CSS Nite LP54

Coder's High 2017

2017.11.4

CSS Nite LP54

2017.11.04

ここまでのJavaScriptのスタンダードとこれからのJavaScript ES6について

KDDIウェブコミュニケーションズ
阿部 正幸

KDDI Web Communications

CPI

阿部 正幸（あべ まさゆき）

KDDIウェブコミュニケーションズ エバンジェリスト

ACE01/SmartRelease プロジェクトマネージャー

CPIスタッフブログ編集長

サイト制作

Flexboxで、レスポンシブWeb用のフレームを作成する

製品情報

テストサイトに構築したWordPressを公開サイトにリリースする


mochiya
ホーム
システム開発
Search
Search
mochiya
物が売れる、集客できるウェブサイトを安価に提供し社会貢献します
サイト内検索

日本酒大大好き

日本酒歴 2年以上

目次

・オブジェクトについて
・オブジェクトを使いプログラミング
・ECMAScript6
・DOMについて

JavaScriptについて

JavaScriptを学ぶには

- JavaScriptの記述場所
- 変数
- 文字列操作
- 数字演算
- 条件文
- ループ
- 関数
- オブジェクト
- DOM

JavaScriptに苦手意識がある方の共通点

{ Object }

が理解できていない

今回のセッションでは

{ Object }

に、重点を置いて今とこれからについて
解説します

オブジェクト以外は自習

http://bit.ly/js-hands-on

Objectについて

オブジェクトとは

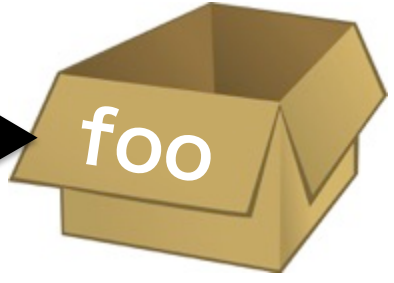
オブジェクトとは**変数**と**関数**の集合体

{ Object

sum(A1:A10)

ExcelでA1からA10までの
足し算をする関数

文字
数字
論理値



}

DEMO

http://bit.ly/lp54zipfile

(sample.html)

オブジェクトのメリット

- 可読性の向上
- 品質の向上
- 開発工数の削減
- 保守の向上

大規模になれば
なるほど必要になる

開発運用工数を下げ、他のライブラリとの衝突を避ける
世に配られているライブラリが読めるようになる
他の言語も読めるようになる

オブジェクトのメリット

そしてなによりも

書いてて気持ちイイッ

でも難しいんでしょう

慣れるって

```
var
    version = "3.2.1",

    // Define a local copy of jQuery
    jQuery = function( selector, context ) {

        // The jQuery object is actually just the init cons
        // Need init if jQuery is called (just allow error
        return new jQuery.fn.init( selector, context );
    },
```

```
jQuery.fn = jQuery.prototype = {

    // The current version of jQuery being used
    jquery: version,

    constructor: jQuery,

    // The default length of a jQuery object is 0
    length: 0,

    toArray: function() {
        return slice.call( this );
    },
```

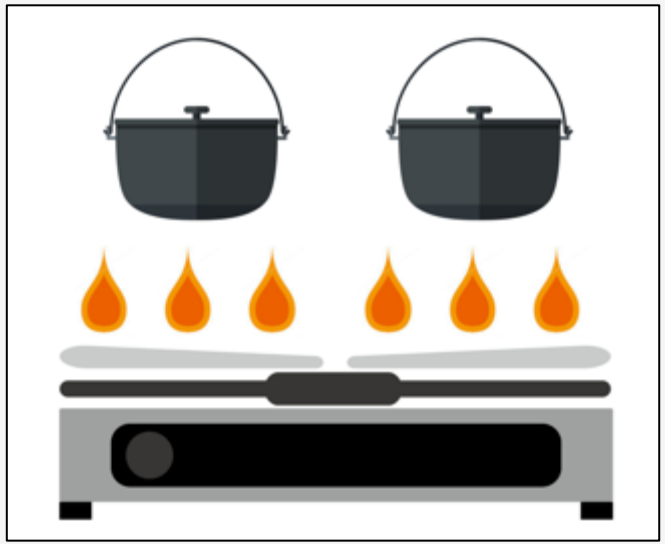
jQueryも本日紹介する書き方

Objectを使い
プログラミング

DEMO

http://bit.ly/lp54zipfile

（ index.html ）



オブジェクト家の日常

しぃちゃん、新しく買ったコンロを
JavaScriptから起動したいから、
オブジェクトに登録しておいてね!!

パパちゃま

しぃちゃん

はーい、オッケー。

オブジェクト家の日常

パパちゃま

コンロの要件は火が付けられて、
鍋が乗せられるようにすればいい？

コンロは2口ある最新型だから、2つ
火が付けられるようにしておいてね !!

しぃちゃん

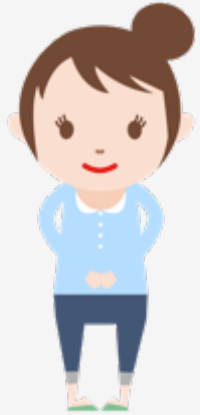
[オブジェクト要件]

- 火をつける / 消す
- 鍋をのせる / 外す
- 火がついているか、鍋が乗っているか
  確認できる

プログラミングしてみる

グローバルネームスペース

世の中には沢山のコンロがあるわ。
違うコンロから処理が上書きされたり、今回使うコンロから上書きしたりしないようにユニークなネームスペースを付けないと。



しぃちゃん

```
// グローバルネームスペースの作成
var MYAPP = {} || MYAPP;
```

コンストラクタ

「鍋」や「火」の名前と、火がついているか
鍋が乗っているか、コンストラクタに登録
しましょ。

しぃちゃん

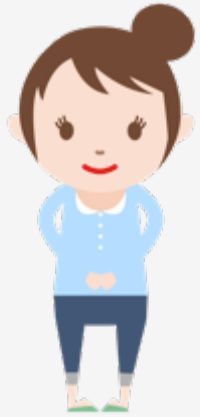
nabe1

fire1

konro1

コンストラクタ

nabeNameに鍋の名前、fireNameに火の名前を登録する箱を設定

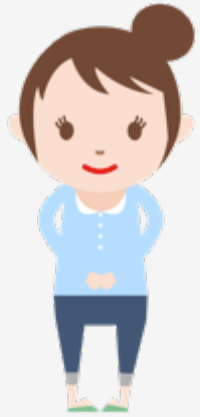


しぃちゃん

```
// コンストラクタの登録
MYAPP.konro = function(val1,val2){
    this.nabeName = val1;
    this.fireName = val2;
    this.fire = 'stop';
    this.nabe = 'off';
}
```

インスタンス起動

名前を入れる箱ができたから名前を登録しつつ
オブジェクトのインスタンスを立ち上げるわ



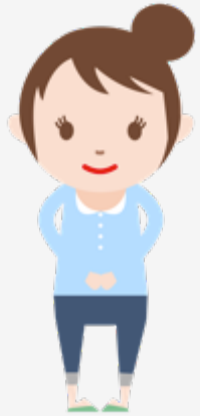
しぃちゃん

```
var konro1 = new MYAPP.konro('nabe1','fire1');
konsole.dir(konro1);
```

```
▼MYAPP.cooking {nabeName: "nabe1",
    fire: "stop"
    fireName: "fire1"
    nabe: "off"
    nabeName: "nabe1"
  ▶ __proto__: Object
```

メソッド追加

konro1のオブジェクトが立ち上がったので、
あとは「火」をつけて、「鍋」を乗せる

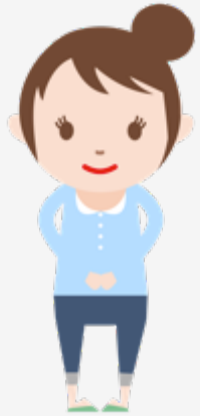


しぃちゃん

```
// 火をつけるメソッド
MYAPP.konro.prototype.addFire = function(){
    // display: none を initialに変更し、火をつける
    var fire = document.getElementById(this.fireName);
    fire.children[0].style.display = 'initial';
    // プロパティの変更
    this.fire = 'fire'
    return this;
}
```

メソッド追加

```
// 鍋を乗せるメソッド
MYAPP.konro.prototype.addNabe = function(){
    // display: none を initialに変更し、鍋を乗せる
    var nabe = document.getElementById(this.nabeName);
    nabe.style.display = 'initial';
    // プロパティの変更
    this.nabe = 'on';
    return this;
}
```

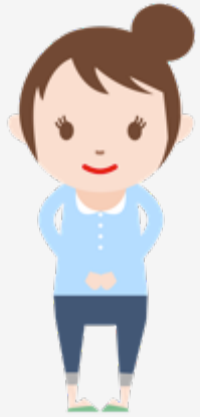


しぃちゃん

MYAPP.konroのオブジェクトはfunctionを使い作成しました。
functionを使い作成すると「prototype」が、自動で生成されるので、そこにメソッドを追加する。

メソッドの実行

メソッドの準備ができたので、火を付けて、
鍋を乗せましょ



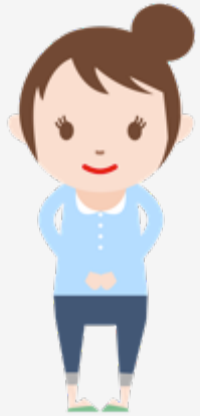
しぃちゃん

```
// オブジェクトを立ち上げる
var konro1 = new MYAPP.konro('nabe1','fire1');
// 火をつける
konro1.addFire();
// 鍋を乗せる
konro1.addNabe();
```

オブジェクトぽい感じになってきたわ

メソッド追加

現在火が付いているか、鍋が乗っているかを
確認するメソッドを追加するわ

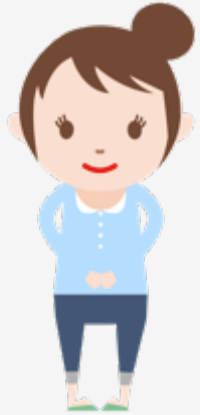


しぃちゃん

```
// コンロの状態を表示する
MYAPP.konro.prototype.showKonro = function(){
    console.log(
        'コンロの状態：（火）' + this.fire +
        '、（鍋）' + this.nabe
    );
}
```

メソッドの実行

```
// コンロの状態を確認
konro1.showKonro();
```

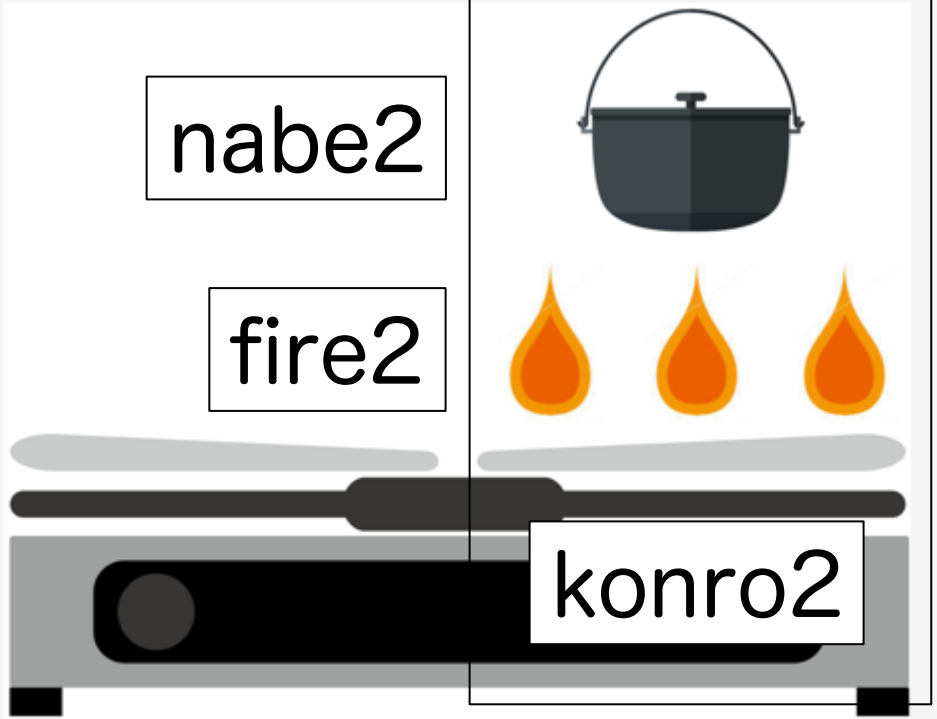


しぃちゃん

インスタンス起動

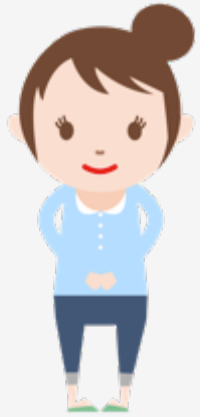
コンロは2口あるから、もう1つオブジェクトを
立ちあげましょ。

しぃちゃん



インスタンス起動

```
// オブジェクトを立ち上げる
var konro2 = new MYAPP.konro('nabe1','fire1');
// 火をつける
konro2.addFire();
// 鍋を乗せる
konro2.addNabe();
```

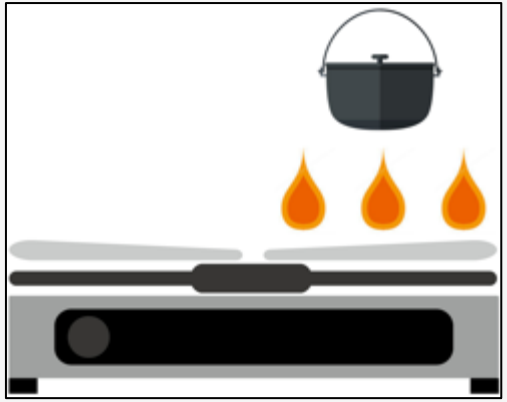


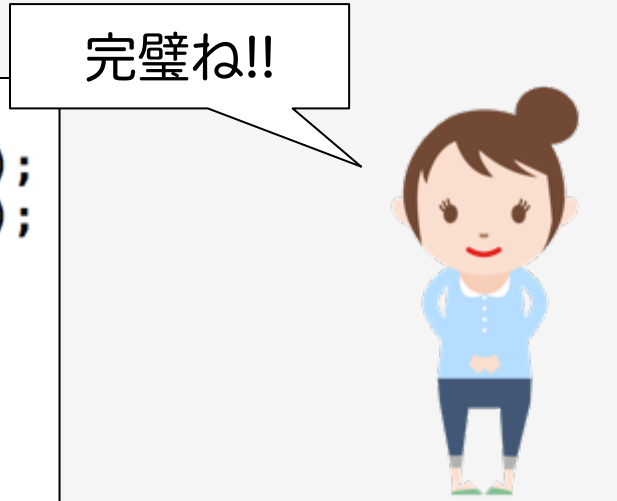
しぃちゃん

コンストラクタや、メソッドはすでに登録
しているから、new演算子を使い、
インスタンスを起動するだけで、オブジェクトが
利用できて、楽ね!!

使用例

```
// オブジェクトを立ち上げる
var konro1 = new MYAPP.konro('nabe1','fire1');
var konro2 = new MYAPP.konro('nabe1','fire1');
// 火をつける
konro1.addFire().addNabe();
// 状態確認
konro1.showKonro();
konro2.showKonro();
```





しぃちゃん

```
var
    version = "3.2.1",

    // Define a local copy of jQuery
    jQuery = function( selector, context ) {

        // The jQuery object is actually just the init cons
        // Need init if jQuery is called (just allow error
        return new jQuery.fn.init( selector, context );
    },
```

```
jQuery.fn = jQuery.prototype = {

    // The current version of jQuery being used
    jquery: version,

    constructor: jQuery,

    // The default length of a jQuery object is 0
    length: 0,

    toArray: function() {
        return slice.call( this );
    },
```

jQueryも
同じ書き方

ECMAScript6

ECMAScript 5 6 2016+ next intl non-standard compatibility table

Flattr | by kangax | Gratipay | & webbedspace & zloirock

Sort by Engine types | Show obsolete platforms | Show unstable platforms

V8 | SpiderMonkey | JavaScriptCore | Chakra | Carakan | KJS

Minor difference (1 point) | Small feature (2 points) | Medium feature (4 points) | Large feature (8 points)

Feature name	Current browser	Compilers/polyfills: Traceur	Babel + core-js[2]	Closure	TypeScript + core-js	es6-shim	Kong 4.14[3]	IE 11	Edge 15	Edge 16	Edge 17 Preview	FF 52 ESR	FF 56	FF 57 Beta
	97%	56%	71%	48%	59%	17%	5%	11%	96%	96%	96%	94%	97%	97%
Optimisation														
proper tail calls (tail call optimisation)	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Syntax														
default function parameters	7/7	4/7	4/7	5/7	5/7	0/7	0/7	0/7	7/7	7/7	7/7	6/7	7/7	7/7
rest parameters	5/5	4/5	3/5	2/5	4/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5
spread (...) operator	15/15	15/15	13/15	12/15	4/15	0/15	0/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15
object literal extensions	6/6	6/6	6/6	4/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6
for..of loops	9/9	9/9	9/9	6/9	3/9	0/9	0/9	0/9	9/9	9/9	9/9	7/9	9/9	9/9
octal and binary literals	4/4	2/4	4/4	4/4	4/4	2/4	0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4
template literals	5/5	4/5	4/5	3/5	3/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5
RegExp "y" and "u" flags	5/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5
destructuring, declarations	22/22	20/22	21/22	20/22	15/22	0/22	0/22	0/22	22/22	22/22	22/22	21/22	22/22	22/22
destructuring, assignment	24/24	23/24	24/24	21/24	19/24	0/24	0/24	0/24	24/24	24/24	24/24	23/24	24/24	24/24
destructuring, parameters	24/24	19/24	21/24	18/24	16/24	0/24	0/24	0/24	23/24	23/24	23/24	21/24	24/24	24/24
Unicode code point escapes	2/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2	2/2	2/2	2/2	1/2	2/2	2/2
new.target	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2
Bindings														

Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Put in next-gen JavaScript	Get browser-compatible JavaScript out
`const x = [1, 2, 3];` `foo([...x]);`	`var x = [1, 2, 3];` `foo([].concat(x));`

Check out our REPL to experiment more with Babel!

Latest From Our Blog: Babel Turns Three

```es5
var testObject = function(val){
    this.name = val;
}
testObject.prototype.showConsole = function(val){
    console.log(val + this.name);
}
var objTest = new testObject('テストだよ');
objTest.showConsole('オブジェクトの');
```

ES5

```es6
class testClass{
    constructor(hoge){
        this.name = hoge;
    }
    showConsole(val){
        console.log(val + this.name);
    }
}
var testClassIn = new testClass('テストだよ');
testClassIn.showConsole('クラスの');
```

ES6

DOMについて

DOMとは

DOM（Document Object Model）とは
ブラウザそのものがオブジェクト化したもの。

「window.document」をコンソール画面に表示

```
console.dir(window.document);
```

```
▼#document
    URL: "file:///Users/abechiyo/hands-on/lp54/sample.html"
  ▶ activeElement: body
    alinkColor: ""
  ▶ all: (7) [html, head, title, meta, body, h1, script]
  ▶ anchors: []
  ▶ applets: []
    baseURI: "file:///Users/abechiyo/hands-on/lp54/sample.html"
    bgColor: ""
  ▶ body: body
    characterSet: "UTF-8"
    charset: "UTF-8"
    childElementCount: 1
  ▶ childNodes: (2) [<!DOCTYPE html>, html]
```

DOMとは

documentオブジェクトを読み込んだり、書き込んだりしてサイトを動的に見せている。

「window.document」をコンソール画面に表示

```
console.dir(window.document);
```

DEMO

http://bit.ly/lp54zipfile

(sample.html)

まとめ

・オブジェクトとは変数、関数の集合体
・オブジェクト指向型メリット
  - 誰でも使える
  - 拡張性の向上 / 運用が楽
  - 可読性の向上
・会社で使うアニメーションなどの動作を
  オブジェクトにまとめておくと便利

ありがとうございました

LINE
CPIスタッフブログを
ともだち追加

cpi-line

阿部 正幸
ID：chiyo.abe

KDDI Web Communications

CPI®