ソフトウェア開発
第 2 回目授業

平野 照比古

2018/10/5

レポートの形式 (再掲)

- 一番上に復習問題の用紙を置き、全体をステープラで止める。表紙は不要。
- 問題の結果をそのまま手書きでもよい。
- キャプチャや画像を印刷したときは、関係する説明文や考察も同じ用紙に記述すること。
- ブラウザを全画面表示したものをキャプチャしないこと。
- 必要な結果が入る範囲でウィンドウの表示のサイズを決めること。また、キャプチャ内の文字が読める程度にすること。
- アクティブウィンドウだけのキャプチャすること。
- 実行結果それぞれに対して説明をつけること。最後にまとめた考察を書く。
- ルーブリック評価は点数も含めて必ず自己評価をすること。採点結果との差がなくなるようにするためである。

テンプレートリテラルについて

- 問 2 における出力が間違っているものが多数見受けられた。
- 出力結果は数になる。
- テンプレートリテラルはテキストにあるようにバッククオート (@ の上の文字) で囲む。
- バッククオート、シングルクオート、ダブルクオートを区別すること。

変数のスコープについて

- 定義された変数が有効である範囲を変数のスコープという。
- 関数内で宣言された変数は関数内でしか有効ではない。
- したがって、復習問題のリストで、`foo();` コンソールから実行すると、戻ってきた時点で変数は消えている。したがって、そこで変数 `i` を参照すると「変数がない」というエラーが返る。
- 宣言がないと変数はグローバルで定義される。したがって、変数 `i` の宣言をなくすと、`foo();` の実行後に変数の値が参照できる。
- 変数のスコープは第 3 回の授業でもう一度解説をする。(重要)

CDATA セクションについて

ブラウザの HTML 文書の解釈の手順を理解しておくこと

- ブラウザは要素 (タグ) に基づいて文書の構造を解析する。
- <と>をもとに要素の構成を調べる。要素名については関知しない。
- したがって<script>要素内の比較演算子の<も要素の開始ととらえる。
- そのように解釈されないために書いてある文字をそのまま取り扱うようにするための範囲を定めたのが CDATA セクション
- 構造の解釈が終わると要素の解釈が行われる。
- その際に、<script>要素内に書かれた CDATA セクションの記述は JavaScript の文法では正しくないのでその前に JavaScript のコメントとする必要がある。

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。ES2015 では文字列内に式の値を埋め込み可能なテンプレートリテラル形式 (バッククオート ‘ ではさむ) が追加
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。

計算の途中でこれらの値が得られてもプログラムの実行は中断されない。

String 型のリテラル

- 文字列はダブルクオート (") または シングルクオート (’) ではさむ。
- 1 文字だけの文字の型はない。
- テンプレートリテラルは文字列の中に式の値を埋め込むことができる。
- テンプレートリテラルはバッククオート (‘) ではさむ。
- 埋め込む式は${2+3} のように$ の後に { と } の間に式を記述
- ‘2+3=${2+3}‘ は "2+3=5" という文字列になる。
- 2 つの文字列をつなげる連接演算子+ついては後述

String 型のプロパティとメソッド (1)

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 型のプロパティとメソッド (2)

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。
slice(start,end)	文字列の start から end の前の位置までの部分文字列を返す。値が負のときは文字列末尾から数えた位置を表す。

Bool 型

- `true` と `false` の 2 つの値をとる。
- この 2 つは予約語
- 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。

undefined

- 値が存在しないことを示す読み出し可能な変数
- 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
- 関数に戻り値を指定しないときの値

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。

非プリミティブデータ

プリミティブ型以外のデータ型は「オブジェクト」(() 内はオブジェクトを作成するコンストラクタ) の名称

- 配列 (Array)
- 数学 (Math)
- 日時 (Date)
- 正規表現 (RegExp)
- マップ (Map) とウィークマップ (WeakMap)
- セット (Set) とウィークセット (WeakSet)

これらのデータ型のうち、マップ (Map)、セット (Set)、ウィークセット (WeakSet) 以外は後で解説

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 `let` または `const` で行う。
- `const` による変数の宣言では初期化時の値の変更ができない。
- 従来の宣言方法の `var` は問題があるのでこの授業では使用しない。
- 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
- 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可能

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。

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

a は空の配列で初期化
b は b[0]=1,b[1]=2,b[2]=3 となる配列で初期化

配列に関する注意 (1)

- 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。

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

配列に関する注意 (2)

配列の変数の代入は参照

```
>let a = [0,1,2,3,4];
>let b = a;  //別の変数への代入
>b;
(5) [0, 1, 2, 3, 4]  // a と同じ内容が表示
>b[3]= b[3]*10;      // 4 番目の要素を 10 倍 [0, 1, 2, 30, 4]
30
>a;
(5) [0, 1, 2, 30, 4] // a も b と同じ値の要素を持つ
```

分割代入

配列の要素の一部をまとめて別の変数に代入する。

```
>\Verb+[a,,b,...c] = [0,1,2,3,4,5,6,7];
a ==> 0
b ==> 2
c ==> [3, 4, 5, 6, 7]
```

- 変数 a には 0 番目の、b には 3 番目の、c には 4 番目以降の配列の要素が代入される。
- a と b の間に , , があるので 2 番目の要素は代入されない。
- 4 番目以降の要素はまとめて変数 c に代入

配列が関数の戻り値のとき、戻り値の必要なところだけ利用するために便利な機能
. . . は展開演算子と呼ばれ、配列に対して要素を並べたものを表す。

配列のメソッド (1)

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 (2)

slice(start,end)	start から end の前の位置にある要素までからなる配列を返す。
splice(start,end,i1,i2,...)	start から end の位置にある要素までからなる配列を返す。元の配列からこれらの要素を取り除き、その位置に i1,i1,... 以下の要素を追加
indexOf(value,start)	配列の要素で value に等しい (===) ものを探し、その位置を返す。見つからない場合は-1 を返す。オプションの引数 start はその位置以降から探すことを指定
lastIndexOf(value,start)	配列の要素で value に等しいものを後ろから探し、その位置を返す。見つからない場合は-1 を返す。オプションの引数 start はその位置以前から探すことを指定

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。

```
let 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。

制御構造

- C 言語などと同様に `if` 文、`for` 文、`while` 文、`switch` 文がある。
- 使い方は同様なのでここでは説明をしない。
- Java や C++のように `for` 文の初期化のところに現れる変数を `let` を用いて宣言することもできる。
- この宣言された変数に関する JavaScript 特有の注意は次回解説

組み込み関数

名称	説明
parseInt(string,radix)	引数は string(文字列) と radix(基数、デフォルトは 10)。先頭から見て正しい整数表現の場所まで整数に変換
parseFloat(string)	引数は string(文字列)。先頭から見て正しい浮動小数点表現の場所まで変換
isNaN(N)	N が数であれば false、そうでなければ true を返す。
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。
encodeURIComponent(string)	string を URI エンコードする。
decodeURIComponent(string)	encodeURIComponent(string) の逆の操作
encodeURI(string)	string を URI エンコーディングする。プロトコル部分などはエンコードしない。
decodeURI(string)	encodeURI(string) の逆の操作

組み込みオブジェクト

- 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。
- 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。