プログラミングに最低限必要な知識

- 変数
- 式の評価
- 変数のスコープ
- リテラル
- 制御構文
- WebAPI
- JSON
- XML

前提

- 調べて出てくる情報が読める様になる程度の知識をつけるのが目的
  ※プログラミングの超入門書に書いてあることを要約した程度の知識
  ※まともな入門書には詳しい説明とともに全部書いてあるはず
- Objective-Cに関することは詳しく説明しない
  ※このスライドの内容が理解できれば自分で調べられるはず
- 理解しきれてない可能性のものをピックアップしたもの

変数

- 値を入れておくためのもの
- 使う前に必ず「宣言する」必要がある
- 「*」はクラス型のときのみつける

  ※クラス型：@interfaceで宣言されている型のこと

```
//  「変数に入れる値の型  変数名;」で宣言する
NSInteger number; // NSInteger型の変数number
NSString *string; // NSString型の変数string
```

変数

- 値を入れることを「代入する」という

```
NSInteger number;

//  「1」という値を変数numberに代入
number = 1;

//※「NSInteger number = 1;」と書くと「変数numberを定義して1を代入する」という意味になる
```

変数

- 変数の中に入っている値を別の変数に代入することもできる
- 「代入」は変数の中身を上書きするのと同じ意味になる

```
NSInteger number1 = 1;
NSInteger number2 = 2;
// 変数number2に変数number1の中に入っている値を代入
number2 = number1;

// Q：この時点でのnumber1とnumber2それぞれに入っている値は何？
```

変数

- 代入しないと変数の中身は謎
- 代入を忘れると重大なバグの原因になる
  ※そもそも、値を代入されない変数は使わない変数なので削除するべき
  ※使わない変数を残すとコードが読みにくくなるだけなので絶対に削除すること！！

```
// プリミティブ型の変数は代入されなければ「0」かもしれないし「100」かもしれないし他の値かもしれない
NSInteger number1;

// クラス型の変数は代入されなければnil
NSString *string;
```

式の評価

- 「1+2」のような計算や「メソッドの呼び出し」のことを式という
- 式を計算することを「評価する」という
- 変数には式を評価した結果を入れる

  ※代入は右辺の評価がすべて完了してから行われる

```
// 「1 + 2」を評価した結果をnumberに代入する
NSInteger number = 1 + 2;

// 「[NSString stringWithFormat:@"number = %ld", number]」を評価した結果をstringに代入する
// ※評価した結果は「stringWithFormat」メソッドの戻り値になる
NSString *string = [NSString stringWithFormat:@"number = %ld", number];

// Q:変数stringに入っている値は何？
```

変数のスコープ

- 変数には使用可能な範囲がある
- 「使用可能な範囲」は「スコープ(scope)」という
- 言語によってスコープは異なる

変数のスコープ

- 変数は宣言した後でしか使用できない
- Objective-Cの変数は基本的に「ブロックスコープ」
  ※ブロックスコープ：「使用可能な範囲」がブロック「{...}」で区切られていること

```
@property NSString *globalString;

- (void)methodWithNumber1:(NSInteger)number1 {
    // Q: この行で使える変数はどれ？
    NSInteger number2 = 1;
    if (number1 == number2) {
        // Q: この行で使える変数はどれ？
        NSString *string1 = @"number1 equals number2";
    } else {
        NSString *string2 = @"number1 do not equals number2";
        // Q: この行で使える変数はどれ？
    }
    // Q: この行で使える変数はどれ？
}
```

リテラル

- ある値の直接的な表現のこと

```
// 右辺がぞれぞれの型に対応したリテラルの例
NSInteger n = 1;
NSNumber n2 = @1;
NSString *s = @"Any string you want";
NSArray<NSString *> *a = @[@"string1", @"string2", @"string3"];
NSDictionary<NSString *, NSString *> *d = @{
    @"key1" : @"value1",
    @"key2" : @"value2",
    @"key3" : @"value3",
    @"key4" : @"value4",
};
CGFloat f = 0.1;
```

制御構文

- if (条件) {} else {}
  条件分岐
- switch (式) { case 値:, case 値: }
  式を評価した結果と一致するcaseを実行する
- while (条件) {}
  条件を満たす間ブロックの中を繰り返す
- for (終了条件付きの条件) {}
  条件を満たす間ブロックの中を繰り返す
  配列の要素を一つずつ取り出して繰り返す

WebAPI

- あるURLにアクセスすると文字列を返してくれるWebサービスのこと
- 基本的にHTTPを用いてURLにアクセスする
- リクエストする方法と返してくれる文字列の詳細は「WebAPIの仕様書」か「HTTPレスポンスの内容」を見ないとわからない
- 大抵のWebAPIはJSONまたはXMLを返す

JSON

- データをJavaScriptのオブジェクト形式で構造化した文字列
- 下記の値を組み合わせて表現する
- Objective-Cで使用するときはそれぞれの値を対応した型の変数に入れて使用する

```
オブジェクト:{ "キー1" : 値、"キー2" : 値、...}
配列：[値, 値, 値, ...]
文字列:"文字列"
数値:1, 2, 3, ...
Bool：true, false
※値は上記５種類のどれでも良い
※「オブジェクトの中にオブジェクトを入れる」や「配列の中に配列を入れる」といった入れ子になることもある
```

XML

- データをマークアップした形式で構造化した文字列
- HTMLと同じ形式　※HTMLと異なりタグは任意
- 値はすべて文字列