LPI-Japan主催

HTML5プロフェッショナル認定試験 レベル2　ポイント解説無料セミナー

2014年11月15日

アシアル株式会社　生形可奈子

アシアルについて

Asial
Engineering the future

事業内容
HTML5アプリ開発環境（Monaca）、ネイティブアプリ開発、サーバーサイド開発、インフラ、教育事業など

自己紹介

生形　可奈子（うぶかた・かなこ）

- 講師・エバンジェリスト
- 元エンジニア
- 著書：「スラスラわかるJavaScript」（翔泳社）

目次

- HTML5プロフェッショナル認定試験とは
  - 概要
  - 試験範囲
- 頻出ポイント解説
  - JavaScriptのデータ型
  - オブジェクト
  - 関数とスコープ
  - prototype

HTML5プロフェッショナル認定試験

HTML5プロフェッショナル認定試験とは

- 特定非営利活動法人LPI-Japanが実施する、HTML5および周辺技術の知識レベルを測る認定制度です。

- 試験の難易度を示す2種類のレベルがあり、段階的に受験します。
  - Level1

    マルチデバイスに対応した静的なWebコンテンツを HTML5を使ってデザイン、作成できるレベル

  - Level2

    システム間連携や最新のマルチメディア術に対応したWebアプリケーションや動的Webコンテンツの開発・設計ができるレベル

Level2の出題範囲（1/5）

出題範囲	重要度
JavaScript	
JavaScript文法	★★★★★★★★★★ 10
WebブラウザにおけるJavaScript API	
イベント	★★★★★★★★ 8
ドキュメントオブジェクト／DOM	★★★★★★ 6
ウィンドウオブジェクト	★★★★★★★★ 8
Selectors API	★★★★ 4
テスト・デバッグ	★★ 2

Level2の出題範囲 (2/5)

出題範囲	重要度
グラフィックス	
Canvas(2D)	★★★★★★ 6
SVG	★ 1
マルチメディア	
video要素,audio要素	★★ 2
オフラインアプリケーションAPI	
アプリケーションキャッシュの制御	★★ 2
Session History and Navigation	
History　API	★★★ 3

Level2の出題範囲（3/5）

出題範囲	重要度
表示制御	
Page Visibility	★★ 2
Timing control for script-based animations	★★ 2
ストレージ	
Web Storage	★★★★ 4
Indexed Database API	★★ 2
File API	★★ 2

Level2の出題範囲　(4/5)

出題範囲	重要度
通信	
WebSocket	★★ 2
XMLHttpRequest	★★★★ 4
Geolocation API	
Geolocation APIの基本と位置情報の取得	★★ 2
Web Workers	
並列処理の記述	★★★★ 4

Level2の出題範囲　(5/5)

出題範囲	重要度
パフォーマンス	
Navigation Timing	★★★★ 4
High Resolution Time	★ 1

受験について

- **試験方式はコンピュータベーストテスト（CBT）です。試験配信会社の「ピアソンVUE」を通して受験します。**

問題数	40～45問
試験時間	90分
合格ライン	約7割
回答方式	殆どが選択式（複数回答あり） 記述式も１問程度 コードリーディング問題が多い
受験料	¥15,000（税抜）

JavaScriptのデータ型

JavaScriptのデータ型

- 代表的なデータ型

型の分類	代表的な型
プリミティブ型 （１つの値のみを持つ）	数値、文字列、論理値
オブジェクト型 （プロパティの集合体）	オブジェクト、関数、配列など
特殊な型	null（値が存在しない） undefined（値が定義されていない）

問題：プリミティブ型に対してプロパティを指定

- 以下の実行結果は？

```
console.log("hoge".length);
```

- 4
- null
- undefined
- エラー

ラッパーオブジェクト

- ラッパーオブジェクト … プリミティブ型に対応するオブジェクト

プリミティブ型	ラッパーオブジェクト
数値	Number
文字列	String
論理値	Boolean

```
"hoge".length
```

上記のように記述すると、プリミティブ型の文字列がラッパーオブジェクトであるString型に暗黙的に変換される

問題：プリミティブ型とラッパーオブジェクトの比較

- 以下の実行結果は？（true / false）

```
var a = "hoge";
var b = new String("hoge");
console.log(a == b);
```

- 以下の実行結果は？（true / false）

```
var a = "hoge";
var b = new String("hoge");
console.log(a === b);
```

問題：値の比較、参照の比較

- 以下の実行結果は？（true / false）

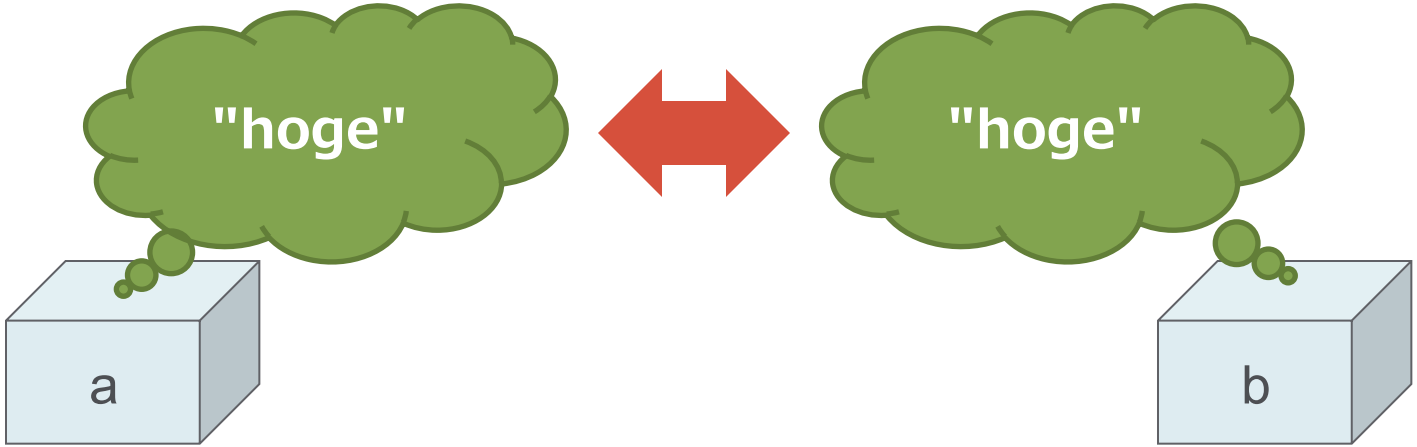
```
var a = "hoge";
var b = "hoge";
console.log(a == b);
```

- 以下の実行結果は？（true / false）

```
var a = new String("hoge");
var b = new String("hoge");
console.log(a == b);
```

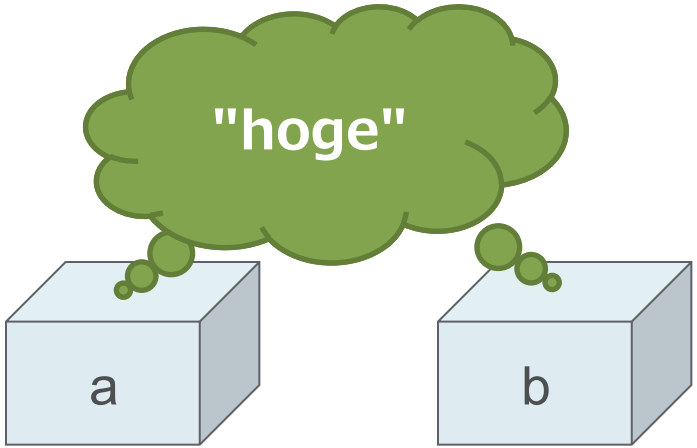
値の比較、参照の比較

- プリミティブ型は値そのものを比較する
- オブジェクト型は参照しているオブジェクトが同じものかどうかを比較する



同じオブジェクトを参照している例

```
var a = new String("hoge");
var b = a;
console.log(a == b);
```



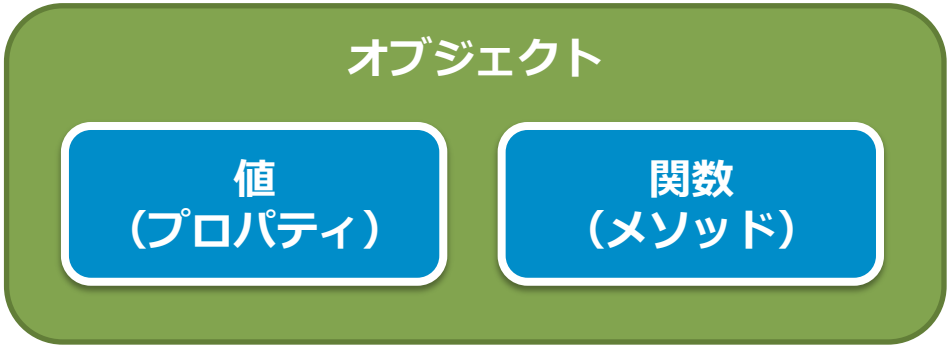
この場合は同じオブジェクトを
参照しているのでtrue

オブジェクト

プロパティとメソッド

プロパティとメソッド

- オブジェクトの要素には、数値や文字列などの値だけではなく、関数も代入することができる
- 値のことを「プロパティ」、関数のことを「メソッド」という



問題：プロパティの削除

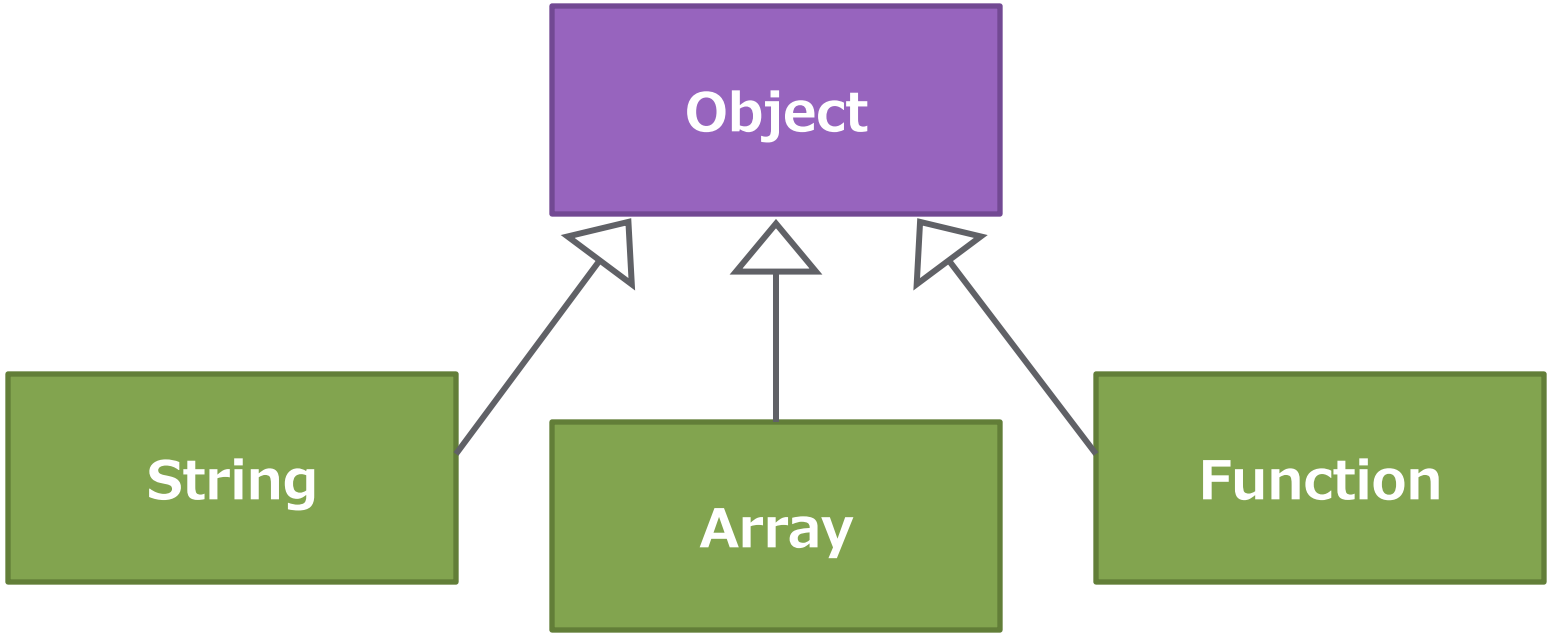
- 以下の実行結果は？

```
var obj = {
        name: "hoge"
};
delete obj.name;
console.log(obj.name);
```

- null
- undefined
- 空文字
- エラー

オブジェクトの種類

- Objectオブジェクトはすべてのオブジェクトのベースとなっている



問題：配列のインデックス

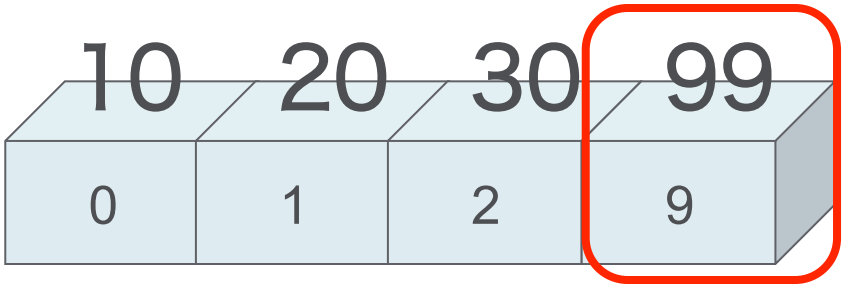
- 以下の実行結果は？

```
var array = [10, 20, 30];
array[9] = 99;
console.log(array.length);
```

- 3
- 4
- 10
- undefined

配列のインデックス

- インデックスは連番でなくても割り振ることができる



- lengthプロパティは最大インデックス+1を返す

this

- thisとは、自分自身のオブジェクトを指す

```
var obj1 = {
        name: "hoge",
        func: function() {
                console.log(this.name);   // hoge
        }
};
obj1.func();
```

call/applyメソッド

- call/applyによって、thisが指すオブジェクトを変更できる

```
var obj1 = {
        name: "hoge",
        func: function() {
                console.log(this.name);   // fuga
        }
};
var obj2 = {
        name: "fuga"
};
obj1.func.call(obj2);   //obj1.func()を、obj2をthisとして実行
```

問題：オブジェクトの外側に記述されたthisは？

- 以下のthisは何を指す？

```
<script>
    console.log(this);
</script>
```

- null
- undefined
- Functionオブジェクト
- Windowオブジェクト

関数とスコープ

関数の定義いろいろ①

- function命令文

```
function func(message) {
        alert(message);
}
func("hello");
```

  - 関数にfuncという名前をつけている

関数の定義いろいろ②

関数リテラル

```
var func = function(message) {
        alert(message);
}
func("hello");
```

- funcという変数に関数を代入している
- この場合、関数名は省略できる（無名関数や匿名関数と呼ばれる）

関数の定義いろいろ③

即時関数

```
(function(message) {
        alert(message);
})("hello");
```

- (関数の定義)(引数); とすることで定義と呼び出しを同時に行う
- 主にスコープを限定する用途などで利用される

問題：JavaScriptのスコープ①

- 以下の実行結果は？

```
for(var i = 0; i < 3; i++) {
    var a = "hoge";
}
console.log(a);
```

- エラー
- undefined
- null
- "hoge"

JavaScriptのスコープ①

- JavaScriptにはブロックスコープがない

```
var a = 10;  // aはグローバル変数なのでどこからでも参照可
function func1() {
    var b = 20;
    function func2() {
        var c = 30;
    }
}
```

cのスコープ

bのスコープ

- スコープが作られるのは関数内のみ
- スコープの範囲外で変数を参照した場合、ReferenceErrorが発生する

問題：JavaScriptのスコープ②

- 以下の実行結果は？

```
var a = "global";
function func() {
    console.log(a);
    var a = "func";
}
func();
```

- エラー
- undefined
- "global"
- "func"

JavaScriptのスコープ②

- ホイスティング（変数の巻き上げ）とは
  - 関数の途中で宣言された変数は、関数の先頭に巻き上げて宣言される
  - ただし、初期値として代入されている値は巻き上げない
  - 先ほどのコードは以下のように解釈される

```
var a = "global";
function func() {
    var a;
    console.log(a);
    a = "func";
}
func();
```

ガベージコレクション

- ガベージコレクションとは、もう参照されなくなったメモリ領域を自動的に解放する機能のこと

```
var func = function() {
    var a = 0;
    console.log(++a);
};

func();    // 1
func();    // 1
```

ガベージコレクション

- ガベージコレクションとは、もう参照されなくなったメモリ領域を自動的に解放する機能のこと

```
var func = function() {
    var a = 0;
    console.log(++a);
};

func();    // 1
func();    // 1
```

aのスコープ

ガベージコレクション

- ガベージコレクションとは、もう参照されなくなったメモリ領域を自動的に解放する機能のこと

```
var func = function() {
    var a = 0;
    console.log(++a);
};

func();     // 新しくaが作られ、消去される
func();     // 新しくaが作られ、消去される
```

関数が最後まで実行し終わったら、
関数内のローカル変数は不要になって
消去される

クロージャ

- クロージャは、変数を参照し続けることで、状態を保持する仕組み

```
var func = (function() {
        var a = 0;
        return function() {
                console.log(++a);
        };
})();

func();    // 1
func();    // 2
```

クロージャ

- クロージャは、変数を参照し続けることで、状態を保持する仕組み

```
var func = (function() {
        var a = 0;
        return function() {
                console.log(++a);
        };
})();

func();    // 1
func();    // 2
```

aのスコープ（外側の関数）

クロージャ

- クロージャは、変数を参照し続けることで、状態を保持する仕組み

```
var func = (function() {
        var a = 0;
        return function() {
                console.log(++a);
        };
})();

func();     // 1
func();     // 2
```

① 即時関数になっている
外側の関数が実行される

② 内側の関数が
戻り値として
返却される

クロージャ

- クロージャは、変数を参照し続けることで、状態を保持する仕組み

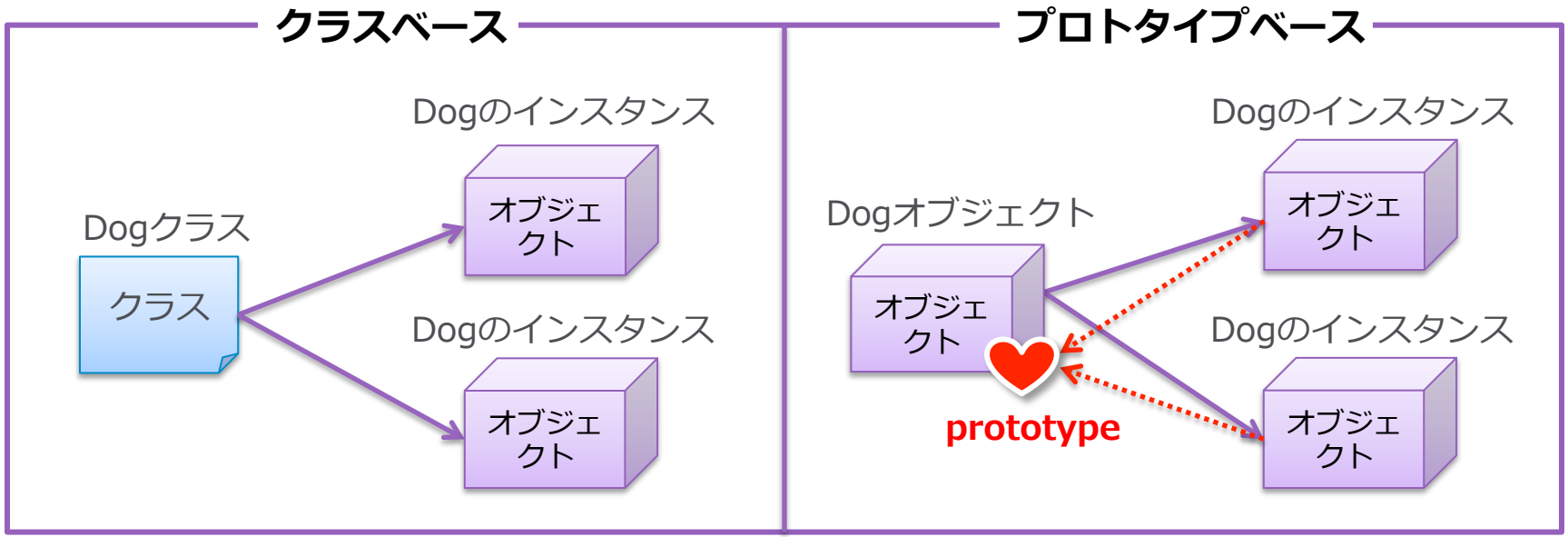
```
var func = function() {
    console.log(++a);
};



func();    // 1
func();    // 2
```

③ 外側の関数は終了したが、
aはまだ参照されているので
消去されない！

prototype

プロトタイプベースオブジェクト指向

- JavaScriptはプロトタイプベースのオブジェクト指向言語
  - 別のオブジェクトを基礎として、新しいオブジェクトを作成する



プロトタイプとなるオブジェクトの作り方

- プロトタイプとなるオブジェクト＝関数オブジェクト
  - 関数内の処理はインスタンス化した時にコンストラクタとして実行される

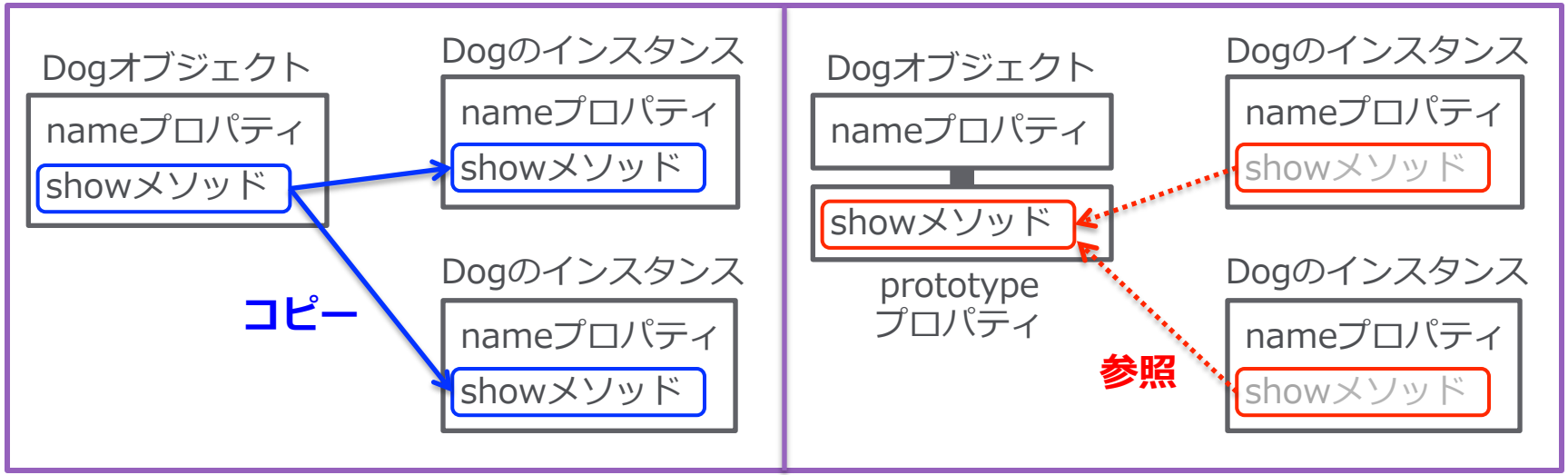
```
var Dog = function(_name) {
    this.name = _name;
    this.show = function() {
        alert(this.name);
    };
};
```

  - 上記のオブジェクトをプロトタイプとしてインスタンスを生成

```
var dog1 = new Dog("ポチ");
```

メソッドの共通化

- オブジェクトをインスタンス化すると、元のオブジェクトで定義されたプロパティとメソッドが、各インスタンスにコピーされる
- 関数オブジェクトが持つprototypeプロパティに定義したプロパティとメソッドは、コピーされずに各インスタンスから参照される



prototypeプロパティ

- 各インスタンスで共通のメソッドはprototypeプロパティに登録する

```
var Dog = function(_name) {
    this.name = _name;
    this.show = function() {
        alert(this.name);
    };
};

Dog.prototype.show = function() {
        alert(this.name);
};
```

受験対策

参考書籍

コア・JavaScript

JavaScript 第6版 	JavaScript本格入門　～モダンスタイルによる基礎からAjax・ｊQueryまで 	フロントエンドエンジニア養成読本 [HTML、CSS、JavaScriptの基本から現場で役立つ技術まで満載!] (Software Design plus) 
独習JavaScript 第2版 	開眼! JavaScript ―言語仕様から学ぶJavaScriptの本質 	

HTML5 API

HTML5 Web標準API バイブル

LPI-JAPAN HTML5 Professional Certification

Open the Future with HTML5.