WebUSBで低まるWeb開発

NAOMASA MATSUBAYASHI

WebUSB API

http://wicg.github.io/webusb

WebUSB API

ホストに接続されたUSBデバイスとブラウザが

OSのデバイスドライバを介さずに

直接会話するためのJavaScriptのAPI

ブラウザ

カーネル

USBデバイス

libusbをWebから叩けるようになったと思えば

だいたいあっている

デバイスを取得する

```
return navigator.usb.requestDevice(
  { 'filters': [
    { 'vendorId': 0x1234, 'productId': 0x5678 }
  ] }
).then(device_ => {
  device = device_;
  return connect();
}).catch(error => {
  console.log('接続エラー: ' + error);
});
```

navigator.usb.requestDevieで
指定したベンダIDとプロダクトIDを持つ
USBデバイスへの接続を要求する

デバイスを取得する

こんなプロンプトが出てくる
ユーザはWebページに触らせて良いデバイスを選ぶ

通信できる状態にする

configurationを選ぶ
USBデバイスは複数の機能を提供していることがある
どの機能を使うかを選ぶのがconfigurationの選択

```
return device.open().then(() => {
  if(device.configuration === null) {
    return device.selectConfiguration(1);
  }
}).then(() => {
  return device.claimInterface(1);
}).then(() => {
  return device.claimInterface(0);
});
```

必要なインターフェースを占有する

デバイスにデータを送る/貰う

デバイスから受信

エンドポイントID

受信するバイト数

```
device.transferIn(1, 1024).then(result => {
  something_on_recieved( result.data );
}
```

Promiseで受信結果とDataViewが返って来る

デバイスに送信

エンドポイントID

送信するArrayBuffer

```
device.transferOut(2, data).then(result => {
  something_on_sent();
}
```

Promiseで送信結果が返って来る

デバイスにデータを送る/貰う
コントロール転送

```
device.controlTransferOut({
    'requestType': 'class',
    'recipient': 'interface',
    'request': 0x22,
    'value': 0x01,
    'index': 0x00
}).then(result => {
  …
}
```

```
device.controlTransferIn({
    'requestType': 'standard',
    'recipient': 'device',
    'request': 0x06,
    'value': 0x0100,
    'index': 0x0000
}, 0x12).then(result => {
  …
}
```

デバイスの情報の取得や
デバイスの初期化を行う

CDC ACM シリアル通信

Webブラウザの中と外で会話してみよう

```
$ modprobe libcomposite
$ modprobe dummy_hcd
$ cd /sys/kernel/config/usb_gadget
$ mkdir g1
$ cd g1
$ mkdir functions/acm.g1
$ mkdir configs/c.1
$ ln -s  functions/acm.g1 configs/c.1
$ echo 0x1234 >idVendor
$ echo 0x5678 >idProduct
$ echo dummy_udc.0 >UDC
$ sleep 1
$ modprobe -r cdc_acm
$ agetty 115200 ttyGS0
```

LinuxのUSB Gadgetで
CDC ACMのシリアル通信デバイスを作り
デバイス側でgettyする
ホスト側ドライバはアンロードしておく

```
function flush() {
  writing = true;
  let buffer_ = buffer;
  buffer = '';
  device.transferOut(2, textEncoder.encode(buffer_)).then(
    result => {
    writing = false;
    if( buffer.length != 0 ) { flush(); }
  }).catch(error => {
    console.log('送信エラー: ' + error);

    writing = false;
    if( buffer.length != 0 ) { flush(); }
  });
}
t.open( document.getElementById('terminal') );
t.on('key', function (key, ev) {
  if(device !== undefined) {
    if( !writing ) {
      buffer += key;
      flush();
    }
    else { buffer += key; }
  }
});
```

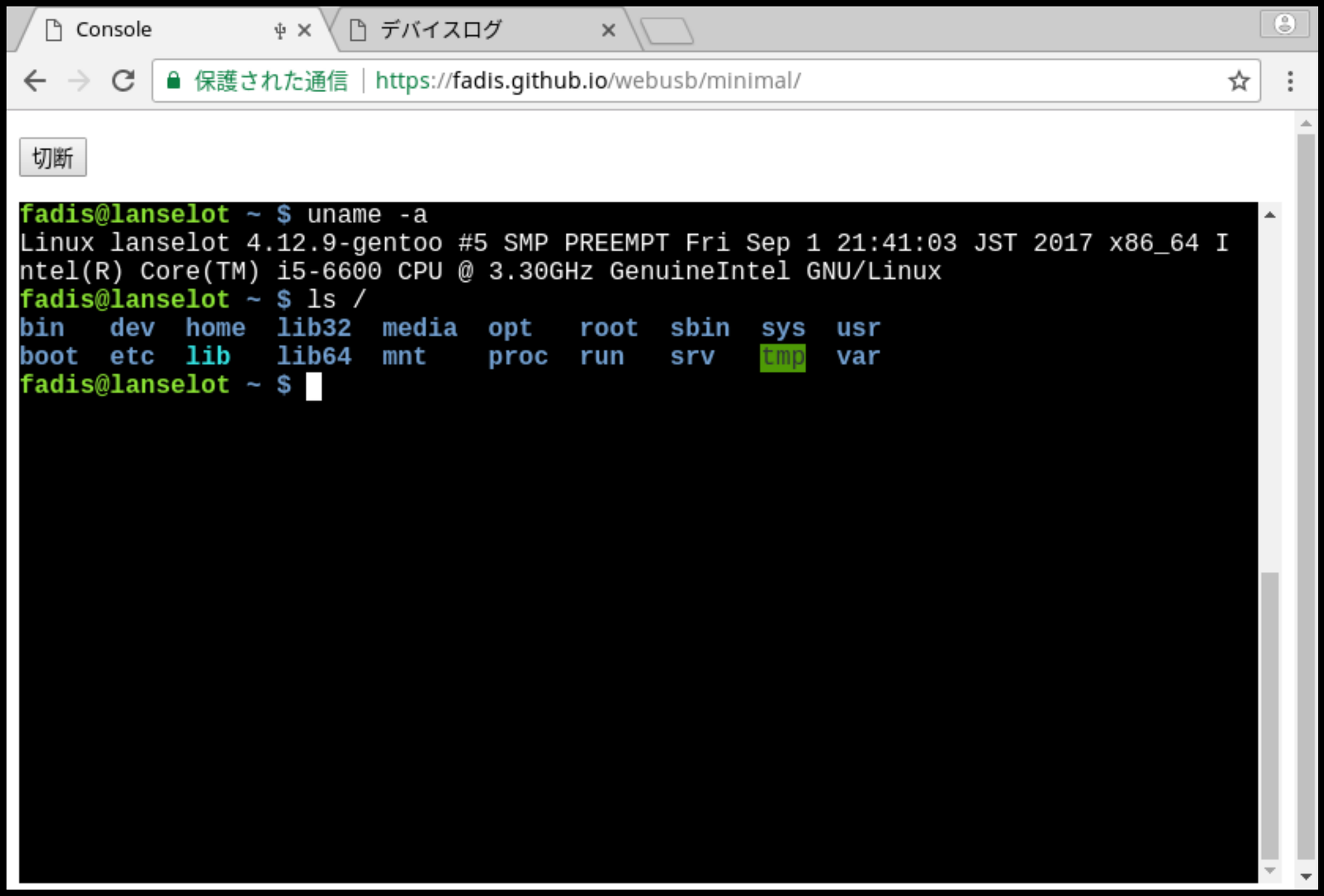
xterm.jsにキー入力があったら

エンドポイント2に入力内容を流す

```
let readLoop = () => {
  if( device ) {
    device.transferIn(1, 1024).then(result => {
      let textDecoder = new TextDecoder();
      t.write(textDecoder.decode(result.data));
      readLoop();
    }, error => {
      console.log( error );
      readLoop();
    });
  }
};
```

エンドポイント1では
デバイスからのデータを待ち受け
何か受け取ったらxterm.jsに流す

USBを通って外の世界に飛び出すよ!



http://fadis.github.io/webusb/minimal/

USBマスストレージ

USB外付けハードディスクや
USBフラッシュメモリと会話してみよう

このUSBフラッシュメモリを

ブラウザから読む

```
$ modprobe -r uas
$ modprobe -r mass_storage
```

まずカーネルが邪魔をしないようにしておく

Bulk Only Transport

Command Block Wrapper

SCSIコマンド

コマンドに付随する
データを送受信

Command State Wrapper

コマンドの実行結果

細かいストレージの操作方法はSCSIそのまま

Command Block Wrapper

```
let cbw = ( tag, len, dir, command ) => {
  let data = new Uint8Array( 15 + 16 );
  data.set([
    0x55, 0x53, 0x42, 0x43, // USBC
    tag & 0xFF,
    ( tag >> 8 ) & 0xFF,
    ( tag >> 16 ) & 0xFF,
    ( tag >> 24 ) & 0xFF, // tag
    len & 0xFF,
    ( len >> 8 ) & 0xFF,
    ( len >> 16 ) & 0xFF,
    ( len >> 24 ) & 0xFF, // len
    dir << 7, // flags
    0, // LUN
    command.byteLength // command length
  ], 0);
  data.set( command, 15 );
  return device.transferOut( 2, data );
}
```

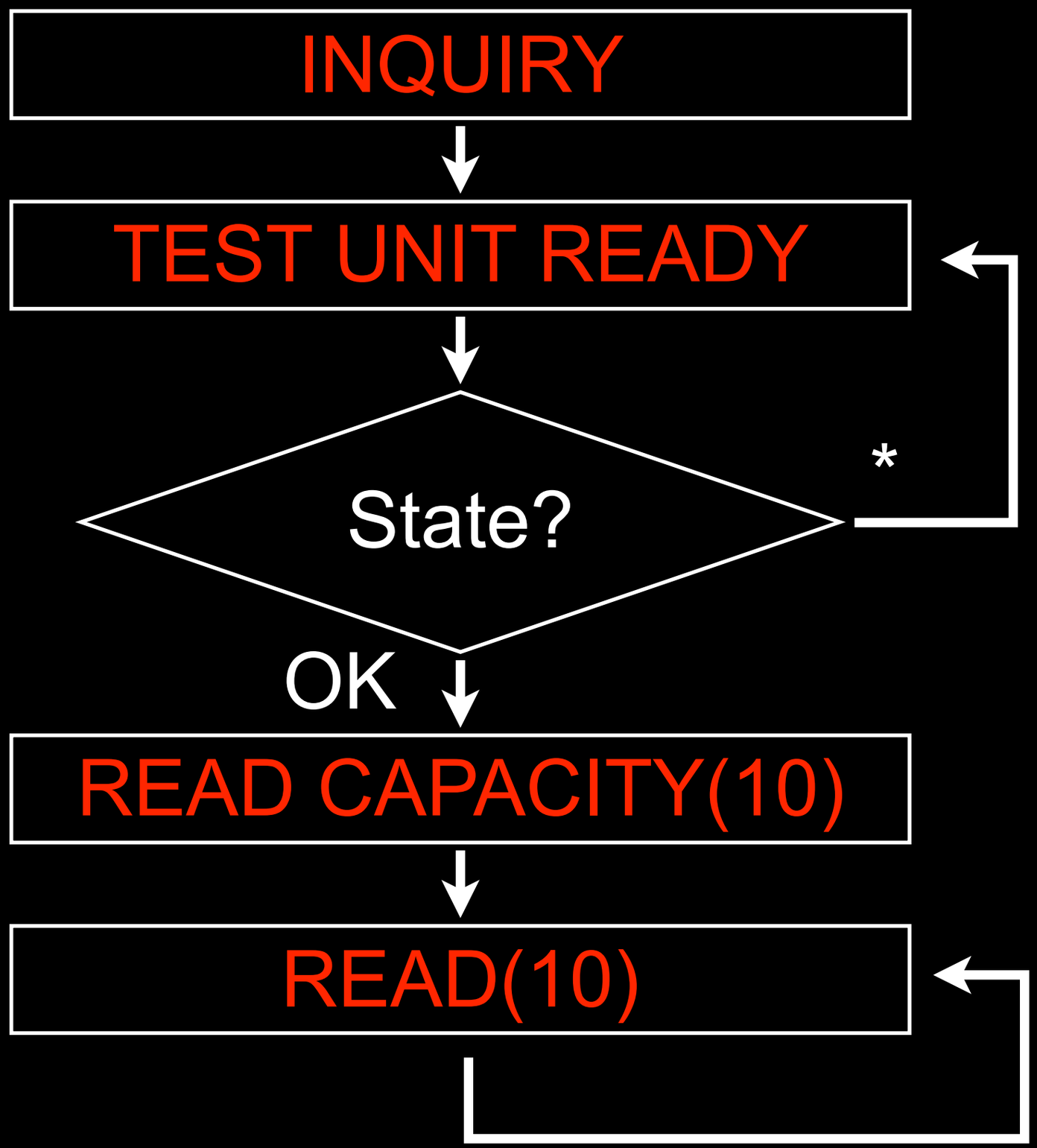
SCSIコマンドに
CBWの
ヘッダをつけて
デバイスに投げる

Command State Wrapper

```
let csw = () => {
  return device.transferIn( 1, 13 ).then(
    result => {
      let state = result.data.getUint8( 12 );
      return state == 0;
    }
  );
}
```

コマンドの実行結果をデバイスから受け取る

ストレージを読むのに必要なSCSIコマンド



LUN0にデバイスが
あることを確認

デバイスが
利用可能になるのを待つ

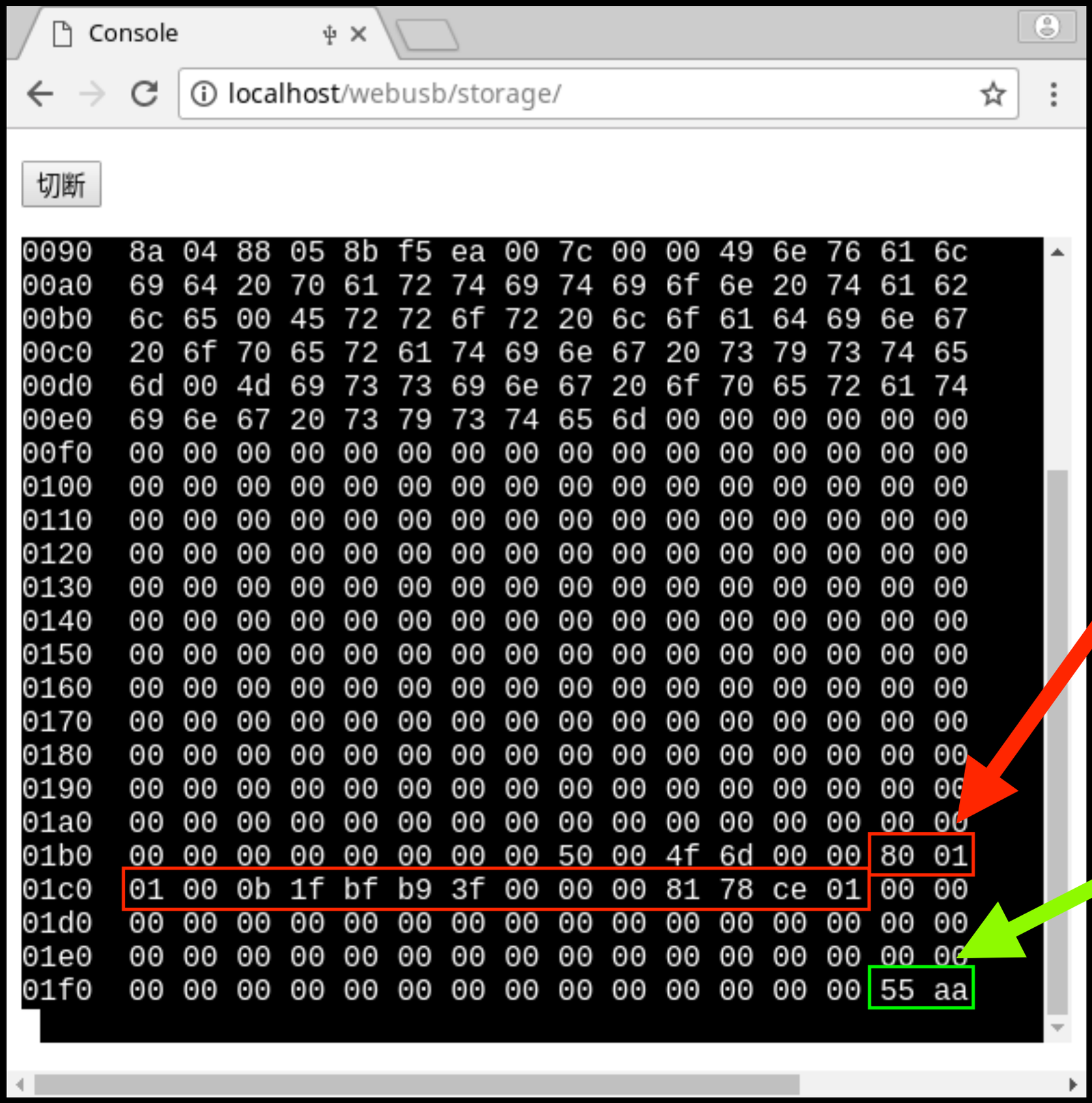
デバイスの容量と
セクタサイズを取得

デバイスから
データを読み出す

必要なSCSIコマンドを実装していく

```
let inquiry = () => {
  let command = new Uint8Array([
    // INQUIRY LUN reserved reserved size reserved
    0x12, 0, 0, 0, 36, 0
  ]);
  return cbw( 1, 36, 1, command ).then( result => {
    return device.transferIn( 1, 36 );
  }).then( result => {
    return result.data;
  }).then( data => {
    return csw().then( stat => {
      return { 'status': stat, 'data': data }
    });
  });
}
```

マスターブートレコード



LBA0から
1ブロックを取得

パーティション
テーブルの
1つめのエントリ

ブートシグニチャ

55 aa

BIOSパラメータブロック

Console

localhost/webusb/storage0/

切断

```
0000  eb 58 90 54 48 52 45 45 53 20 20 00 02 20 35 00
0010  02 00 00 00 00 f8 00 00 3f 00 20 00 3f 00 00 00
0020  81 78 ce 01 e6 1c 00 00 00 00 00 00 02 00 00 00
0030  01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00
0040  00 00 29 c8 58 32 5c 54 4f 53 48 49 42 41 20 20
0050  20 20 46 41 54 33 32 20 20 20 33 c9 8e d1 bc f4
0060  7b 8e c1 8e d9 bd 00 7c 88 4e 02 8a 56 40 b4 08
0070  cd 13 73 05 b9 ff ff 8a f1 66 0f b6 c6 40 66 0f
0080  b6 d1 80 e2 3f f7 e2 86 cd c0 ed 06 41 66 0f b7
0090  c9 66 f7 e1 66 89 46 f8 83 7e 16 00 75 38 83 7e
00a0  2a 00 77 32 66 8b 46 1c 66 83 c0 0c bb 00 80 b9
00b0  01 00 e8 2b 00 e9 48 03 a0 fa 7d b4 7d 8b f0 ac
00c0  84 c0 74 17 3c ff 74 09 b4 0e bb 07 00 cd 10 eb
00d0  ee a0 fb 7d eb e5 a0 f9 7d eb e0 98 cd 16 cd 19
00e0  66 60 66 3b 46 f8 0f 82 4a 00 66 6a 00 66 50 06
00f0  53 66 68 10 00 01 00 80 7e 02
0100  41 bb aa 55 8a 56 40 cd 13 0f
0110  aa 0f 85 14 00 f6 c1 01 0f 84
0120  42 8a 56 40 8b f4 cd 13 b0 f9
0130  66 58 eb 2a 66 33 d2 66 0f b7
0140  c2 8a ca 66 8b d0 66 c1 ea 10
0150  56 40 8a e8 c0 e4 06 0a cc b8 01 02 cd 13 66 61
0160  0f 82 54 ff 81 c3 00 02 66 40 49 0f 85 71 ff c3
0170  4e 54 4c 44 52 20 20 20 20 20 20 00 00 00 00 00
```

FAT32のヘッダ

46 41 54 33 32 20 20 20

F A T 3 2

FAT32

```
let load_fat32 = partition => {
  return read( partition.at, 1 ).then( result => {
    let cluster_size = result.data.getUint8( 13 );
    let reserved_sector_size =
      result.data.getUint16( 14, true );
    let num_fats = result.data.getUint8( 16 );
    let rde_size = result.data.getUint16( 17, true );
    let fat_size = result.data.getUint32( 36, true );
    let rde = result.data.getUint32( 44, true );
    let fat_lba = partition.at + reserved_sector_size;
```

BIOSパラメータブロックから
ファイルシステムを読むのに必要な値を取得

クラスタサイズ

FATの数とサイズと開始セクタ

ルートディレクトリが書かれいてる位置

FATを読む

```
let fat_lba = partition.at + reserved_sector_size;
return read( fat_lba, fat_size ).then( result => {
  let len = result.data.byteLength / 4;
  let fat = new Uint32Array( len );
  for( let i = 0; i != len; i++ ) {
    fat[ i ] = result.data.getUint32( i * 4, true );
  }
  let clusters_lba = fat_lba + num_fats * fat_size;
```

FATには
今読んでいるクラスタの次に読むべきクラスタが
どこかが記録されている

ルートディレクトリを読む

```
let clusters_lba = fat_lba + num_fats * fat_size;
let fsinfo = {
  'cluster_size': cluster_size, 'fat': fat,
  'clusters_lba': clusters_lba, 'rootdir_entry': rde
};
return load_fat32file(
  fsinfo, fsinfo.rootdir_entry, 0
).then( raw_root_dir => {
  let root_dir =
    parse_fat32directory( raw_root_dir );
  fsinfo[ 'rootdir' ] = root_dir;
  return fsinfo;
});
});
});
};
```

クラスタチェイン

```
let get_fat32clusters_reversed =
  ( fsinfo, head ) => {
   let next = fsinfo.fat[ head ] & 0x0FFFFFFF;
   if( next >= 0x00000002 && next <= 0x0ffffff6 ) {
     let tail =
       get_fat32clusters_reversed( fsinfo, next );
     tail.push( head );
     return tail;
   }
   else return [ head ];
};
```

FATから読む必要があるクラスタを調べて

クラスタチェイン

```
let get_fat32clusters = ( fsinfo, head ) => {
  let clusters = get_fat32clusters_reversed( fsinfo, head );
  clusters.reverse();
  let chunks = [ { 'at': clusters[ 0 ], 'length': 1 } ];
  for( let i = 1; i != clusters.length; i++ ) {
    if( clusters[ i - 1 ] + 1 == clusters[ i ] ) {
      chunks[ chunks.length - 1 ].length++;
    }
    else {
      chunks.push( { 'at': clusters[ i ], 'length': 1 } );
    }
  }
  return chunks;
}
```

連続したクラスタはまとめて

クラスタチェイン

```
let load_fat32cluster = ( fsinfo, chunks, index, data ) => {
  let lba = fsinfo.clusters_lba +
    ( chunks[ index ].at - 2 ) * fsinfo.cluster_size;
  return read(
    lba, chunks[ index ].length * fsinfo.cluster_size
  ).then(
    result => {
    data.push( result.data );
    if( index + 1 < chunks.length ) {
      return load_fat32cluster(
        fsinfo, chunks, index + 1, data
      );
    }
    else return data;
  });
}
```

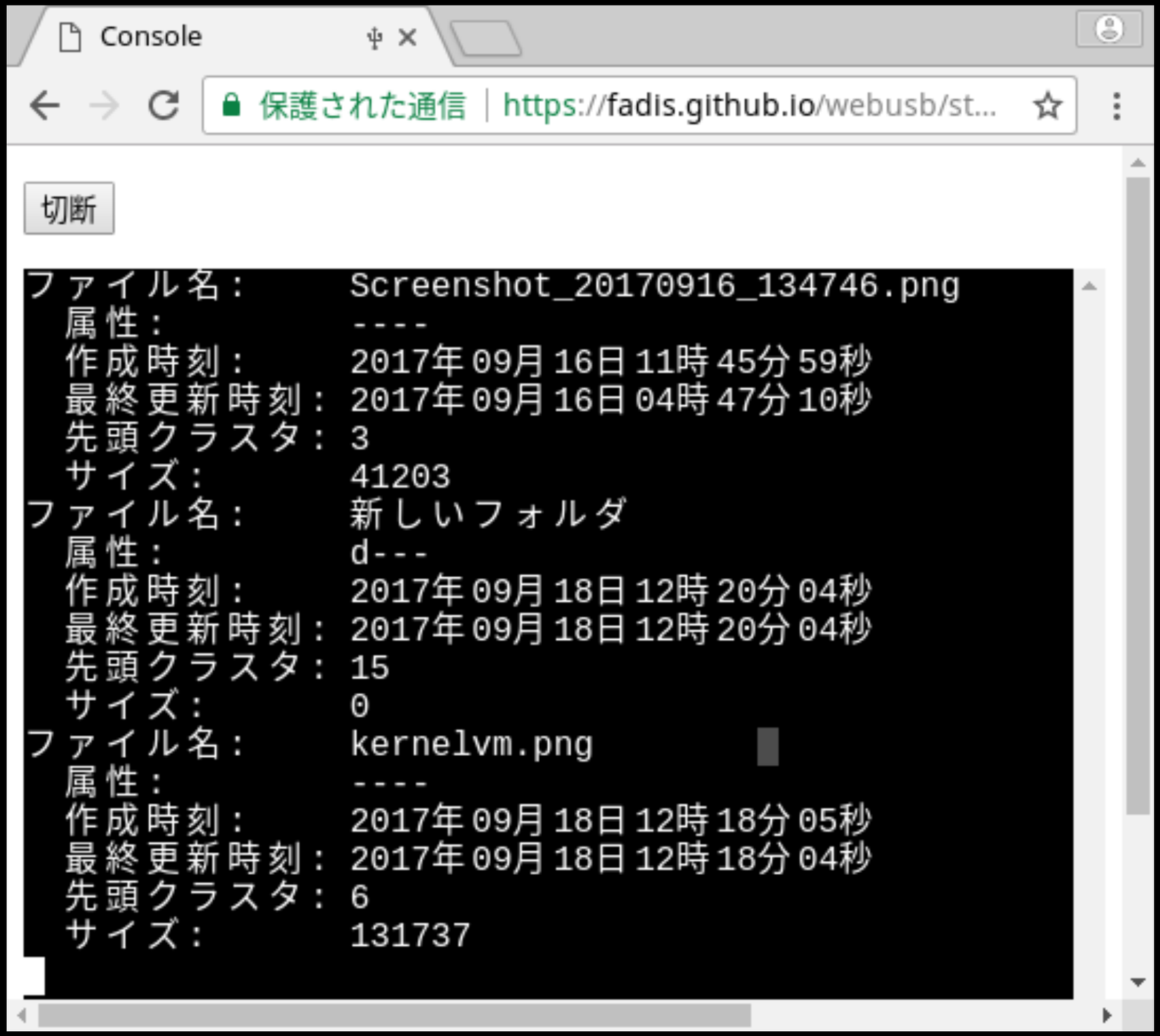
READ(10)

ディレクトリエントリ

```
let parse_fat32directory = ( data ) => {
  let files = [];
  let lfn = [];
  for( let offset = 0; offset != data.byteLength; offset += 32 ) {
    let entry = data.subarray( offset, offset + 32 );
    let attribute = entry[ 11 ];
    let sfn_head = entry[ 0 ];
    if( sfn_head == 0x00 ) {
      break;
    }
    else if( sfn_head == 0xE5 ) {
      continue;
    }
    else if( attribute == 0x0F ) {
      let fragment = new Uint8Array( 26 );
      fragment.set( entry.subarray( 1, 11 ), 0 );
      fragment.set( entry.subarray( 14, 26 ), 10 );
      fragment.set( entry.subarray( 28, 32 ), 22 );
      lfn.push( fragment );
    }
```

そんなに複雑じゃないので気合いでパースする

ブラウザでUSBフラッシュメモリを読むよ!



http://fadis.github.io/webusb/storage/

USBフラッシュメモリから読んだ画像をWebページに表示するよ!

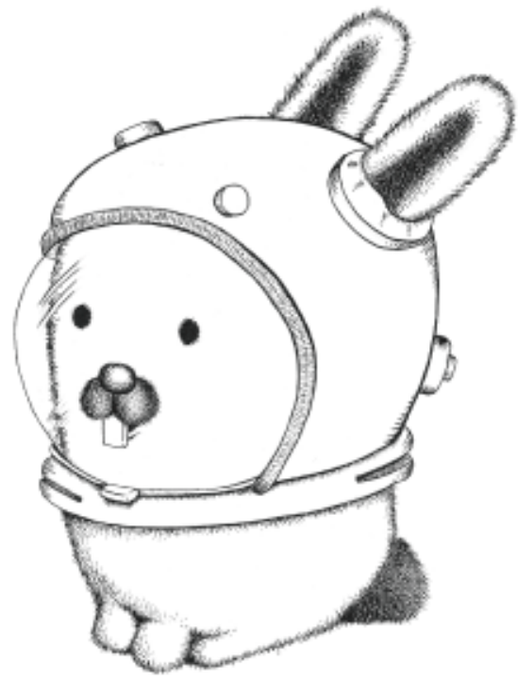
Console

保護された通信 | https://fadis.github.io/webusb/storage/

VendorID 2352 ProductID 5128

File path kernelvm.png

切断



KERNEL/VM 探検隊

```
ファイル名：      Screenshot_20170916_134746.png
  属性：          ----
  作成時刻：      2017年09月16日 11時45分59秒
  最終更新時刻：  2017年09月16日 11時47分10秒
  先頭クラス：
  サイズ：
```

TOSHIBA

検索

名前	変更日
kernelvm.png	一昨日 12:18
Screenshot_20170916_134746.png	2017年9月16日 4:47
新しいフォルダ	一昨日 12:19

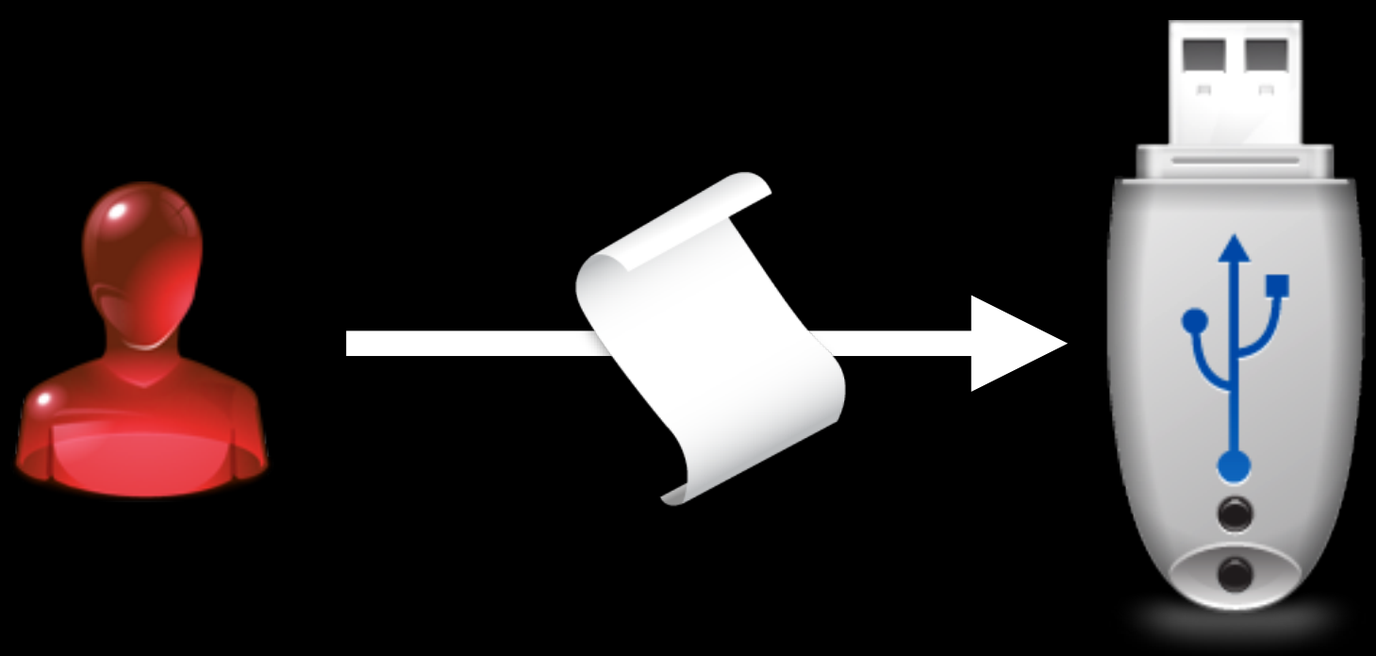
http://fadis.github.io/webusb/storage/

WebUSBのセキュリティ

WebUSBはUSBデバイスに対する
あらゆる操作を行う権限を
リモートから飛んで来たJavaScriptに与える

デバイスによっては**壊せる**

デバイスによっては**BadUSB**にできる



WebUSBのセキュリティ

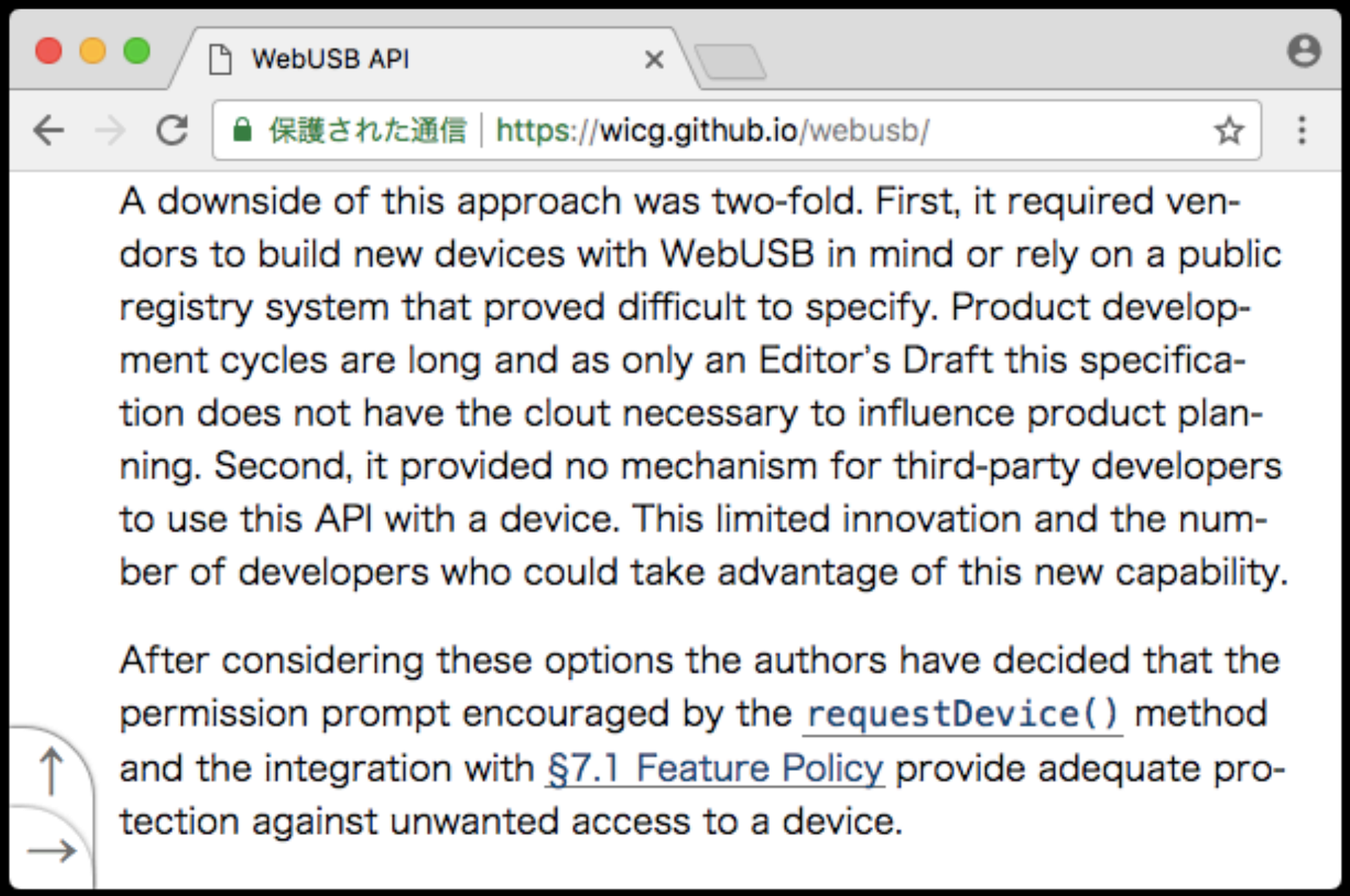
当初WebUSBはデバイスに情報を埋める事で
意図しないWebサイトに
デバイスを使われないようにしようとした

https://example.org/
なんだけど
USBデバイスを使わせてよ

https://example.com/
で使われる為に作られた
WebUSB対応デバイスです

ダメ

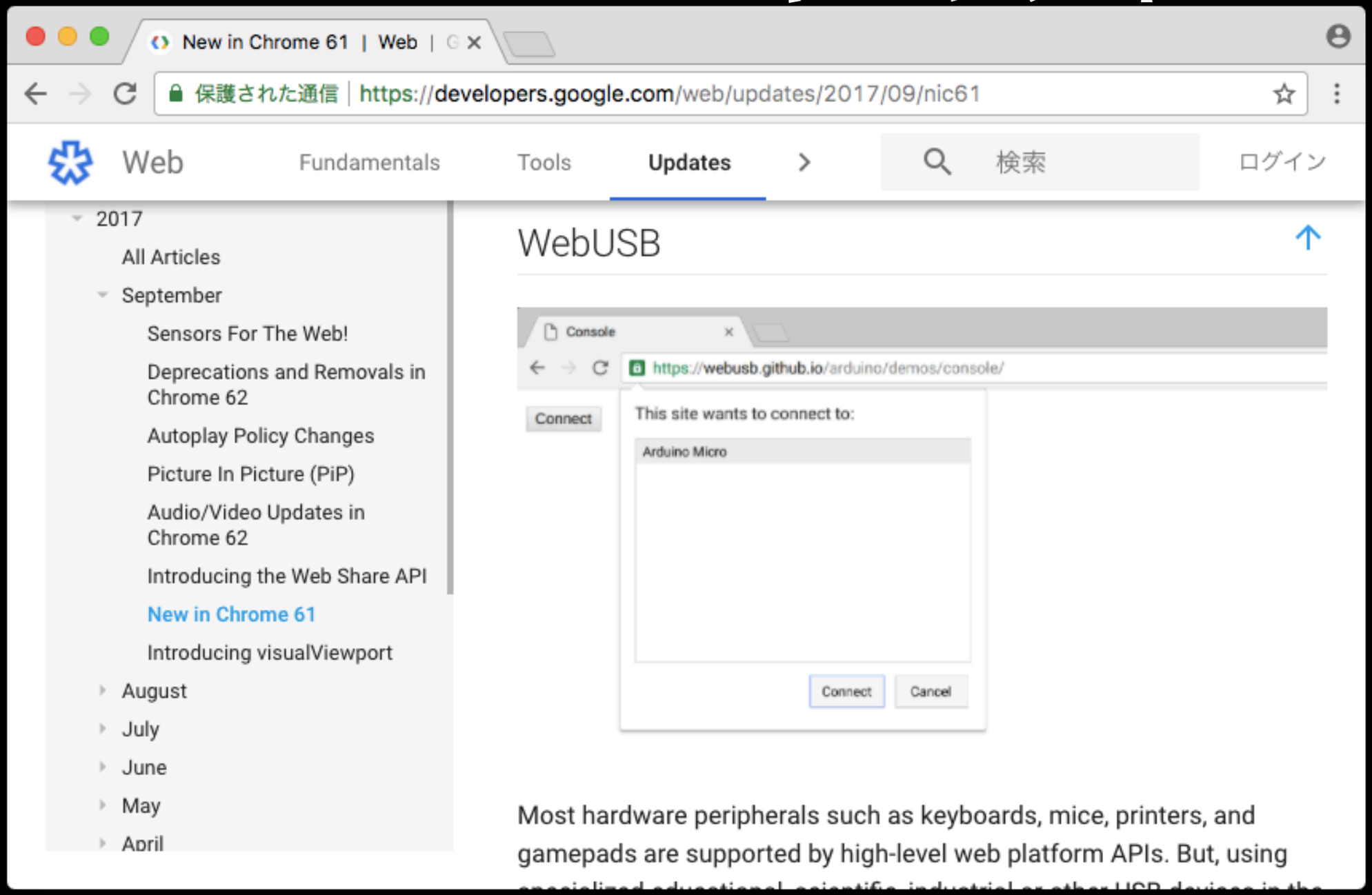
当初WebUSBはデバイスに情報を埋める事で
意図しないWebサイトに
デバイスを使われないようにしようとした



http://wicg.github.io/webusb/

が、諸般の理由から諦めた

WebUSBのセキュリティ



https://developers.google.com/web/updates/2017/09/nic61

デバイス接続プロンプト以外に
悪意あるドライバの実行を防ぐ術がない
WebUSBが安定版のChromeに実装された

WebUSBのセキュリティ

このプロンプトの意味は

閲覧中のWebページが得体の知れないドライバで
USBデバイスを好き勝手する事に同意しますか?

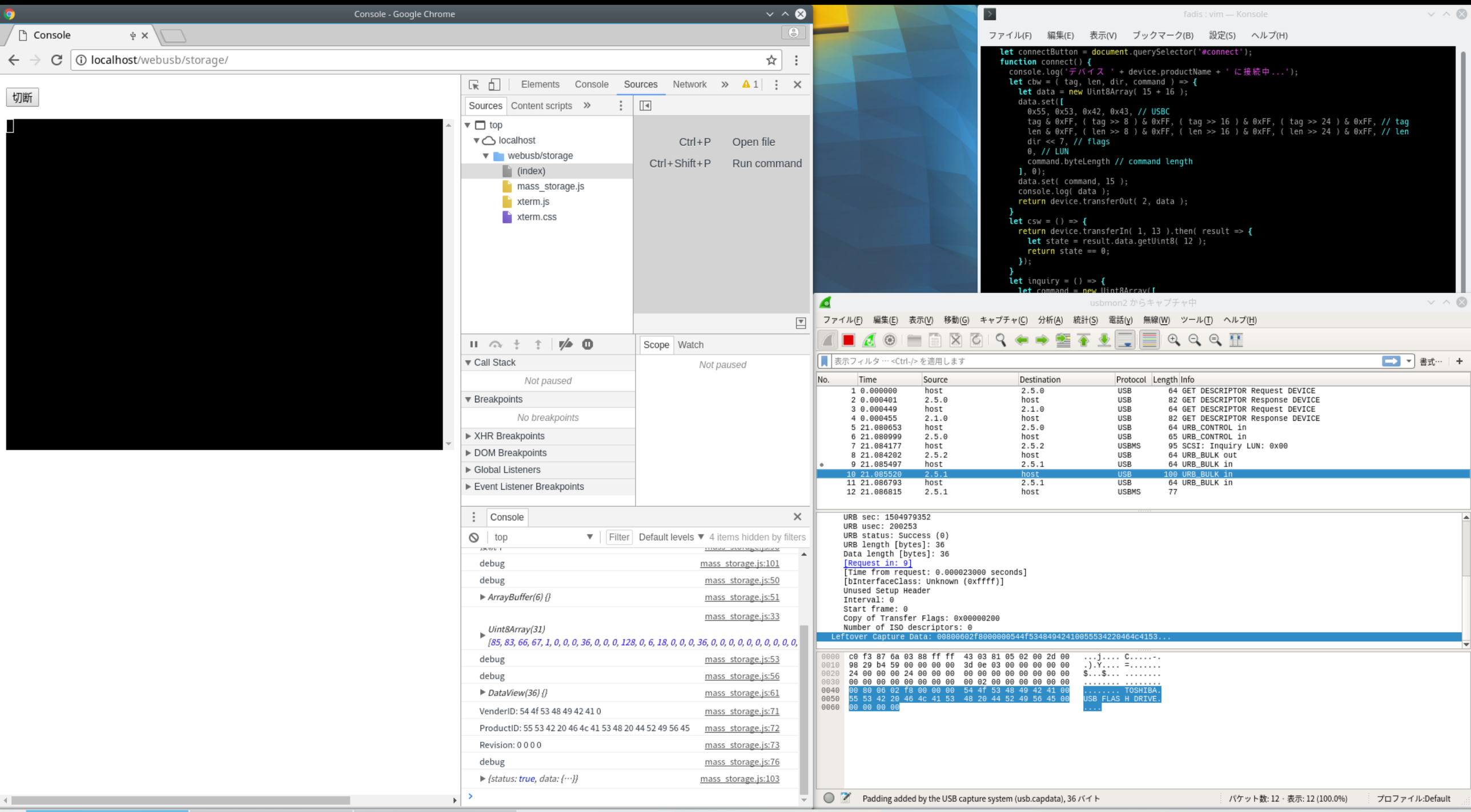
まとめ

WebUSBはブラウザとUSBデバイスが
直接会話することを可能にするAPIである

JavaScriptでドライバを書けば
どんなUSBデバイスでも動かすことができる

でもリモートから降って来た
得体の知れないドライバは信用できるのか

デバッグ風景



うまく通信できない時はWiresharkでUSBを見よう