ソフトウェア開発
第 2 回目授業

平野 照比古

2016/9/30

プリミティブデータ型

型	説明
Number	浮動小数点数だけ
String	文字列型、1 文字だけのデータ型はない。ダブルクオート (") かシングルクオート (') で囲む。
Boolean	`true` か `false` の値のみ
`undefined`	変数の値が定義されていないことを示す
`null`	`null` という値しか取ることができない特別なオブジェクト

変数や値の型を知りたいときは `typeof` 演算子を使う。

Number 型

JavaScript で扱う数は 64 ビット浮動小数点形式

- 整数リテラル 10 進整数は通常通りの形式である。16 進数を表す場合は先頭に `0x` か `0X` をつける。`0` で始まりそのあとに `x` または `X` が来ない場合には 8 進数と解釈される場合があるので注意が必要である。
  `strict` モードではこの形式はエラーとなる。
- 浮動小数点リテラル 整数部、そのあとに必要ならば小数点、小数部そのあとに指数部がある形式である。

特別な Number

- `Infinity` 無限大を表す読み出し可能な変数である。オーバーフローした場合や `1/0` などの結果としてこの値が設定される。
- `NaN` Not a Number の略である。計算ができなかった場合表す読み出し可能な変数である。文字列を数値に変換できない場合や `0/0` などの結果としてこの値が設定される。

String 型

文字列に関する情報や操作には次のようなものがある 。

メンバー	説明
length	文字列の長さ
indexOf(needle,start)	needle が与えられた文字列内にあればその位置を返す。start の引数がある場合には、指定された位置以降から調べる。見つからない場合は -1 を返す。
lastIndexOf(needle,start)	needle が与えられた文字列内にあればそれが一番最後に現れる位置を返す。start の引数がある場合には、指定された位置以前から調べる。見つからない場合は -1 を返す。

String 型

メンバー	説明
split(separator,limit)	separator で与えられた文字列で与えられた文字列を分けて配列で返す。セパレーターの部分は返されない。2 番目の引数はオプションで分割する最大数を返す。
substring(start,end)	与えられた文字列の start から end の位置までの部分文字列を返す。

Bool 型

`true` と `false` の 2 つの値をとる。この 2 つは予約語である。論理式の結果としてこれらの値が設定されたり、論理値が必要なところでこれらの値に設定される。詳しくは後述する。

undefined

値が存在しないことを示す読み出し可能な変数である。
変数が宣言されたのに値が設定されていない場合などはこの値に初期化される。

null

typeof null の値が "object"であることを示すように、オブジェクトが存在しないことを示す特別なオブジェクト値（であると同時にオブジェクトでもある）である。

JavaScript における変数

- 変数名はアルファベットまたは_(アンダースコア) で始まる英数字または_で始まる文字列
- 大文字と小文字は区別される。
- 変数の宣言は `var` で行う。
- 宣言時に初期化ができる。
- 非 `strict` モードのときは変数は宣言をしなくても使用できる。初期化していない変数を使うとエラーが起こる。詳しくは後述する。
- 変数に保存するデータの型には制限がない。途中で変更することもできる。

配列の宣言と初期化

配列を使うためには、変数を配列で初期化する必要がある。変数の宣言と同時に行ってもよい。

```
var a = [];
var b = [1,2,3];
```

a は空の配列で初期化されている。b は b[0]=1,b[1]=2,b[2]=3 となる配列で初期化されている。

配列に関する注意

- 配列の各要素のデータの型は同じでなくてもよい。
- 実行時に配列の大きさを自由に変えられる。
- 配列の要素に配列を置くことができる。

```
var a=[1,[2,3,4],"a"];
```

配列のメソッド

メンバー	説明
`length`	配列の要素の数
`join(separator)`	配列を文字列に変換する。`separator` はオプションの引数で、省略された場合はカンマ , である。
`pop()`	配列の最後の要素を削除し、その値を返す。配列をスタックとして利用できる。
`push(i1,i2,...)`	引数で渡された要素を配列の最後に付け加える。配列をスタックやキューとして利用できる。
`shift()`	配列の最初の要素を削除し、その値を返す。配列をキューとして利用できる。

配列のメソッド

`slice(start,end)`	`start` から `end` の前の位置にある要素を取り出した配列を返す。元の配列は変化しない。
`splice(start,end,i1,i2,...)`	`start` から `end` の位置にある要素を取り出した配列を返す。元の配列からこれらのデータを取り除き、その位置に `i1,i1,...` 以下の要素を付け加える。

代入、四則演算

- +演算子は文字列の連接にも使用できる。+演算子は左右のオペランドが Number のときだけ、数の和をとる。どちらかが数でもう一方が文字列の場合は数を文字列に直して、文字列の連接を行う。

```
1+2   => 3
1+"2" => 12
```

- そのほかの演算子 (-*/) については文字列を数に変換してから数として計算する。
- 文字列全体が数にならない場合には変換の結果が `NaN` になる。
- 整数を整数で割った場合、割り切れなければ小数となる。小数部分を切り捨てたいときは `Math.floor()` を用いる。

```
1/3 => 0.3333333333333333
Math.floor(1/3) =>0
```

論理演算子

Boolean 型に対して使用できる演算子は次の 3 つ

- ! 論理否定
- && 論理積
- || 論理和

論理演算子に関する注意

論理演算子を Boolean 型でない値を与えると元の値が Boolean 型に変換されてから実行される。次の値は `false` に変換される。

- 空文字列 `""`
- `null`
- `undefined`
- 数字の 0
- 数値の `NaN`
- Boolean の `false`

論理演算子に関する注意

論理和や論理積では左のオペランドの結果により、式の値が決まる場合は右のオペランドの評価は行われない。たとえば、論理和の場合、左の値が `true` であれば右のオペランドの評価が行われない。

```
var a=1; true ||(a=3);
```

では変数 `a` の値は 1 のままである。

比較演算子

比較演算子は比較の結果、Boolean の値を返す演算である。C 言語と同様の比較演算子が使用できる。>,>=,<,<=など。等しいことを比較するためには==(等価比較演算子) のほかに 型変換を伴わない等価比較演算子=== がある。等価比較演算子==は必要に応じて型変換を行う。

```
0 == "0" => true
0 === "0" => false
```

同様に非等価比較演算子!=にも型変換を伴わない非等価演算子!==がある。
また、`NaN == NaN` の結果は `false` である。値が `NaN` であるかどうかを調べる関数がある。

組み込み関数

JavaScript に初めから組み込まれている関数としては次のものがある。

- `parseInt(string,radix)` `string`(文字列) と `radix`(基数、省略したときは 10) をとり、先頭から見て正しい整数表現のところまで整数に変換する。
- `parseFloat(string)` `string`(文字列) をとり、先頭から見て正しい浮動小数点表現のところまで浮動小数に変換する。
- `isNaN(N)` `N` が数であれば `true`、そうでなければ `false` を返す関数
- `isFinite(N)` `N` が数値または数値に変換できる値でかつ `Infinity` または `-Infinity` でないときに `true`、そうでないとき、`false` を返す。

このほかにも関数があるが、省略する。

組み込みオブジェクト

- Date オブジェクト
  日付や時間に関するデータを扱うオブジェクトである。基本的にはミリ秒単位の値が返ってくるので、実行時間の測定などにも使える。
- Math オブジェクト
  数学的な定数の定義 (円周率など) や三角関数などの関数が定義されている。