

～新宿伊勢丹-家電売り場で日立の白くまくん（8畳用）を購入～

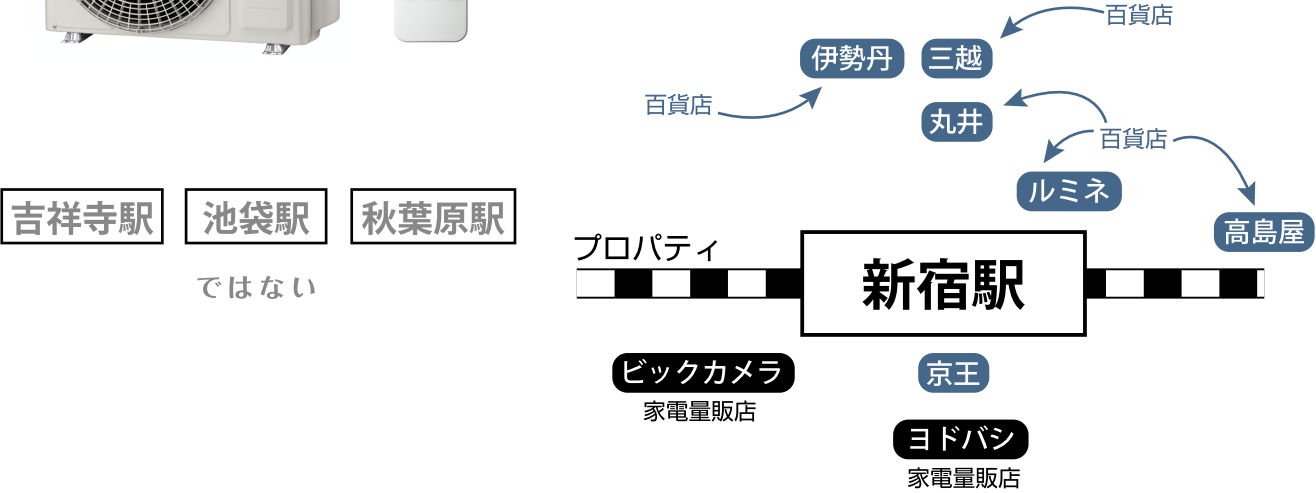
要件（必要条件）を整理してまとめておく

要件 (自分の目的)	購入	リサーチ / 下見	修理 / リフォーム	委託 / リース
	店頭 / 通販	調査 / 見積もり	交換 / 工事	メリット / デメリット

購入したい商品 日立の白くまくん (8畳用)
(ダイキン / 三菱 / パナソニックではない)



購入する百貨店 伊勢丹 / 家電売り場 / 空調季節家電コーナー
(家電量販店ではない) (玩具売り場ではない) (空気清浄機コーナーではない)



JavaScript には [window] というオブジェクト (最上位) が組み込まれている。
[百貨店] [家電量販店]

JavaScript には [document] というオブジェクトが組み込まれている。
[伊勢丹] [三越] [丸井]

[メーカー] / [商品名] / [会計] というプロパティも組み込まれている。
[見積もり] [修理]

このように, JavaScript に最初から入っている「明確な分類」には **組み込み** という言葉を付けます。

組み込まれていない「名前や機能」を作成したら **自作** という言葉を付けます。

組み込みオブジェクト
組み込みプロパティ + バリュー
組み込みメソッド

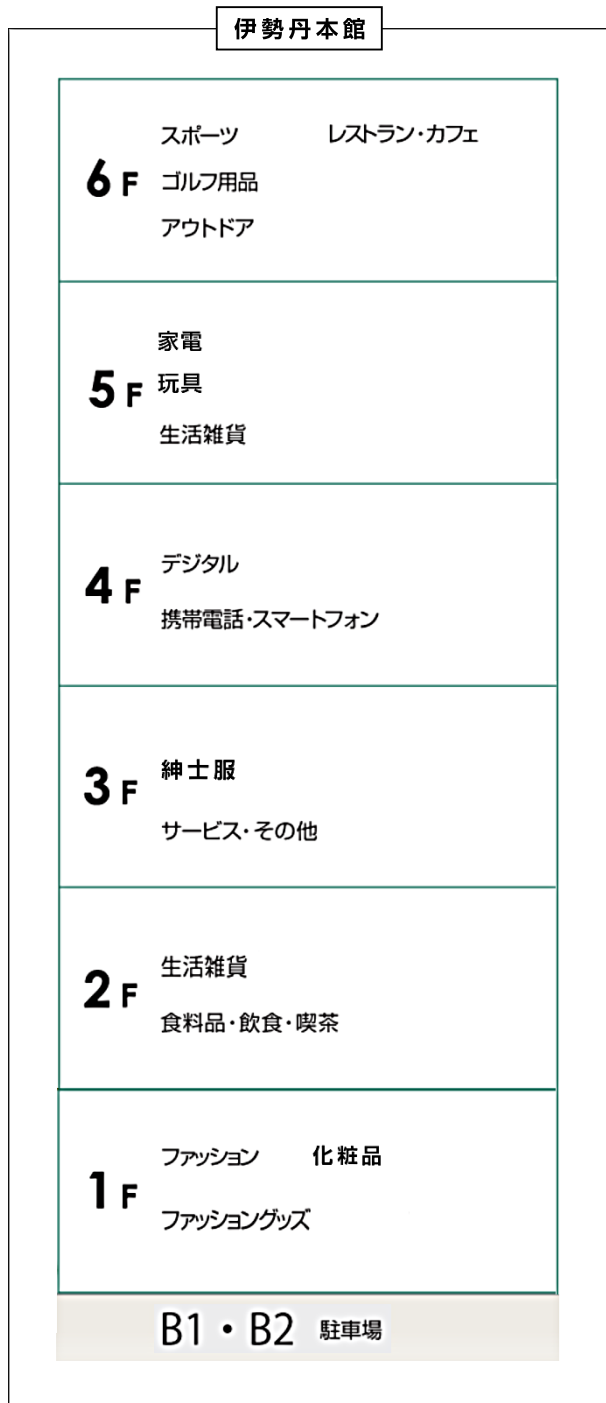
自作オブジェクト
自作プロパティ + バリュー
自作メソッド

分類クイズ：それぞれの用語で要件を分類するために組み込まれている定義は何ですか？

組み込みオブジェクト / 組み込みプロパティ + バリュー / クラス / コンストラクタ / 組み込みメソッドとして

百貨店	ヨドバシ
冬のセール開催中! 張り紙	新宿駅
うるるとさらら	伊勢丹
倉庫から商品を探し出す	要件を聞く人
家電量販店	営業時間のお知らせ：10:00～20:00
ダイキン	カタログ制作
売り場	見積みもり
今月のオススメ商品一覧	配達日の電話連絡
家電売り場	注意書き張り紙
メリットの説明	霧ヶ峰
日立	吉祥寺駅
配送業者への手配	購入手続き
玩具売り場	レジでお会計 / 領収書の発行
白くまくん	故障部品の修理
購入商品	価格表
フロア案内板	交換作業

百貨店ではあらゆる商品が
分かりやすく分類されている



大きな分類（家電）
細かい分類季節（家電）
さらに細かい分類（メーカー）

ECMAScript の組み込みオブジェクトは
重複がなく、すべてが明確に整理されている

`document`

→ 伊勢丹本館全体

`document.body`

→ 伊勢丹本館全体の 5 階フロア全部

`document.body.textContent`

→ 伊勢丹本館全体の本館 5 階フロア全部の
説明書や価格表

`document.title`

→ 伊勢丹本館全体のすべての項目

`document.images`

→ 伊勢丹本館全体に並んでいる商品のカタログ
(= img タグ)

`document.forms`

→ 伊勢丹本館全体のサービス・カウンター /
レジ / 受付 (送信できるフォーム)

`document.getElementById()`

→ 伊勢丹本館全体の店員さんによるサービス
(倉庫から商品を探し出す /
お会計 / 包装 / レシート発行など)

JavaScript の学習が難しく感じる最大の理由



「伊勢丹の全フロアの構造と設備 / 商品とサービス内容を完全に知っている」
これを前提として説明されるから

私たちは百貨店に地上や地下の入り口から入り、
1 階 → 2 階 → 3 階と 順番に歩いて店内のことを確認する。
歩きながら見ているうちに「この階は婦人服 / この階は家電 / この階はスポーツ用品」と、
売り場のことを徐々に理解する。
最初から 6 階建て百貨店のすべてを把握している人はいない。

なので、JavaScript の最初の勉強は
「伊勢丹で日立のエアコン（白くまくん）を購入すること」をイメージして
script を一つ書いてみることに。
いろいろな要件が分類されている仕組みが分かれば OK。

重要な用語

Object(オブジェクト) / **Property**(プロパティ) と **Value**(バリュー / **Method**(メソッド))

Class(クラス) と **Constructor**(コンストラクター)

▶ JS=Built-in_Object.html
JavaScript 組み込みオブジェクト一覧

「伊勢丹」や「三越」などの 1 つの具体的なお店 **組み込みオブジェクト**

document

各階の「各階フロア全体」は **組み込みプロパティ**

body

「商品名」「価格」などはプロパティの値

value

変数名「えあこん」を宣言することは **自作オブジェクト** を作っている

const えあこん

「店員の業務」や「店員業務さんのサービス」、「店員の行動」などはメソッド

具体的な用語の分類は クラス / コンストラクタ に相当する

Legacy JavaScript と Modern JavaScript の比較

Legacy JavaScript

```
<html lang="ja"translate="no">
<meta charset="utf-8">
<body>
<script>
const えあこん ={ めーかー : " 日立 ", 商品名 : " 白くまくん ", 大きさ : " 八畳用 " };
document.body.textContent=えあこん . めーかー + " の "+ えあこん . 商品名 + " で "+ えあこん . 大きさ + " を購入します .";
</script>
<hr>
<p> 日立の白くまくんで八畳用を購入します.</p>
</body>
</html>
```

結果 日立の白くまくんで八畳用を購入します .

日立の白くまくんで八畳用を購入します .

Modern JavaScript

```
<html lang="ja"translate="no">
<meta charset="utf-8">
<body>
<script>
const えあこん ={ めーかー : " 日立 ", 商品名 : " 白くまくん ", 大きさ : " 八畳用 " };
document.body.textContent=`${ えあこん . めーかー } の ${ えあこん . 商品名 } で ${ えあこん . 大きさ } を購入します .`;
</script>
<hr>
<p> 日立の白くまくんで八畳用を購入します.</p>
</body>
</html>
```

結果 日立の白くまくんで八畳用を購入します .

日立の白くまくんで八畳用を購入します .

Legacy JavaScript

```
<body>
<script>

const えあこん={ めーかー:"日立",商品名:"白くまくん",大きさ:"八畳用"};

document.body.textContent=えあこん.めーかー+"の"+えあこん.商品名+"で"+えあこん.大きさ+"を購入します.";

</script>
</body>
```

結果 日立の白くまくんで八畳用を購入します。

Modern JavaScript

```
<body>
<script>

const えあこん={ めーかー:"日立",商品名:"白くまくん",大きさ:"八畳用"};

document.body.textContent=`${えあこん.めーかー}の${えあこん.商品名}で${えあこん.大きさ}を購入します.`;

</script>
</body>
```

結果 日立の白くまくんで八畳用を購入します。

あらかじめ JavaScript が提供しているオブジェクト / プロパティ + バリュー

Built-in Object

JavaScript のオブジェクトとは
「複数のプロパティ (名前と値の組み合わせ)」をまとめた **データ構造** のこと
集合体
入れ物

組み込みオブジェクトは通常、組み込みプロパティを参照する

Built-in property と value 組み込みプロパティとバリューは JavaScript に最初から組み込まれている
オブジェクトにくっ付いている **特製の名前と値** のこと
(組み込みオブジェクトが持っている組み込みプロパティとバリュー)

自作オブジェクト / プロパティ + バリュー

User-defined Object User-defined property と value

自作プロパティはキーとも言う

メーカー: " 日立 " 商品名: " 白くまくん " 大きさ: " 八畳 "

document.body.textContent と 3 つの組み込みプロパティが繋がると
「表示せよという」命令文になる (メソッドと同じ)

{ → opening brace (開き波かっこ)

}

} → closing brace (閉じ波かっこ)

{ }

オブジェクト・リテラル (中身 → プロパティ名と値のペア)

JavaScript が最初から用意している関数 (オブジェクトや配列などに標準で付いているもの)

Built-in Method

自作メソッドとは自分でオブジェクトの中に作った関数 のこと

User-defined Method

オブジェクト・リテラル object literal

解説

const でラム・メモリの中に箱（領域）を確保しながら、
中身である「めーかー:"日立", 日立商品名:"白くまくん", 大きさ:"八畳用"」を [フワフワした可変データ] として生成し、
箱に「えあこん」という変数名を付けて宣言した。これが自作オブジェクトです。
つまり自作オブジェクトとは、複数の自作プロパティをひとまとめにして扱えるように変形したもの（可変データ）を
入れた箱のような存在。
* 注意：「えあこん」は箱の名前であって可変データの名前ではない。
次の行 (document から始まる部分) で、この html ファイルに書かれている <body> タグの中に
「オブジェクトの情報（文字列）を表示せよ」という命令を与える。
ただし「... の / ... で / ... を購入します。」を間に入れながら、

記述されている document が JavaScript の「組み込みオブジェクト」である。
body と textContent は document に付いている「組み込みプロパティ」。
* document → 「この HTML 文書全体」という組み込みオブジェクト (window.document と同じ意味)

組み込みプロパティ の body → <body> タグの中身を指している（つまりページ内の <body> というエレメントを
オブジェクトとして取得している。
組み込みプロパティ の textContent → 「表示するための文字列」を確保している。

document . body . textContent は HTML 本文に文字を表示するために組み合わせてある。
言葉で言うと「body エレメントの中に指定したテキストを書き込みなさい」という命令文。
\${ } は変数を文字列に埋め込む文法記号。

記述の読み上げ

Modern JavaScript

ばく バック・クオート
漠 → `

ボディ スクリプト からスクリプト ボディ
<body><script>

コンスト
const えあこん は すなわち が "日立" で が "白くまくん" で が "八畳用" ナリ ;

document の body の textContent は 漠 (バック・クオート)

ドルすなわち ナリ ドルすなわち ナリ ドルすなわち ナリ ばく 漠
\$ { えあこん . めーかー } 文字 \$ { えあこん . 商品名 } 文字 \$ { えあこん . 大きさ } 文字 ;

</script></body>

まとめ

1. 新宿伊勢丹の家電売り場にて、日立の白くまくん (8 畳用) を購入することをイメージする。

2. HTML タグで文章を書いてみる。 → `<p> 日立の白くまくんで八畳用を購入します .</p>`

結果 → 日立の白くまくんで八畳用を購入します。

3. タグで表示させた結果とまったく同じ文章を JavaScript で記述してみる。