



WebUSBで 低まるWeb開発

レイヤーが

NAOMASA MATSUBAYASHI

WebUSB API



WebUSB API

Editor's Draft, 10 September 2017

This version:
<https://wicg.github.io/webusb/>

Issue Tracking:
[GitHub](#)
[Inline In Spec](#)

Editors:
[Reilly Grant](#) (Google Inc.)
[Ken Rockot](#) (Google Inc.)

Participate:
[Join the W3C Community Group](#)
[IRC: #webusb on W3C's IRC](#) (Stay around for an answer, it make take a while)
[Ask questions on StackOverflow](#)

Copyright © 2017 the Contributors to the WebUSB API Specification, published by the [Web Platform Incubator Community Group](#) under the [W3C Community Contributor License Agreement \(CLA\)](#). A human-readable [summary](#) is available.

<https://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.requestDevice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; }
```

```
  }
```

```
});
```

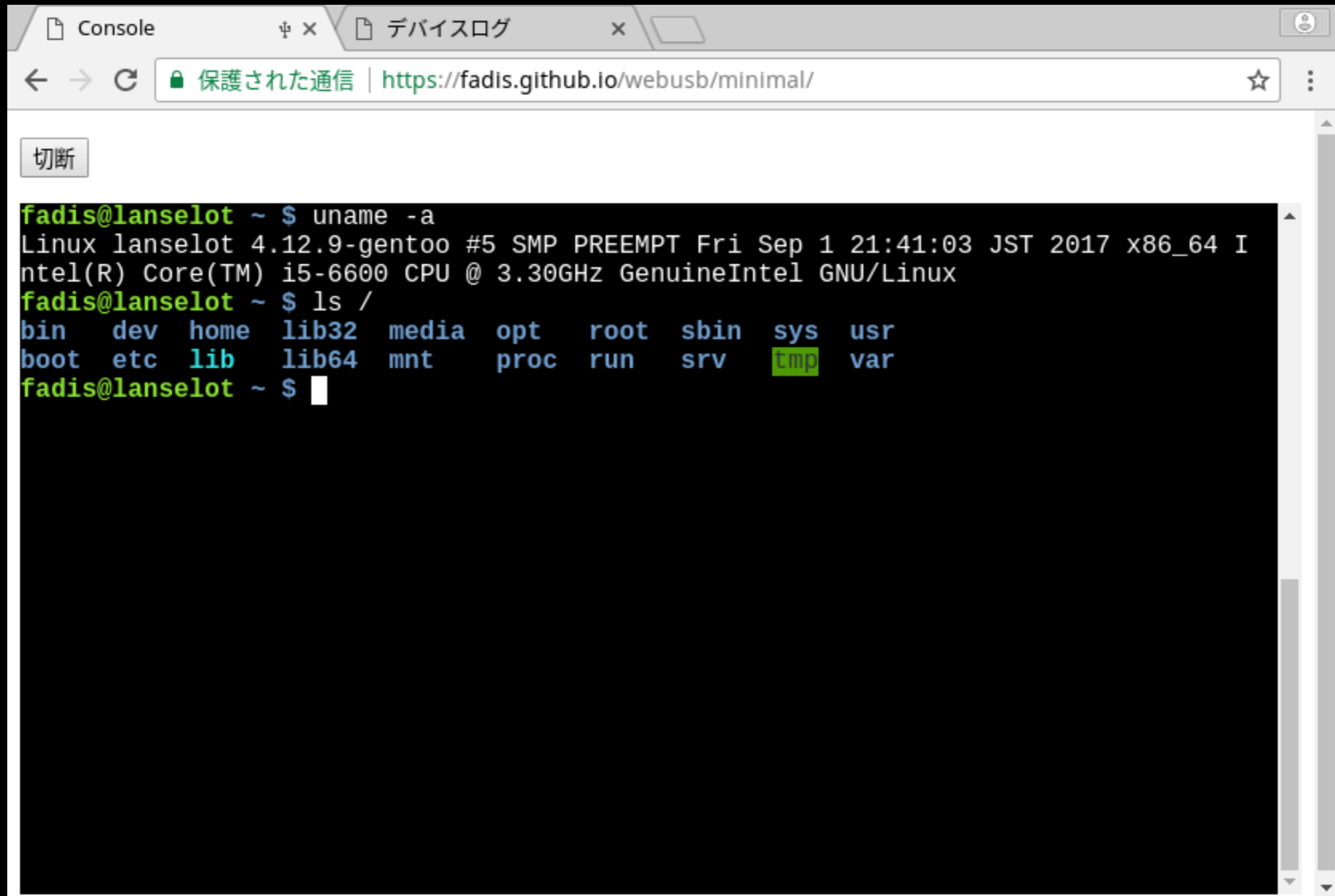
xterm.jsにキー入力があったら

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を通して外の世界に飛び出すよ!



The screenshot shows a web browser window with the address bar displaying `https://fadis.github.io/webusb/minimal/`. The browser tabs include "Console" and "デバイスログ". A "切断" (Disconnect) button is visible in the top left. The main content is a terminal window with the following text:

```
fadis@lanselot ~ $ uname -a
Linux lanselot 4.12.9-gentoo #5 SMP PREEMPT Fri Sep 1 21:41:03 JST 2017 x86_64 I
ntel(R) Core(TM) i5-6600 CPU @ 3.30GHz GenuineIntel GNU/Linux
fadis@lanselot ~ $ ls /
bin    dev  home  lib32  media  opt    root  sbin  sys  usr
boot  etc  lib   lib64  mnt    proc   run   srv   tmp  var
fadis@lanselot ~ $
```

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

USBマスタストレージ



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

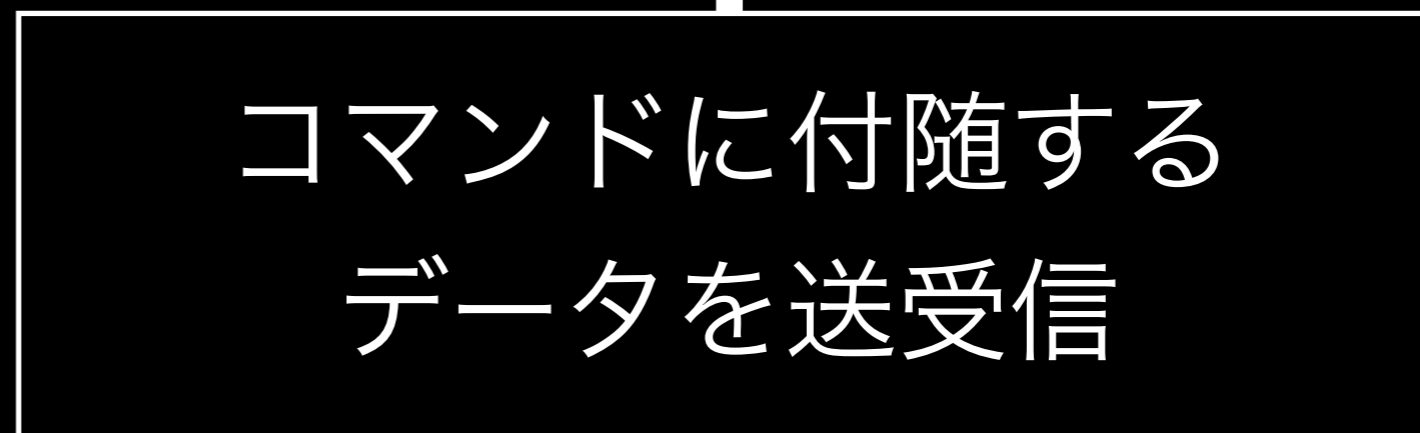
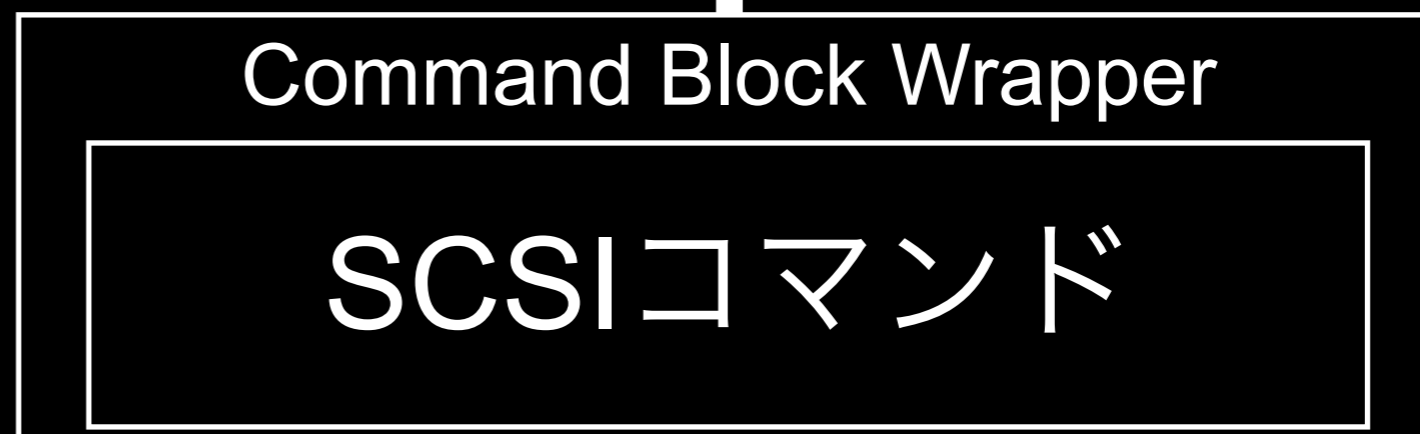


このUSBフラッシュメモリを
ブラウザから読む

```
$ 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



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

INQUIRY

LUN0にデバイスがあることを確認

TEST UNIT READY

State?

* デバイスが利用可能になるのを待つ

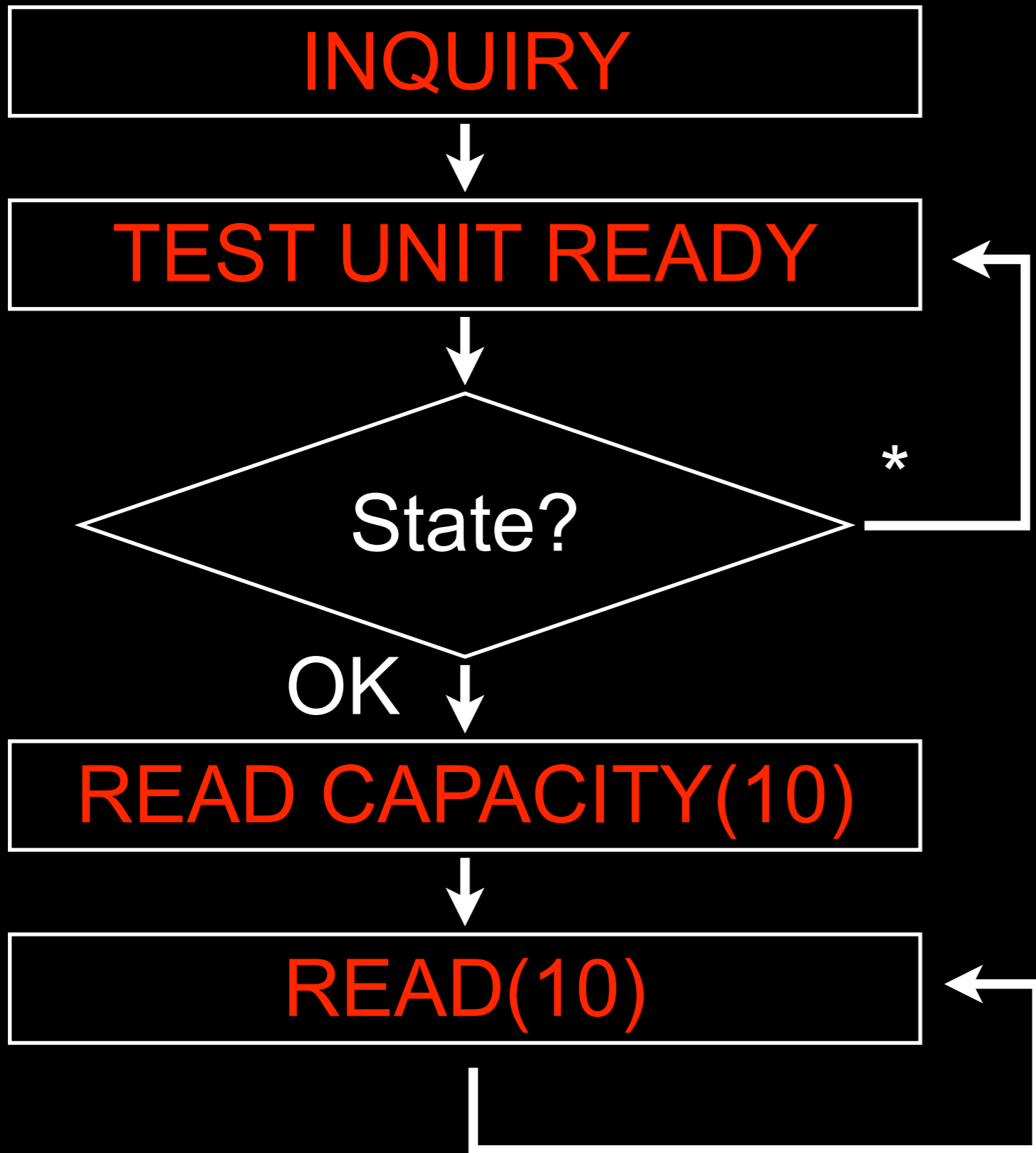
OK

READ CAPACITY(10)

デバイスの容量とセクタサイズを取得

READ(10)

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

```
Console
localhost/webusb/storage/
切断
0090 8a 04 88 05 8b f5 ea 00 7c 00 00 49 6e 76 61 6c
00a0 69 64 20 70 61 72 74 69 74 69 6f 6e 20 74 61 62
00b0 6c 65 00 45 72 72 6f 72 20 6c 6f 61 64 69 6e 67
00c0 20 6f 70 65 72 61 74 69 6e 67 20 73 79 73 74 65
00d0 6d 00 4d 69 73 73 69 6e 67 20 6f 70 65 72 61 74
00e0 69 6e 67 20 73 79 73 74 65 6d 00 00 00 00 00 00
00f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01b0 00 00 00 00 00 00 00 00 00 50 00 4f 6d 00 00 80 01
01c0 01 00 0b 1f bf b9 3f 00 00 00 81 78 ce 01 00 00
01d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 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 70 88 4e 02 8a 56 40 b4 08
0070 cd 13 73 05 b9 ff ff 8a 11 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 71 16 00 75 38 83 7e
00a0 2a 00 77 32 66 8b 46 1c 66 83 20 0c bb 00 80 b9
00b0 01 00 e8 2b 00 e9 48 03 a0 fa 70 b4 7d 8b f0 ac
00c0 84 c0 74 17 3c ff 74 09 b4 0e bb 17 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ている位置

FATを読む

```
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;
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;
  let fsinfo = {
```

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

```
return load( fat_lba, fat_size, fsinfo, fsinfo.rootdir_entry, 0
).then( raw_root_dir => {
  let root_dir =
    parse_fat32directory( raw_root_dir );
  fsinfo.rootdir_lba = root_dir
```


ルートディレクトリを読む

```
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;
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 <= 0x0fffffff6 ) {  
      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

クラスチェーン

```
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;
    }
  );
}
```

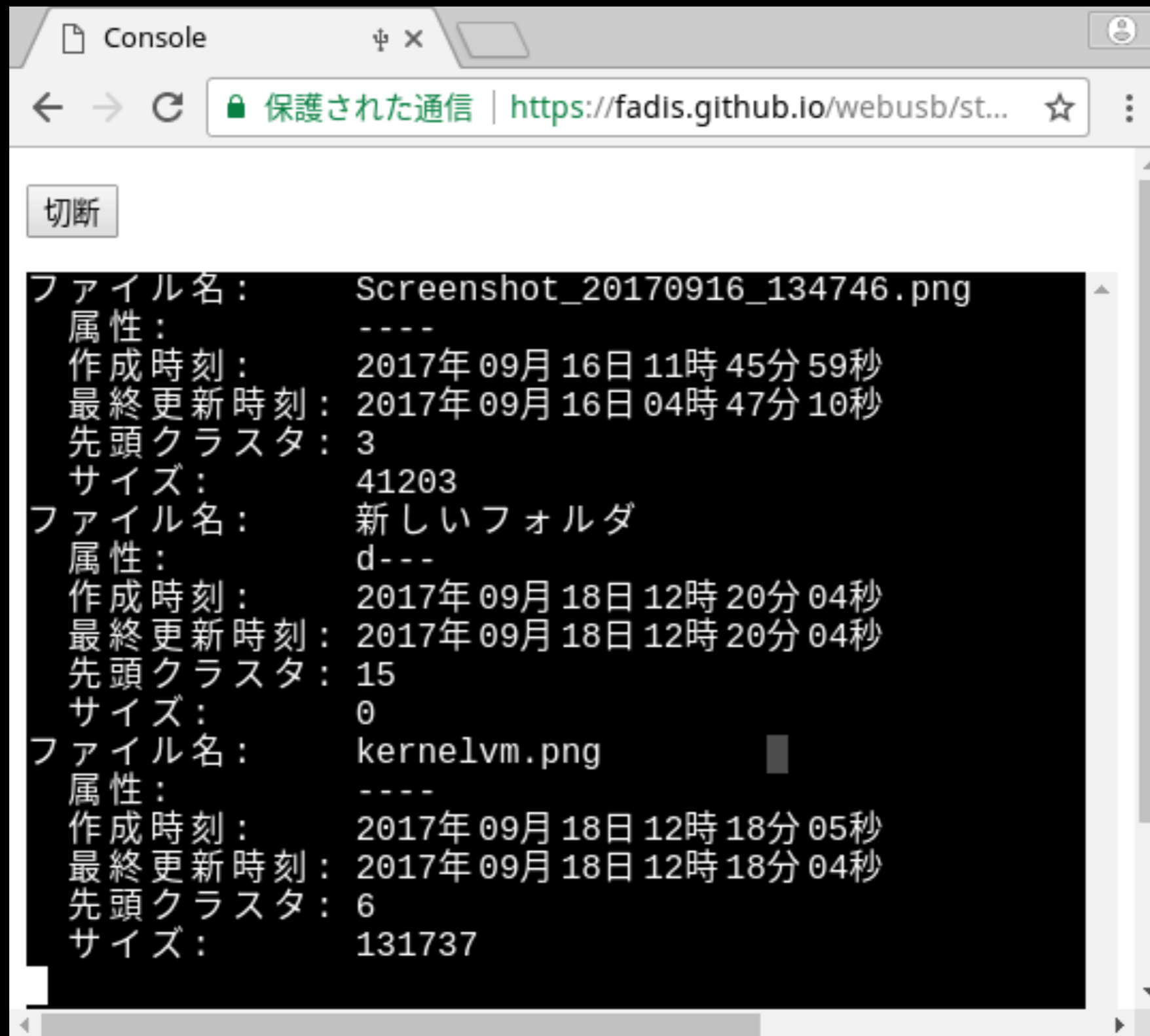
READ(10)

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 );
    }
  }
}
```

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 ProductID

File path

切断

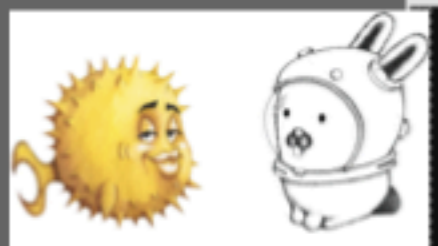


KERNELVM 探検隊

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
先頭クォーター: 1024
サイズ: 1024

TOSHIBA

検索



kernelvm.png 探検隊

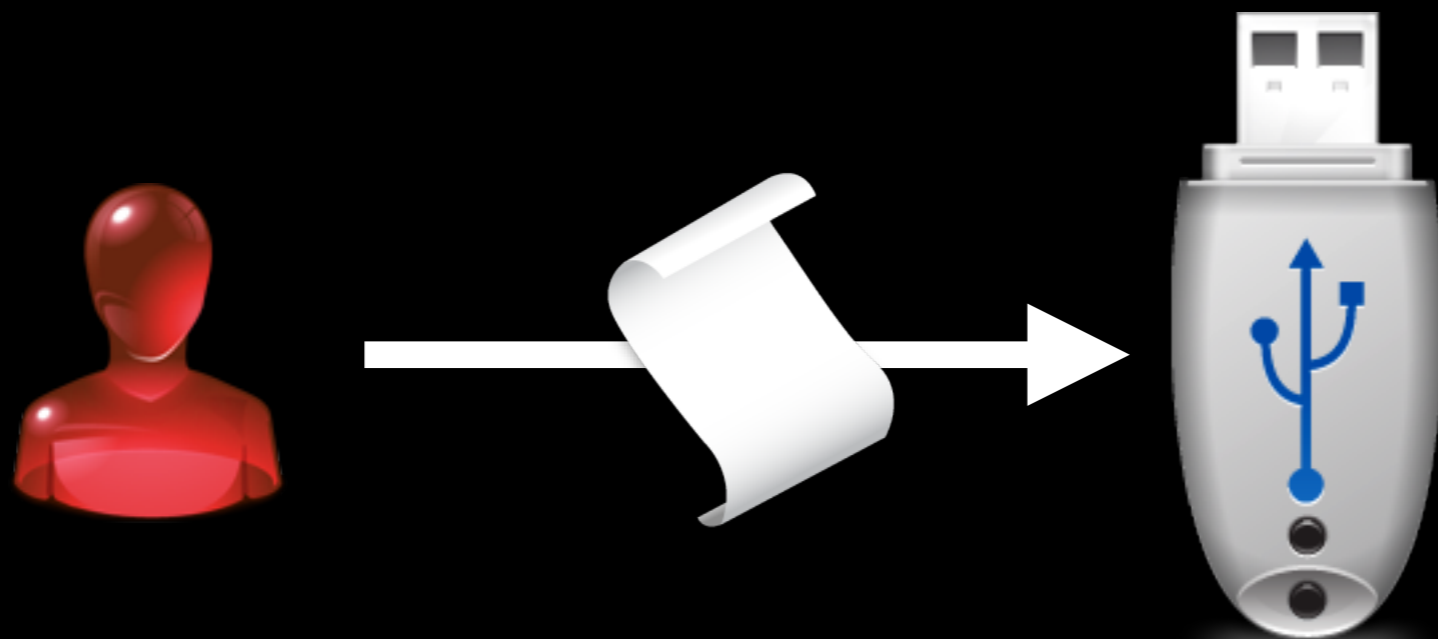
名前	変更日
kernelvm.png	一昨日 12:18
Screenshot_20170916_134746.png	2017年9月16日 4:47
新しいフォルダ	一昨日 12:19

<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

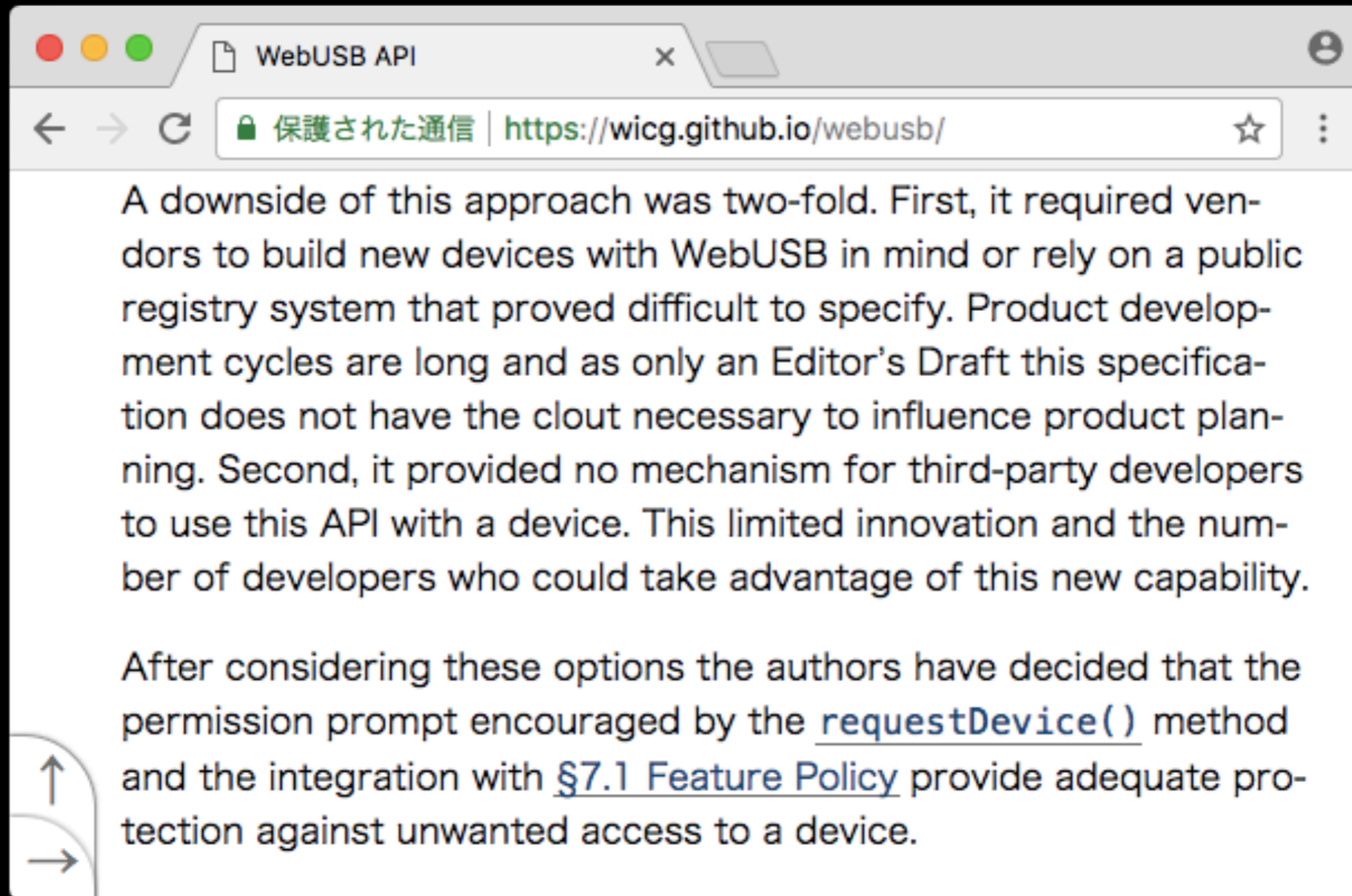
<https://example.com/>
で使われる為に作られた

WebUSB対応デバイスです

ダメ

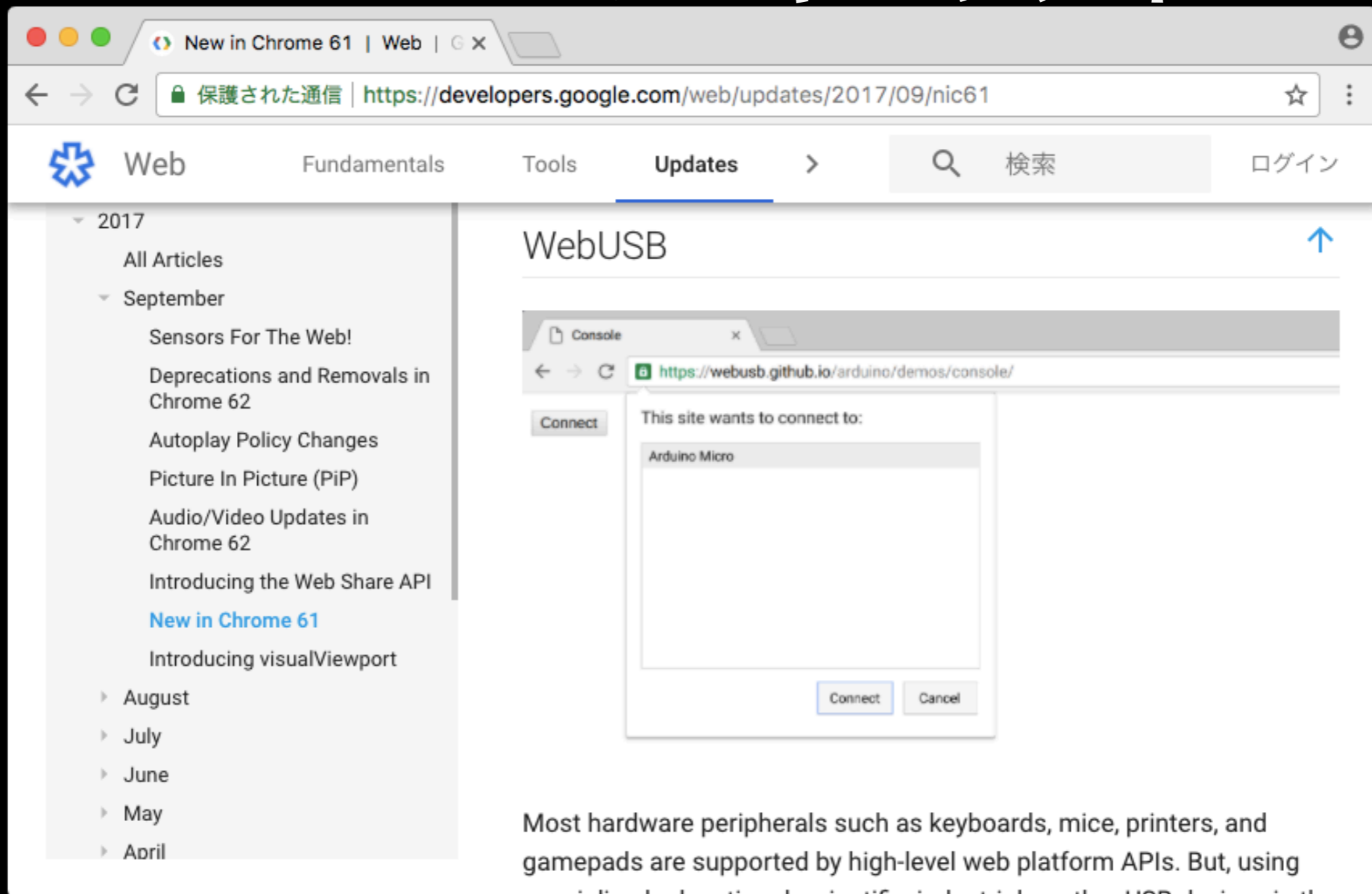


当初WebUSBはデバイスに情報を埋める事で
意図しないWebサイトに
デバイスを使われないようにしようとした



<http://wicg.github.io/webusb/>
が、諸般の理由から諦めた

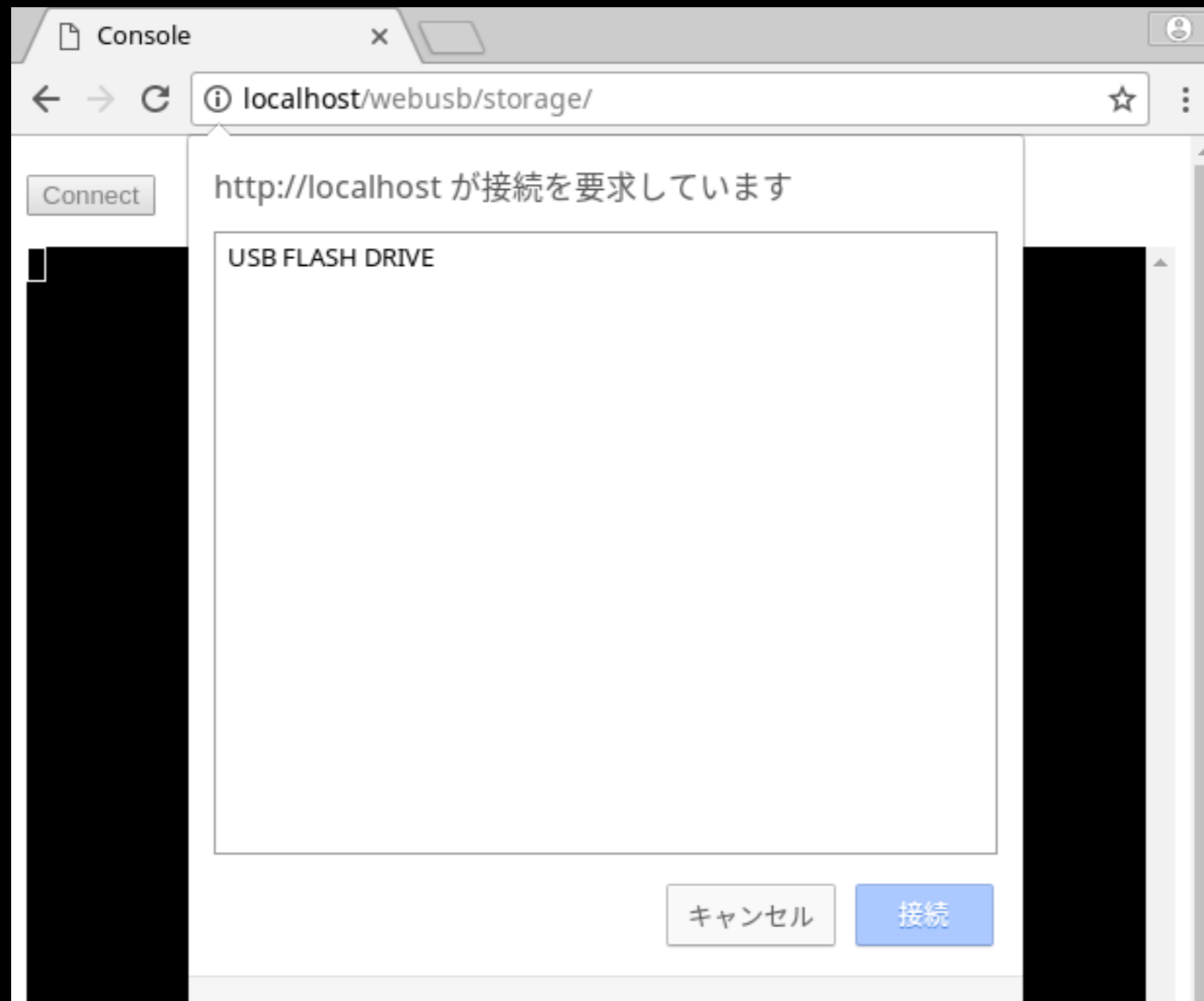
WebUSBのセキュリティ



<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

The image displays a multi-paneled development and debugging environment. On the left, a Chrome browser window shows the developer tools interface for 'localhost/webusb/storage/'. The 'Sources' pane highlights the 'mass_storage.js' file. The 'Console' pane shows several debug logs, including an 'ArrayBuffer(6) {}' and a 'Uint8Array(31)' containing binary data. The 'Call Stack' pane shows the current execution context in 'mass_storage.js:101'. In the center, a terminal window shows JavaScript code for connecting to a USB device, including functions for 'connect', 'csw', and 'inquiry'. On the right, the Wireshark interface shows a captured USB packet (No. 10) with details for 'USBMS' and 'USB BULK in'. The packet data section shows hexadecimal and ASCII representations of the captured data, including the string 'TOSHIBA USB FLAS H DRIVE'. The bottom status bar indicates 'Padding added by the USB capture system (usb.capdata), 36 バイト' and 'パケット数: 12 / 表示: 12 (100.0%)'.

うまく通信できない時はWiresharkでUSBを見よう