easyble.js

Jump To ...

README. / app.coffeelibs / evothings / easyble / easyble.jslibs / evothings / tisensortag / tisensortag.jslibs / evothings / util / util.js. / pages.coffee

---

easyble.js

libs/evothings/easyble/

¶
File: easyble.js
Description: Library for making BLE programming easier.
Author: Miki
Note: The object type called "device" below, is the "DeviceInfo" object obtained by calling evothings.ble.startScan, enhanced with additional properties and functions to allow easy access to object methods. Properties are also added to the Characteristic and Descriptor object. Added properties are prefixed with two underscores.

```
if (!window.evothings) { window.evothings = {} }
```

¶
Object that holds BLE data and functions.

```
exports.easyble = (function()
{
```

¶
Main object in the EasyBLE API.

```
  var easyble = {};
```

¶

- Set to true to report found devices only once,
  - set to false to report continuously.

```
var reportDeviceOnce = false;
```

¶
Internal properties and functions.

```
var internal = {};
```

¶
Internal variable used to track reading of service data.

```
var readCounter = 0;
```

¶
Table of discovered devices.

```
internal.knownDevices = {};
```

¶
Table of connected devices.

```
internal.connectedDevices = {};
```

¶

- Set to true to report found devices only once,
  - set to false to report continuously.

```
easyble.reportDeviceOnce = function(reportOnce)
{
  reportDeviceOnce = reportOnce;
};
```

¶
Start scanning for devices.

```
easyble.startScan = function(win, fail)
{
  easyble.stopScan();
  internal.knownDevices = {};
  evothings.ble.startScan(function(device)
  {
```

¶
Check if we already have got the device.

```
    var existingDevice = internal.knownDevices[device.ad
    if (existingDevice)
    {
```

¶
Do not report device again if flag is set.

```
      if (reportDeviceOnce) { return; }
```

¶
Flag not set, report device again.

```
      existingDevice.rssi = device.rssi;
      existingDevice.name = device.name;
      win(existingDevice);
      return;
    }
```

¶
New device, add to known devices.

```
    internal.knownDevices[device.address] = device;
```

¶
Add methods to the device info object.

```
    internal.addMethodsToDeviceObject(device);
```

¶

Call callback function with device info.

```
    win(device);
  },
  function(errorCode)
  {
    fail(errorCode);
  });
};
```

¶

Stop scanning for devices.

```
easyble.stopScan = function()
{
  evothings.ble.stopScan();
};
```

¶

Close all connected devices.

```
easyble.closeConnectedDevices = function()
{
  for (var key in internal.connectedDevices)
  {
    var device = internal.connectedDevices[key];
    device && device.close();
    internal.connectedDevices[key] = null;
  }
};
```

¶

- Add functions to the device object to allow calling them
  - in an object-oriented style.

```
internal.addMethodsToDeviceObject = function(device)
{
```

¶

Connect to the device.

```
  device.connect = function(win, fail)
  {
    internal.connectToDevice(device, win, fail);
  };
```

¶

Close the device.

```
  device.close = function()
  {
    device.deviceHandle && evothings.ble.close(device.de
  };
```

¶

Read devices RSSI. Device must be connected.

```
  device.readRSSI = function(win, fail)
  {
    evothings.ble.rssi(device.deviceHandle, win, fail);
  };
```

¶

- Read all service info for the specified service UUIDs. * @param serviceUUIDs - array of UUID strings * @param win - success callback * @param fail - error callback * If serviceUUIDs is null, info for all services is read * (this can be time-consuming compared to reading a * selected number of services).

```
device.readServices = function(serviceUUIDs, win, fail
{
  internal.readServices(device, serviceUUIDs, win, fai
};
```

¶
Read value of characteristic.

```
device.readCharacteristic = function(characteristicUUI
{
  internal.readCharacteristic(device, characteristicUU
};
```

¶
Read value of descriptor.

```
device.readDescriptor = function(characteristicUUID, c
{
  internal.readDescriptor(device, characteristicUUID,
};
```

¶
Write value of characteristic.

```
device.writeCharacteristic = function(characteristicUU
{
  internal.writeCharacteristic(device, characteristicU
};
```

¶
Write value of descriptor.

```
device.writeDescriptor = function(characteristicUUID,
{
  internal.writeDescriptor(device, characteristicUUID,
};
```

¶
Subscribe to characteristic value updates.

```
device.enableNotification = function(characteristicUUI
{
  internal.enableNotification(device, characteristicUU
};
```

¶
Unsubscribe from characteristic updates.

```
device.disableNotification = function(characteristicUU
{
  internal.disableNotification(device, characteristicU
};
};
```

¶
Connect to a device.

```
internal.connectToDevice = function(device, win, fail)
{
  evothings.ble.connect(device.address, function(connect
  {
    if (connectInfo.state == 2) // connected
    {
      device.deviceHandle = connectInfo.deviceHandle;
      device.__uuidMap = {};
      internal.connectedDevices[device.address] = device

      win(device);
    }
    else if (connectInfo.state == 0) // disconnected
    {
      internal.connectedDevices[device.address] = null;
      fail && fail('disconnected');
    }
  },
  function(errorCode)
  {
    fail(errorCode);
  });
};
```

¶
TODO: How to signal disconnect? Call error callback? Additional callback? (connect, disconnect, fail) Additional parameter on win callback with connect state? (Last one is the best option I think).

¶

- Obtain device services, them read characteristics and descriptors
  - for the services with the given uuid(s).
  - If serviceUUIDs is null, info is read for all services.

```
internal.readServices = function(device, serviceUUIDs, w
{
```

¶
Read services.

```
  evothings.ble.services(
    device.deviceHandle,
    function(services)
    {
```

¶

Array that stores services.

```
      device.__services = [];

      for (var i = 0; i < services.length; ++i)
      {
        var service = services[i];
        device.__services.push(service);
        device.__uuidMap[service.uuid] = service;
      }

      internal.readCharacteristicsForServices(
        device, serviceUUIDs, win, fail);
    },
    function(errorCode)
    {
      fail(errorCode);
    });
};
```

¶

- Read characteristics and descriptors for the services with the given uuid(s).
  - If serviceUUIDs is null, info for all services are read.
  - Internal function.

```
internal.readCharacteristicsForServices = function(devic
{
  var characteristicsCallbackFun = function(service)
  {
```

¶

Array with characteristics for service.

```
    service.__characteristics = [];

    return function(characteristics)
    {
      --readCounter; // Decrements the count added by se
      readCounter += characteristics.length;
      for (var i = 0; i < characteristics.length; ++i)
      {
        var characteristic = characteristics[i];
        service.__characteristics.push(characteristic);
        device.__uuidMap[characteristic.uuid] = characte
```

¶
Read descriptors for characteristic.

```
      evothings.ble.descriptors(
        device.deviceHandle,
        characteristic.handle,
        descriptorsCallbackFun(characteristic),
        function(errorCode)
        {
          fail(errorCode);
        });
    }
  };
};

var descriptorsCallbackFun = function(characteristic)
{
  characteristic.__descriptors = [];

  return function(descriptors)
  {
    --readCounter; // Decrements the count added by ch
    for (var i = 0; i < descriptors.length; ++i)
    {
      var descriptor = descriptors[i];
      characteristic.__descriptors.push(descriptor);
      device.__uuidMap[characteristic.uuid + ':' + des
    }
    if (0 == readCounter)
    {
      win(device);
    }
  };
};
```

¶
Array with descriptors for characteristic.

¶
Everything is read.

¶
Initialize read counter.

```
readCounter = 0;

if (null != serviceUUIDs)
{
  readCounter = serviceUUIDs.length;
  for (var i = 0; i < serviceUUIDs.length; ++i)
  {
    var uuid = serviceUUIDs[i];
    var service = device.__uuidMap[uuid];
    if (!service)
    {
      fail('Service not found: ' + uuid);
      return;
    }
```

¶
Read info for service UUIDs.

¶
Read characteristics for service. Will also read descriptors.

```
      evothings.ble.characteristics(
        device.deviceHandle,
        service.handle,
        characteristicsCallbackFun(service),
        function(errorCode)
        {
          fail(errorCode);
        });
    }
  }
  else
  {
```

¶
Read info for all services.

```
    readCounter = device.__services.length;
    for (var i = 0; i < device.__services.length; ++i)
    {
```

¶
Read characteristics for service. Will also read descriptors.

```
      var service = device.__services[i];
      evothings.ble.characteristics(
        device.deviceHandle,
        service.handle,
        characteristicsCallbackFun(service),
        function(errorCode)
        {
          fail(errorCode);
        });
    }
  }
};

internal.readCharacteristic = function(device, characte
{
  var characteristic = device.__uuidMap[characteristicUU
  if (!characteristic)
  {
    fail('Characteristic not found: ' + characteristicUU
    return;
  }

  evothings.ble.readCharacteristic(
    device.deviceHandle,
    characteristic.handle,
    win,
    fail);
};

internal.readDescriptor = function(device, characteristi
{
  var descriptor = device.__uuidMap[characteristicUUID +
  if (!descriptor)
  {
    fail('Descriptor not found: ' + descriptorUUID);
    return;
  }

  evothings.ble.readDescriptor(
    device.deviceHandle,
    descriptor.handle,
    value,
    function()
    {
      win();
    },
    function(errorCode)
    {
      fail(errorCode);
    });
};

internal.writeCharacteristic = function(device, characte
{
  var characteristic = device.__uuidMap[characteristicUU
  if (!characteristic)
```

¶
For debugging. Example call: easyble.printObject(device, console.log);

```
easyble.printObject = function(obj, printFun)
{
  function print(obj, level)
  {
    var indent = new Array(level + 1).join('  ');
    for (var prop in obj)
    {
      if (obj.hasOwnProperty(prop))
      {
        var value = obj[prop];
        if (typeof value == 'object')
        {
          printFun(indent + prop + ':');
          print(value, level + 1);
        }
        else
        {
          printFun(indent + prop + ': ' + value);
        }
      }
    }
  }
  print(obj, 0);
};

easyble.reset = function()
{
  evothings.ble.reset();
};

return easyble;
})();
```

---

generated Sun Apr 26 2015 14:18:06 GMT+1000 (ChST)

Stag App