

ソフトウェア開発 第 5 回目授業

平野 照比古

2017/10/20

レポート課題 4.1 について

ソフトウェア開発
第 5 回授業

平野 照比古

レポート課題につ
いて

```
for( i in window) console.log(`${i} ${window[i]}`);
```

として実行する。

- ▶ 特定のものをチェックするのではなくすべてを見て、どのようなプロパティがあるかを見てほしい。
- ▶ 見慣れた関数名 (alert()) がいくつか見つかるはず
- ▶ 新しいオブジェクトが出てきて、文書化されていないときなどに利用できる。
- ▶ コンソールでオブジェクトを出力すれば同様のことは可能

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェ
クトの継承

WeakMap によるイ
ンスタンスの安全
性の確保

エラーオブジェクト
について

エラー処理の例
エラーからの復帰

レポート課題 4.2 について

ソフトウェア開発
第 5 回目授業

平野 照比古

レポート課題につ
いて

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェ
クトの継承

WeakMap によるイ
テラブル安全
性の確保

エラーオブジェクト
について

エラー処理の例
エラーからの復帰

- ▶ 出力結果は [{}, {}]

```
>JSON.stringify(persons, ["year"]);  
"[{}], {}]"
```

- ▶ year の値を出すためには親のキーも含める。

```
>JSON.stringify(persons, ["year", "birthday", "day", "month", "name"]);  
"[{"birthday":{"year":2001,"day":1,"month":4},"name":"foo"},  
{"birthday":{"year":2010,"day":5,"month":5},"name":"Foo"}]"
```

レポート課題 4.3 について

ソフトウェア開発
第 5 回授業

平野 照比古

レポート課題につ
いて

class の実体

オブジェクト属性

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェ
クトの継承

WeakMap によるイ
ンスタンスの安全
性の確保

エラーオブジェクト
について

エラー処理の例

エラーからの復帰

同様のことができることの確認が少ないレポートが多
かった。

レポート課題 4.4 について

ソフトウェア開発
第 5 回目授業

平野 照比古

- ▶ ルーブリックにヒントを書いておいた。月と日の値はデフォルトの値を書いておくとプログラムが簡単になる。
- ▶ 仮引数に値が与えられないときは undefined が渡されることを利用する。

```
age(year, month=1, day=1) {  
  let today;  
  if(year === undefined) today = new Date();  
  else today = new Date(year, month-1, day);  
  let Age = today.getFullYear() - this.birthday.year;  
  if(today.getTime() <  
    new Date(today.getFullYear(),  
              this.birthday.month-1,  
              this.birthday.day).getTime()) Age--;  
  return Age;  
}
```

レポート課題につ
いて

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェ
クトの継承

WeakMap によるイ
ンスタンスの安全
性の確保

エラーオブジェクト
について

エラー処理の例
エラーからの復帰

レポート課題 4.6 について

ソフトウェア開発
第 5 回授業

平野 照比古

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

```
set age(x){  
  window.alert("値をセットできません");  
}
```

- ▶ セッターには仮引数が必要
- ▶ 通常はこの値をチェックして、条件を満たせば該当する値をセットし、しかるべき追加の処理を行う

従来の ECMAScript では関数を用いてクラスを定義する。`class` でもその手法が使われていることを示す。

エラーからの復帰

従来の方法によるオブジェクト指向

ソフトウェア開発
第 5 回目授業

平野 照比古

前回の授業の Person を従来の方法で定義する。

```
function PersonF(name, year, month, day, hometown = "神奈川県"){  
  this.name = name;  
  this.birthday = {  
    year : year,  
    month : month,  
    day : day  
  };  
  this["hometown"] = hometown;  
}
```

- ▶ この定義では constructor の内容をそのまま関数にしている。
- ▶ メソッドなどが定義されていないが、それについては後で解説

レポート課題について

class の実体

オブジェクト属性

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

実行結果

ソフトウェア開発
第 5 回目授業

平野 照比古

レポート課題につ
いて

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェ
クト

WeakMap によるイ
ンスタンスの安全
性の確保

エラーオブジェクト
について

エラー処理の例
エラーからの復帰

```
>p = new PersonF("foo",2001,4,1);  
PersonF {name: "foo", birthday: {…}, hometown: "神奈川県"}
```

関数オブジェクトの属性

ソフトウェア開発
第 5 回目授業

平野 照比古

レポート課題につ
いて

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェ
クトの継承

WeakMap によるイ
ンスタンスの安全
性の確保

エラーオブジェクト
について

エラー処理の例
エラーからの復帰

- ▶ JavaScript の関数オブジェクトの属性は prototype、class と extensible の 3 つ
- ▶ prototype 属性はオブジェクトの継承に関するもので、最後に説明

- ▶ `class` 属性はオブジェクトの型情報を表す文字列
- ▶ この属性を設定する方法はない。
- ▶ クラス属性は間接的にしか得られない。
- ▶ 組み込みのコンストラクタで生成されたオブジェクトではそのクラス名が間接的に得られる。
- ▶ 独自のコンストラクタ関数では、"`Object`"しか得られない。

前の例 PersonF では次のようになる。

```
>p = new PersonF("foo",2001,4,1);  
PersonF {name: "foo", birthday: {...}, hometown: "神奈川県"}  
>`${p}`;  
"[object Object]"
```

- ▶ PersonF() コンストラクタには toString() メソッドが定義されていない。
- ▶ もともとの Object で定義されている toString() が呼び出される。

レポート課題について

class の実体

オブジェクト属性

class 属性

class.prototype 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

- ▶ オブジェクトに対してプロパティの追加ができるかどうかを指定
- ▶ この属性の取得や設定ができる関数がある。
- ▶ 属性の取得は `Object.isExtensible()` に調べたいオブジェクトを引数にして渡す。
- ▶ オブジェクトのプロパティを拡張できなくするためには `Object.preventExtension()` に引数として設定したいオブジェクトを渡す。

レポート課題について

class の実体

オブジェクト属性

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

extensible 属性の実行例

平野 照比古

```
>p = new PersonF("foo",2001,4,1);
PersonF {name: "foo", birthday: {...}, hometown: "神奈川県"}
PersonF
>p.mother = "Alice";
Alice
>p.grandmother = "Old Alice";
Old Alice
>Object.preventExtensions(p);
PersonF
>p.father = "Bob";
Bob
>p.father;
undefined
>delete p.mother;
true
>p.mother;
undefined
```

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

エラーオブジェクトについて

エラーからの復帰

- ▶ `Object.preventExtensions(p)` を実行する前では存在しない属性の追加が可能 (`p.mother`)。
- ▶ `Object.preventExtensions(p)` を実行後は、新しい属性が定義できない (`p.father`)。
- ▶ `Object.preventExtensions(p)` では属性の削除は禁止できない (`delete p.mother` が成功し、値を参照すると `undefined` が返る)。

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

属性の削除の禁止-Object.seal()

ソフトウェア開発
第 5 回目授業

平野 照比古

この関数を実行されたオブジェクトは削除の禁止を解除できない。

```
>Object.seal(p);
PersonF
>Object.isSealed(p);
true
>delete p.grandmother;
false
>p.grandmother;
Old Alice
>p.grandmother = "Very Old Alice";
Very Old Alice
>p.grandmother;;
Very Old Alice
```

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

属性の削除の禁止-実行例 (解説)

ソフトウェア開発
第 5 回目授業

平野 照比古

レポート課題につ
いて

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェ
クトの継承

WeakMap によるイ
ンスタンスの安全
性の確保

エラーオブジェクト
について

エラー処理の例
エラーからの復帰

- ▶ すでに seal された状態になっているかどうかは、`Object.isSealed()` で調べる。
- ▶ seal された状態ではプロパティを削除できない (`delete p.grandmother` の結果が `false`)。
- ▶ seal された状態ではプロパティの値を変えることができる (`p.grandmother` の値の書き換えができています)。

オブジェクトの固定化-Object.freeze()

ソフトウェア開発
第 5 回目授業

平野 照比古

レポート課題について

class の実体

オブジェクト属性

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

- ▶ オブジェクトを最も強く固定するためには Object.freeze() を用いる。
- ▶ この状態を確認するためには Object.isFrozen() を用いる。

```
>Object.freeze(p);
PersonF
>p.grandmother = "Very Very Old Alice";
Very Very Old Alice
>p.grandmother;
Very Old Alice
```

オブジェクトの固定化-実行例 (解説)

ソフトウェア開発
第 5 回目授業

平野 照比古

レポート課題について

class の実体

オブジェクト属性

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

- ▶ `p.grandmother` の値が設定できていない。
- ▶ このような状態で属性値を変えたい場合には、属性にたいするセッターメソッドを定義する

オブジェクトの固定化-注意

ソフトウェア開発
第 5 回目授業

平野 照比古

このオブジェクトのプロパティ birthday の値はオブジェクトで、seal されていない。

```
>p.birthday;
{year: 2001, month: 4, day: 1}
>delete p.birthday;
false
>p.birthday = {};
{}
>p.birthday;
{year: 2001, month: 4, day: 1}
>delete p.birthday.year;
true
>p.birthday;
{month: 4, day: 1}
```

p.birthday は削除できないし、値の変更もできない。一方で、p.birthday.year は削除できる。
理由はレポート問題 (この後にも同じような状況が起こる)

レポート課題について

class の実体

オブジェクト属性

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

オブジェクトの操作に対する安全性の確保

ソフトウェア開発
第 5 回目授業

平野 照比古

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

- ▶ JavaScript におけるクラスとそれから生成されるインスタンスは通常のオブジェクト指向言語と異なり、構造を変えることができる。
- ▶ 固定化はインスタンスごとに行う。
- ▶ 固定化などはコンストラクタ内で `this` の `extensible` 属性を設定することで可能
- ▶ 別の方法について後で解説する。

- エラーからの復帰

前の例でメソッドを prototype に追加

ソフトウェア開発
第 5 回目授業

平野 照比古

```
PersonF.prototype = {
  toString:function() {
    return `${this.name}さんは`+
      `${this.birthday.year}年${this.birthday.month}月${this.birthday.day}日に` +
      `${this.hometown}で生まれました。`;
  },
  get age(){
    let today = new Date();console.log(x);
    let age = today.getFullYear() - this.birthday.year;
    if(today.getTime() <
      new Date(today.getFullYear(),
        this.birthday.month-1,
        this.birthday.day).getTime()) age--;
    return age;
  },
  constructor: PersonF
}
```

レポート課題につ
いて

class の実体

オブジェクト属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェ
クトの継承

WeakMap によるイ
ンスタンスの安全
性の確保

エラーオブジェクト
について

エラー処理の例

エラーからの復帰

メソッドを prototype に追加-解説

ソフトウェア開発
第 5 回目授業

平野 照比古

- ▶ これらの定義は前と同じ
- ▶ オブジェクトリテラルの形式で prototype に代入
- ▶ 異なるところは constructor プロパティに自分自身を定義している

レポート課題について

class の実体

オブジェクト属性

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

関数によるメソッドの継承

ソフトウェア開発
第 5 回目授業

平野 照比古

JavaScript では prototype を用いることでメソッドの継承が可能

Person を継承して、学籍番号を追加のプロパティとする Student オブジェクトのコンストラクタ関数である。

```
const StudentF = (function(){
  let id = 10000;
  return function(name, year, month, day, hometown = "神奈川") {
    this.name = name
    this.birthday = {
      year : year,
      month : month,
      day : day
    };
    this["hometown"] = hometown;
    this.id = id++;
  }
})();
StudentF.prototype = new PersonF();
StudentF.prototype.constructor = StudentF;
```

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

- ▶ 関数を用いた継承では `super` が使えないので、継承元のクラスで定義されたプロパティ (`name` など) は継承先で定義する
- ▶ メソッドはインスタンスで共有されるので継承元で定義をする必要はないが、参照できるようにする必要がある。
- ▶ 継承元のクラスを作成して継承するクラスの `prototype` を書き直している
- ▶ このままでは `constructor` プロパティが `PersonF` になるので、`StudentF.prototype.constructor` を継承するクラスに置き換える

レポート課題について

`class` の実体

オブジェクト属性

`class` 属性

`extensible` 属性

`prototype` 属性

`prototype` の使用例

関数によるオブジェクトの継承

`WeakMap` によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

- ▶ あるクラスのインスタンスごとにプロパティを保存するのに `this.name` のようなキーを用いると、データの安全性が保障されない。
- ▶ クラス式のクローージャに入れるとクラスで共通のものになってしまい、目的に合わない。
- ▶ インスタンスごとに値をしまうためにはオブジェクトと値の集まりを関連付ける必要がある。
- ▶ WeakMap はオブジェクトと同じようにキーと値の集まり
- ▶ WeakMap のキーはオブジェクトしか使用できない。

レポート課題について

class の実体

オブジェクト属性

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

ソフトウェア開発
第 5 回目授業

```
const Person = (function() {
  const properties = new WeakMap();
  return class {
    constructor(name, year, month, day, hometown="神奈川県"){
      properties.set(this,
        {
          "name" : name,
          "birthday" :{
            "year" : year,
            "month": month,
            "day" : day
          },
          "hometown" : hometown
        });
    }
    get name(){
      return properties.get(this).name;
    }
    get birthday(){
      return properties.get(this).birthday;
    }
    get hometown(){
      return properties.get(this).hometown;
    }
  }
})
```

- class 属性
- extensible 属性
- prototye 属性
- prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラーからの復帰

平野 照比古

エラーからの復帰

- ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ▶ ↺ 🔍 ↻

平野 照比古

- class 属性
- extensible 属性
- prototye 属性
- prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラーからの復帰

実行例 (2)

メソッドも正しく動作する。

```
>p.age;
```

```
16
```

```
>`${p}`;
```

```
"foo さんは 2001 年 4 月 1 日に神奈川で生まれました。"
```

プロパティの値は変更できない。

```
>p.name = "Alice"; //name の値を "Alice" にする試み
```

```
>p.name;
```

```
"foo"
```

//name の値は変更されていない。

平野 照比古

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

ていゝる

エラーオブジェクト

- ▶ エラーオブジェクトとはエラーが発生したことを知らせるオブジェクト
- ▶ 通常は計算の継続ができなくなったときにエラーオブジェクトをシステムに送る操作が必要
- ▶ これを通常、エラーを投げる (throw する) という。

下表のようなプロパティがある。

Table: エラーオブジェクトのプロパティ

プロパティ	説明
message	エラーに関する詳細なメッセージ。コンストラクタで渡された文字列か、デフォルトの文字列
name	エラーの名前。エラーを作成したコンストラクタ名になる

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

エラー処理の例

以前に定義した Person クラスのコンストラクタに与えられた引数をチェックして不正な値の場合にはエラーを投げるように書き直したもの

```
class Person{
    static checkName(name) {
        if(name === "") throw new Error("名前がありません");
        return name;
    }
    static checkDate(y, m, d) {
        if(m<1 || m>12) throw new Error("月が不正です");
        let date = new Date(y,m,0);
        if(d<1 || d>date.getDate()) throw new Error("日が不正です");
        return {year: y, month: m, day: d};
    }
    constructor(name, year, month, day, hometown="神奈川県"){
        this.name = Person.checkName(name);
        this.birthday = Person.checkDate(year, month, day);
        this["hometown"] = hometown;
    }
    toString() {
        ...
    }
    get age(){
        ...
    }
}
```

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

- ▶ name が空文字であればエラーを発生させ、正しければ値をそのまま返すクラスメソッドを定義
- ▶ 正しい日付でないとエラーを発生させ、正しいときは日付のオブジェクトを返すクラスメソッドを定義
 - ▶ 月の値の範囲をチェックしている。
 - ▶ 与えられた年と月からその月の最終の日を求めている。
Date.getMonth() の戻り値が 0(1 月) から 11(12 月) になっているので、new Date(y,m,0) により翌月の 1 日の 1 日前、つまり、問題としている月の最終日が設定できる。
 - ▶ 与えられた範囲に日が含まれていなければエラーを発生
 - ▶ 日付のオブジェクトを作成し、戻り値として返す。
- ▶ 名前と日付を指定したプロパティにクラスメソッドからの戻り値で設定

- class 属性
- extensible 属性
- prototye 属性
- prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

- ▶ 通常の日時ならば問題なく、オブジェクトが構成される。

```
>p = new Person("foo",1995,4,1);
Person {name: "foo", year: 1995, month: 4, day:
```

- ▶ うるう年の 1996 年には 2 月 29 日あるので、エラーは起こらずオブジェクトが作成できる。

```
>p = new Person("foo",1996,2,29);
Person {name: "foo", year: 1996, month: 2, day: 29}
```

- ▶ うるう年ではない 1995 年には 2 月 29 日がないので、エラーが起きる。

```
>p = new Person("foo",1995,2,29);
Uncaught Error: 日が不正です (...)
```

- ▶ 不正な月や日では当然、エラーが起こる。

```
>p = new Person("foo",1995,13,29);
Uncaught Error: 月が不正です (…)
```

```
>p = new Person("foo",1995,12,0);
Uncaught Error: 日が不正です (...)
```

class の実体
1}
オブジェクト属性

- class 属性
- extensible 属性
- prototye 属性
- prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

エラーからの復帰-try{...}catch{...} 構文

ソフトウェア開発
第 5 回目授業

平野 照比古

レポート課題につ
いて

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェ
クトの継承

WeakMap によるイ
ンスタンスの安全
性の確保

エラーオブジェクト
について

エラー処理の例
エラーからの復帰

- ▶ 前節の例ではエラーが発生するとそこでプログラムの実行が停止
- ▶ エラーが発生したときに、投げられた (throw された) エラーを捕まえる (catch する) ことが必要
- ▶ このためには try{...}catch{...} 構文を使用

try{...}catch{...} 構文の解説

ソフトウェア開発
第 5 回目授業

平野 照比古

- ▶ try のブロック内にエラーが発生する可能性があるコードを記述
- ▶ エラーが発生したときは throw を用いてエラーを発生
- ▶ エラーが発生すると catch(e) ブロックが実行
- ▶ catch の後にある e は投げられたエラーオブジェクトが渡される。
- ▶ この変数のスコープは catch 内
- ▶ finally{} を付けることもできる。
- ▶ try や catch の処理が実行後、必ず呼び出される。これは try や catch の部分が return 文、break 文、continue 文、return 文や新しい例外を投げたとしても呼び出される。
- ▶ try{...}catch{...} 構文は入れ子にできる。投げられたエラーに一番近い catch にエラーが捕まえられる。

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

- ▶ 前の例で `try{...}catch{...}` 構文を用いてオブジェクトが正しくできるまで繰り返すようにしたもの
- ▶ ブラウザで実行することを想定

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

```
function test() {  
  for(;;) {  
    try {  
      let y = Number(prompt("生まれた年を西暦で入力してください"));  
      console.log('年:${y}');  
      let m = Number(prompt("生まれた月を入力してください"));  
      console.log('月:${m}');  
      let d = Number(prompt("生まれた日を入力してください"));  
      console.log('日:${d}');  
      return new Person("foo", y, m, d);  
    } catch(e) {  
      console.log(e.name+": "+e.message);  
    } finally{  
      console.log("finnally");  
    }  
  }  
}
```

レポート課題につ
いて

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェ
クトの継承

WeakMap によるイ
ンスタンスの安全
性の確保

エラーオブジェクト
について

エラー処理の例
エラーからの復帰

- ▶ テストを繰り返す関数 `test()` が定義
- ▶ 無限ループのなかで正しいパラメータが与えられたときに作成されたオブジェクトを戻り値にして関数の実行が終了
- ▶ `try{}` 内にはエラーが発生するかもしれないコードを中に含める。
 - ▶ 年、月、日の入力を `prompt` によるダイアログボックスから入力
 - ▶ 戻り値は文字列なので、`Number` で数に変換
 - ▶ 代入された値が正当であればその値をコンソールに出力
- ▶ 与えられた入力が正しくなければエラーが投げられ、`catch(e)` の中に制御が移動
- ▶ `catch(e)` における `e` には発生したエラーオブジェクトが渡されるので、コンソールにその情報を出力

レポート課題について

class の実体

オブジェクト属性

class 属性

extensible 属性

prototype 属性

prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例

エラーからの復帰

実行例

ソフトウェア開発
第 5 回目授業

平野 照比古

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰

```
>p = test();
  年:2010
  月:4
  日:31
Error:日が不正です
finally
  年:2010
  月:4
  日:30
finally
Person {name: "foo", birthday: {...}, hometown: "神奈川"}
```

- ▶ 200 年 4 月 31 日は不正な日付なので、`console.log()` の出力はない。
- ▶ その代り、エラーの内容が出力されている。
- ▶ エラー出力後は `finally` ブロックが実行されている。
- ▶ 入力データが正しい場合でも `finally` ブロックが実行されている。

レポート課題について

class の実体

オブジェクト属性

class 属性
extensible 属性
prototype 属性
prototype の使用例

関数によるオブジェクトの継承

WeakMap によるインスタンスの安全性の確保

エラーオブジェクトについて

エラー処理の例
エラーからの復帰