Javascript 入門

対象

鄭祥飛@2021/06/12

オブジェクト(対象)とは

- 複数変数を組み合わせ。（対象のプロパティ）
- 対象に関数を設定できる。
- 人間に理解しやすために、現実に関連のある情報を一つにオブジェクトに組み合わせ

例(一)

「車」というオブジェクトは下記のデータを組み合わせできる

- エンジン
- 車輪・４つ
- 走る関数

例(二)

「社員給料」というオブジェクトは下記のデータを組み合わせできる

- 社員名前
- 基本給
- 役職給
- 税
- 給料総額計算関数
- 実際お支払い給料の計算関数

使いました対象

```
let myArray = new Array('a','b');
myArray.slice(1, 4)
console.log(myArray.length);
```

- Array:対象名
- slice():Array対象の関数
- length:Array対象のプロパティ
- 対象の作成： `new` というキーワードを使用する

対象の定義

```
function Car(e,w) {
    this.engine = e;
    this.wheel = w;
    this.run = function() {
        console.log(`speeds:${this.engine * this.wheel}`);
    }
}
```

- Car: 対象名
- this.engine：対象のプロパティ。 `this` ＝対象の自体
- this.wheel: 対象のプロパティ
- this.run: 対象の関数

対象を使う

```
let Honda = new Car(1.8, 4);
Honda.run();

let toyota= new Car(2, 4);
toyota.wheel = 6;
toyota.run();
```

`new` キーワード。対象のインスタンスを作成。対象の定義は、金型ようなものです。 `new` は金型で製品を作成する。

対象の定義と使い――Typescirpt

```
class Car {
  engine: number;
  wheel: number;
  constructor(engine: number, wheel: number) {
    this.engine = engine;
    this.wheel = wheel;
  }
  run() {
    console.log(`speeds:${this.engine * this.wheel}`);
  }
}

let myCar: Car = new Car(2,4);
myCar.run();
```

対象の定義と使い――JAVA

```
public class Car {
  public int engine;
  public int wheel;
  public Car(int engine, int wheel) {
    this.engine = engine;
    this.wheel = wheel;
  }
  run() {
    console.log(`speeds:${this.engine * this.wheel}`);
  }
}

Car myCar = new Car(2,4);
myCar.run();
```

対象の直接初期化

```
let car = {
    engine: 3,
    wheel: 3,
    run:function() {
        console.log(`speeds:${this.engine * this.wheel}`);
    }
};
car.run();
```

- `let arr = [1,2,3,4,5];` の使い方と同じです。
- `{}` 対象初期化の記号
- `engine`, `wheel` 対象のプロパティです。値：文字列、数字、配列、Boolean、関数、対象等など

プロパティの列挙

for...in

```
for (let i in car) {
    if (car.hasOwnProperty(i)) {
        console.log(`プロパティ${i}の値は: ${car[i]}`);
    }
}
```

Object.keys(obj)

```
let p = Object.keys(car);
for (let i of p) {
    if (car.hasOwnProperty(i)) {
        console.log(`プロパティ${i}の値は: ${car[i]}`);
    }
}
```

プロパティの使い方

```
car.wheel = 3;
car['wheel'] = 3;
```

対象の継承

```
function Sample() {
   this.a = 1;
   this.b = 2;
}
let o = new Sample();
Sample.prototype.b = 3;
Sample.prototype.c = 4;
console.log(o.a); // 1
console.log(o.b); // 2
console.log(o.c); // 4
console.log(o.d); // undefined
```

Sample.prototype：「Sample」対象に新しいプロパティを追加する

対象の継承

```
var o = {
  a: 2,
  m: function(){
    return this.a + 1;
  }
};
console.log(o.m()); // 3
var p = Object.create(o);
p.a = 4;
console.log(p.m()); // 5
```

pはOから継承された。

Object対象

- Object.assign(): 指定の対象からコピーして新しい対象を生成する

```
const target = { a: 1, b: 2 };
const source = { b: 4, c: 5 };
const returnedTarget = Object.assign(target, source);
console.log(source); // { b: 4, c: 5 }
console.log(target); // { a: 1, b: 4, c: 5 }
console.log(returnedTarget); // { a: 1, b: 4, c: 5 }
```

Object対象

- Object.create(): 既存の対象から継承して新しい対象を作成する。

```
const person = {
  isHuman: false,
  printIntroduction: function() {
    console.log(`My name is ${this.name}. Am I human? ${this.isHuman}`)
  }
};
const me = Object.create(person);
me.name = 'Matthew';
me.isHuman = true;
me.printIntroduction();
```

Object対象

- Object.freeze()：対象を凍結する。変更できなくなる。

```
const obj = {
  prop: 42
};
Object.freeze(obj);
obj.prop = 33;// Throws an error in strict mode
console.log(obj.prop);
```

JSON文字列と対象

- JSON.parse(json)：文字列から対象に変換する。

```
const json = '{"result":true, "count":42}';
const obj = JSON.parse(json);
console.log(obj.count);// expected output: 42
console.log(obj.result);
```

JSON文字列と対象

- JSON.stringify(obj)：対象を文字列に変更する。

```
console.log(JSON.stringify({ x: 5, y: 6 }));
// expected output: "{"x":5,"y":6}"
console.log(JSON.stringify([new Number(3), new String('false'), new Boo
// expected output: "[3,"false",false]"
console.log(JSON.stringify({ x: [10, undefined, function(){}, Symbol('
// expected output: "{"x":[10,null,null,null]}"
console.log(JSON.stringify(new Date(2006, 0, 2, 15, 4, 5)));
// expected output: ""2006-01-02T15:04:05.000Z""
```

対象の比較

```
var fruit = {name: "apple"};
var fruitbear = {name: "apple"};

fruit == fruitbear // return false
fruit === fruitbear // return false
```

```
var fruit = {name: "apple"};
var fruitbear = fruit;
fruit == fruitbear // return true
fruit === fruitbear // return true
```

各パラメータを比較する

宿題

給料の対象を設計し、下記の要素があり：

- 社員名前　基本給　役職給　税
- 給料総額計算関数
- 実際お支払い給料の計算関数
  下記のJOSN文字列から対象に変換して、各社員の給料を計算する

```
"[
    {"name":"張さん","basic":5000,"position":1000,"tax":200},"
    {"name":"李さん","basic":6000,"position":2000,"tax":500},"
    {"name":"王さん","basic":7000,"position":3000,"tax":800},
]"
```