceId: this.data.devid,
      success: () => {
        my.alert({ content: 'Succeeded to disconnect!' });
      },
      fail: error => {
```

```
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },

    //Obtain the services of the connected device
    getBLEDeviceServices() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          my.getBLEDeviceServices({
            deviceId: this.data.devid,
            success: res => {
              my.alert({ content: JSON.stringify(res) });
              this.setData({
                serid: res.services[0].serviceId,
              });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },

    //Obtain the char ID of the connected device, read and write
  characteristics are respectively screened out.
    getBLEDeviceCharacteristics() {
      my.getConnectedBluetoothDevices({
    getBLEDeviceCharacteristics() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.getBLEDeviceCharacteristics({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            success: res => {
              my.alert({ content: JSON.stringify(res) });
              //See the related document for more information of the
  properties of the characteristics. Pair the characteristics according
  to the properties and record the value for later use.
```

```
                this.setData({
                  charid: res.characteristics[0].characteristicId,
                });
              },
              fail: error => {
                my.alert({ content: JSON.stringify(error) });
              },
            });
        },
      });
    },

    //Read and write data
    readBLECharacteristicValue() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.readBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.notifyId,
            //1  Android reading service
            // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
            success: res => {
              my.alert({ content: JSON.stringify(res) });
            },
            fail: error => {
              my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
            },
          });
        },
      });
    },

    writeBLECharacteristicValue() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
```

```
        });

        my.writeBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.charid,
          //Android writing service
          //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
          value: 'ABCD',
          success: res => {
            my.alert({ content: 'Succeeded to write data!' });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },
  notifyBLECharacteristicValueChange() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });

        my.notifyBLECharacteristicValueChange({
          state: true,
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          success: () => {
            //Listens to characteristic change events
            my.onBLECharacteristicValueChange({
              success: res => {
                //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                my.alert({ content: 'Obtain the response data = ' +
res.value });
              },
            });
            my.alert({ content: 'Succeeded to listen' });
          },
          fail: error => {
            my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
```

```
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChang
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
```

```
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | deviceId | String | The Bluetooth device ID. | | connected | Boolean | The current connection state. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_bluetooth_ble_onbleconnectionstatechanged

---

my.onBLEConnectionStateChanged {#myonbleconnectionstatechanged}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.onBLEConnectionStateChanged

2021-05-09 18:43

Use this API to listen to the Bluetooth Low Energy (BLE) connection error event, including device loss and unusual disconnections.

Instruction:

It is recommended that you call the `off` method and close event listening before you call the `on` method to listen events to prevent the situation where multiple listening event cause multiple callbacks of an event.

Note:

Currently simulation in IDE is not supported. Please debug in production environment.

Sample Code

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
```

```
}

.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>

    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>

    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
       <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
```

```
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>

     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
       <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>

     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
       <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
       <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>
    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
```

```
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {
            // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];
```

```
              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the connected device
  getConnectedBluetoothDevices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connecting devices!' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
        devid = res.devices[0].deviceId;
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
```

```
    },
  });
},

//Obtain all searched devices
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Disconnect the device
disconnectBLEDevice() {
  my.disconnectBLEDevice({
    deviceId: this.data.devid,
    success: () => {
      my.alert({ content: 'Succeeded to disconnect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain the services of the connected device
getBLEDeviceServices() {
  my.getConnectedBluetoothDevices({
```

```
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Obtain the char ID of the connected device, read and write
characteristics are respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            //See the related document for more information of the
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
```

```
      },
    });
  },

  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.readBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          //1  Android reading service
          // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
          success: res => {
            my.alert({ content: JSON.stringify(res) });
          },
          fail: error => {
            my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
          },
        });
      },
    });
  },

  writeBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });

        my.writeBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.charid,
          //Android writing service
          //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
```

```
            //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
            value: 'ABCD',
            success: res => {
              my.alert({ content: 'Succeeded to write data!' });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },
    notifyBLECharacteristicValueChange() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });

          my.notifyBLECharacteristicValueChange({
            state: true,
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.notifyId,
            success: () => {
              //Listens to characteristic change events
              my.onBLECharacteristicValueChange({
                success: res => {
                  //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                  my.alert({ content: 'Obtain the response data = ' +
res.value });
                },
              });
              my.alert({ content: 'Succeeded to listen' });
            },
            fail: error => {
              my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
            },
          });
        },
      });
    },
    offBLECharacteristicValueChange() {
      my.offBLECharacteristicValueChange();
    },
```

```
  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState(
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChang
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | deviceId | String | The Bluetooth device ID. | | connected | Boolean | The current connection state. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_ble_onbleconnectionstatechanged

---

my.onBluetoothAdapterStateChange {#myonbluetoothadapterstatechange}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.onBluetoothAdapterStateChange

2022-07-03 18:44

Use this API to monitor the bluetooth adapter state changes.

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
       <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
       <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
```

```
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
       <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
       <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>

    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
```

```
          my.alert({ content: 'Bluetooth closed!' });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },
    getBluetoothAdapterState() {
      my.getBluetoothAdapterState({
        success: res => {
          if (!res.available) {
            my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
            return;
          }
          my.alert({ content: JSON.stringify(res) });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },

    //Scan the Bluetooth device
    startBluetoothDevicesDiscovery() {
      my.startBluetoothDevicesDiscovery({
        allowDuplicatesKey: false,
        success: () => {
          my.onBluetoothDeviceFound({
            success: res => {
              // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
              var deviceArray = res.devices;
              for (var i = deviceArray.length - 1; i >= 0; i--) {
                var deviceObj = deviceArray[i];
                //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
                if (deviceObj.name == this.data.name) {
                  my.alert({ content: 'Target device is found' });
                  my.offBluetoothDeviceFound();
                  this.setData({
                    deviceId: deviceObj.deviceId,
                  });
                  break;
                }
              }
            },
            fail: error => {
              my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
            },
```

```
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the connected device
  getConnectedBluetoothDevices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connecting devices!' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
        devid = res.devices[0].deviceId;
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain all searched devices
  getBluetoothDevices() {
    my.getBluetoothDevices({
      success: res => {
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  bindKeyInput(e) {
```

```
    this.setData({
      devid: e.detail.value,
    });
  },

  //Connect the device
  connectBLEDevice() {
    my.connectBLEDevice({
      deviceId: this.data.devid,
      success: res => {
        my.alert({ content: 'Succeeded to connect!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Disconnect the device
  disconnectBLEDevice() {
    my.disconnectBLEDevice({
      deviceId: this.data.devid,
      success: () => {
        my.alert({ content: 'Succeeded to disconnect!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the services of the connected device
  getBLEDeviceServices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
```

```
      },
    });
  },

  //Obtain the char ID of the connected device, read and write
characteristics are respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            //See the related document for more information of the
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.readBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
```

```
            //1  Android reading service
            // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
            success: res => {
              my.alert({ content: JSON.stringify(res) });
            },
            fail: error => {
              my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
            },
          });
        },
      });
  },
  writeBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.writeBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.charid,
          //Android writing service
          //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
          value: 'ABCD',
          success: res => {
            my.alert({ content: 'Succeeded to write data!' });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },
  notifyBLECharacteristicValueChange() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
```

```
        });
        my.notifyBLECharacteristicValueChange({
          state: true,
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          success: () => {
            //Listens to characteristic change events
            my.onBLECharacteristicValueChange({
              success: res => {
                //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                my.alert({ content: 'Obtain the response data = ' +
res.value });
              },
            });
            my.alert({ content: 'Succeeded to listen' });
          },
          fail: error => {
            my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
```

```
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChang
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Success Callback Function

The input parameters are displayed in the following table:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | discovering | Boolean | Indicates whether bluetooth device is being discovered. | | available | Boolean | Indicates whether bluetooth is available (BLE should be supported and switched on). |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_bluetooth_bluetooth_onbluetoothadapterstatechange

---

my.onBluetoothAdapterStateChange {#myonbluetoothadapterstatechange}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.onBluetoothAdapterStateChange

2021-05-09 18:43

Use this API to monitor the bluetooth adapter state changes.

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
```

```
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
       <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
       <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
       <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
       <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>

    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },

  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
```

```
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {
            // my.alert({content:'Listens to new
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];
              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
```

```
//Obtain the connected device
getConnectedBluetoothDevices() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connecting devices!' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain all searched devices
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Disconnect the device
disconnectBLEDevice() {
  my.disconnectBLEDevice({
```

```
        deviceId: this.data.devid,
        success: () => {
          my.alert({ content: 'Succeeded to disconnect!' });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },

    //Obtain the services of the connected device
    getBLEDeviceServices() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          my.getBLEDeviceServices({
            deviceId: this.data.devid,
            success: res => {
              my.alert({ content: JSON.stringify(res) });
              this.setData({
                serid: res.services[0].serviceId,
              });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },

    //Obtain the char ID of the connected device, read and write
  characteristics are respectively screened out.
    getBLEDeviceCharacteristics() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.getBLEDeviceCharacteristics({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            success: res => {
              my.alert({ content: JSON.stringify(res) });
```

```
            //See the related document for more information of the
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.readBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          //1  Android reading service
          // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
          // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
          success: res => {
            my.alert({ content: JSON.stringify(res) });
          },
          fail: error => {
            my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  writeBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
```

```
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.writeBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.charid,
            //Android writing service
            //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
            value: 'ABCD',
            success: res => {
              my.alert({ content: 'Succeeded to write data!' });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
    },
    notifyBLECharacteristicValueChange() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.notifyBLECharacteristicValueChange({
            state: true,
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.notifyId,
            success: () => {
              //Listens to characteristic change events
              my.onBLECharacteristicValueChange({
                success: res => {
                  //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                  my.alert({ content: 'Obtain the response data = ' +
res.value });
                },
              });
              my.alert({ content: 'Succeeded to listen' });
            },
            fail: error => {
              my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
```

```
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChang
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
```

```
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Success Callback Function

The input parameters are displayed in the following table:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | discovering | Boolean | Indicates whether bluetooth device is being discovered. | | available | Boolean | Indicates whether bluetooth is available (BLE should be supported and switched on). |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_bluetooth_onbluetoothadapterstatechange

---

my.onBluetoothDeviceFound {#myonbluetoothdevicefound}

Last updated: 2021-05-09

Path: miniprogram_gcash

my.onBluetoothDeviceFound

2021-05-09 18:43

Use this API when a new Bluetooth device is found.

Instructions:

- You may not get the advertisData and RSSI in the emulator. Please debug in the guest.
- For Integrated Development Environment (IDE) and Android devices, the device ID is the MAC address of the device; for iOS devie, the device ID is the UUID of the device. Therefore, do not hard code the device ID. You need to process the device ID on different platforms; iOS devices can be dynamically matched based on properties such as localName, advertisData, and manufacturerData.
- If the API my.onBluetoothDeviceFound callback contains a bluetooth device, the device is added to the array obtained by the API my.getBluetoothDevices .

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
```

```
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
       <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
       <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
       <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
       <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>

    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },
```

```
  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {
            // my.alert({content:'Listens to new
```

```
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];
              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the connected device
  getConnectedBluetoothDevices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connecting devices!' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
```

```
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain all searched devices
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Disconnect the device
disconnectBLEDevice() {
  my.disconnectBLEDevice({
    deviceId: this.data.devid,
    success: () => {
      my.alert({ content: 'Succeeded to disconnect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
```

```
  //Obtain the services of the connected device
  getBLEDeviceServices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Obtain the char ID of the connected device, read and write
characteristics are respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            //See the related document for more information of the
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
```

```
          });
        },
      });
    },

    //Read and write data
    readBLECharacteristicValue() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.readBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.notifyId,
            //1  Android reading service
            // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
            success: res => {
              my.alert({ content: JSON.stringify(res) });
            },
            fail: error => {
              my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
            },
          });
        },
      });
    },
    writeBLECharacteristicValue() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.writeBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.charid,
            //Android writing service
            //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
```

```
            value: 'ABCD',
            success: res => {
              my.alert({ content: 'Succeeded to write data!' });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
  },
  notifyBLECharacteristicValueChange() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.notifyBLECharacteristicValueChange({
          state: true,
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          success: () => {
            //Listens to characteristic change events
            my.onBLECharacteristicValueChange({
              success: res => {
                //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                my.alert({ content: 'Obtain the response data = ' +
res.value });
              },
            });
            my.alert({ content: 'Succeeded to listen' });
          },
          fail: error => {
            my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
```

```
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChang
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Success Callback Function

The input parameters are displayed in the following table:

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | devices | Array | A list of all the devices that are newly discovered. |

Device Object

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | name | String | Name of the bluetooth device.(For some devices, there's no name.) | | deviceName (Compatible with initial version) | String | Name of the bluetooth device. | | localName | String | Name of the local device. | | deviceId | String | Device ID. | | RSSI | Number | Received Signal Strength Indicator. | | advertisData | Hex String | Advertisement data of the device. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_bluetooth_onbluetoothdevicefound

---

my.onBluetoothDeviceFound {#myonbluetoothdevicefound}

Last updated: 2022-07-03

Path: miniprogram_gcash

my.onBluetoothDeviceFound

2022-07-03 18:44

Use this API when a new Bluetooth device is found.

Instructions:

- You may not get the advertisData and RSSI in the emulator. Please debug in the guest.
- For Integrated Development Environment (IDE) and Android devices, the device ID is the MAC address of the device; for iOS devie, the device ID is the UUID of the device. Therefore, do not hard code the device ID. You need to process the device ID on different platforms; iOS devices can be dynamically matched based on properties such as localName, advertisData, and manufacturerData.
- If the API my.onBluetoothDeviceFound callback contains a bluetooth device, the device is added to the array obtained by the API my.getBluetoothDevices .

Note:

Currently simulation in IDE is not supported. Please debug in the production environment.

Code Sample

copy

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

copy

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

copy

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">The Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize
Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close
Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain
Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary"
onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices
found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All
devices connected</button>
       <button type="primary"
onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}"
placeholder="Enter the device ID of the device to connect"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the
```

```
device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain
device services</button>
       <button type="primary"
onTap="getBLEDeviceCharacteristics">Obtain read and write
characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect
the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="notifyBLECharacteristicValueChange">Listens to the
characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read
data</button>
       <button type="primary"
onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary"
onTap="offBLECharacteristicValueChange">Un-listens to characteristic
value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary"
onTap="bluetoothAdapterStateChange">Changes of the Bluetooth
state</button>
       <button type="primary"
onTap="offBluetoothAdapterStateChange">Un-listens to Bluetooth
state</button>
       <button type="primary"
onTap="BLEConnectionStateChanged">Changes of Bluetooth connection
state</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Un-
listens to Bluetooth connection state</button>

    </view>
  </view>
</view>
```

copy

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },
```

```
  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Succeeded to initialize!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is
unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Scan the Bluetooth device
  startBluetoothDevicesDiscovery() {
    my.startBluetoothDevicesDiscovery({
      allowDuplicatesKey: false,
      success: () => {
        my.onBluetoothDeviceFound({
          success: res => {
            // my.alert({content:'Listens to new
```

```
device'+JSON.stringify(res)});
            var deviceArray = res.devices;
            for (var i = deviceArray.length - 1; i >= 0; i--) {
              var deviceObj = deviceArray[i];
              //Pair the target device with the device name or
broadcast data, and then record the device ID for later use.
              if (deviceObj.name == this.data.name) {
                my.alert({ content: 'Target device is found' });
                my.offBluetoothDeviceFound();
                this.setData({
                  deviceId: deviceObj.deviceId,
                });
                break;
              }
            }
          },
          fail: error => {
            my.alert({ content: 'Failed to listen to new device' +
JSON.stringify(error) });
          },
        });
      },
      fail: error => {
        my.alert({ content: 'Failed to start scanning' +
JSON.stringify(error) });
      },
    });
  },

  //Stop scanning
  stopBluetoothDevicesDiscovery() {
    my.stopBluetoothDevicesDiscovery({
      success: res => {
        my.offBluetoothDeviceFound();
        my.alert({ content: 'Succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },

  //Obtain the connected device
  getConnectedBluetoothDevices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connecting devices!' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
```

```
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Obtain all searched devices
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},

//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
      my.alert({ content: 'Succeeded to connect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},

//Disconnect the device
disconnectBLEDevice() {
  my.disconnectBLEDevice({
    deviceId: this.data.devid,
    success: () => {
      my.alert({ content: 'Succeeded to disconnect!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
```

```
  //Obtain the services of the connected device
  getBLEDeviceServices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },

  //Obtain the char ID of the connected device, read and write
characteristics are respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            //See the related document for more information of the
properties of the characteristics. Pair the characteristics according
to the properties and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
```

```
          });
        },
      });
    },

    //Read and write data
    readBLECharacteristicValue() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.readBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.notifyId,
            //1  Android reading service
            // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
            success: res => {
              my.alert({ content: JSON.stringify(res) });
            },
            fail: error => {
              my.alert({ content: 'Failed to read' +
JSON.stringify(error) });
            },
          });
        },
      });
    },
    writeBLECharacteristicValue() {
      my.getConnectedBluetoothDevices({
        success: res => {
          if (res.devices.length === 0) {
            my.alert({ content: 'No connected devices' });
            return;
          }
          this.setData({
            devid: res.devices[0].deviceId,
          });
          my.writeBLECharacteristicValue({
            deviceId: this.data.devid,
            serviceId: this.data.serid,
            characteristicId: this.data.charid,
            //Android writing service
            //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
            //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
```

```
            value: 'ABCD',
            success: res => {
              my.alert({ content: 'Succeeded to write data!' });
            },
            fail: error => {
              my.alert({ content: JSON.stringify(error) });
            },
          });
        },
      });
  },
  notifyBLECharacteristicValueChange() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.notifyBLECharacteristicValueChange({
          state: true,
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
          success: () => {
            //Listens to characteristic change events
            my.onBLECharacteristicValueChange({
              success: res => {
                //  my.alert({content: 'Changes of
characteristics  '+JSON.stringify(res)});
                my.alert({ content: 'Obtain the response data = ' +
res.value });
              },
            });
            my.alert({ content: 'Succeeded to listen' });
          },
          fail: error => {
            my.alert({ content: 'Failed to listen' +
JSON.stringify(error) });
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },

  //Other events
```

```
  bluetoothAdapterStateChange() {

my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of the Bluetooth state  ' +
JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {

my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterState
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {

my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChange
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Changes of connection state  ' +
JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {

my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChang
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Success Callback Function

The input parameters are displayed in the following table: