よく使うデータ構造

- Objective-Cのドキュメントの読み方
- 配列
- 辞書
- enum

Objective-Cのドキュメントの読み方

- 各クラスはドキュメントに書いてあるプロパティしか持たない
- 各クラスはドキュメントに書いてあるメソッドしか持たない

ドキュメントを読めば「各クラスでできること」、「各クラスの使い方」が全て書いてある！！

※Google先生に聞くよりドキュメントを読んだほうが早い。ただし、全部英語...

配列

- 値を順番にまとめたもの
- 添字で前から何個目かを指定しての代入や値の取得ができる
- 可変の配列を使用したいときはNSMutableArrayを使用する
- JSONの配列はNSArrayに対応する

```
// クラス型の値を入れる
// NSIntegerなどのプリミティブ型の値は入れられない
NSArray *array = @[someObject, @"Hello, World!", @42];
// array[1]と書くと@"Hello"を取得できるが、id型になるのでキャストが必要
NSString *a = (NSString *)array[1];

// NSArray<NSString *>のように書くと、NSStringのみを入れるように指定できる
// ※NSString以外を入れてもコンパイルエラーにはならず、警告が表示されるだけ
// ※light weight genericsという
NSArray<NSString *> *array2 = @[@"string", @"Hello, World!", @"any string"];
// NSStringのみを指定したので、キャストは不要
NSString *a2 = array2[1];
```

辞書

- 値をキーに対応させてまとめたもの
- 可変の辞書を使用したいときはNSMutableDictionaryを使用する
- JSONのオブジェクトはNSDictionaryに対応する

```
NSDictionary *dictionary = @{
        @"anObject" : someObject,
    @"helloString" : @"Hello, World!",
    @"magicNumber" : @42,
         @"aValue" : someValue
};
// dictionary[@"helloString"]で@"Hello, World!"を取得できる
// NSArrayと同様にid型での取得なのでキャストが必要
// NSArrayと同様にlight weight genericsを使用できる
```

列挙体

- 複数の定数をひとまとめにしたもの
- 種類を表す定数など「値に特別な意味は無いがまとめておきたい定数」を定義するときに使用するのが本来の使い方
- 自動で数値が採番されるため配列の添字に使われることもある

```
// 果物の種類をまとめた例
typedef NS_ENUM(NSInteger, Fruit) {
    FruitApple,  // 値は0
    FruitBanana  // 値は1
};

// 下記のように変数の型として使える。
Fruit enumVariable = FruitApple;

NSArray *costs = @{@100, @200} // {りんごの値段、バナナの値段};
NSNumber *value = costs[FruitApple]; // FruitAppleは0なので、りんごの値段が取得できる
```