演算

演算とは

変数を作成したらデータの加工や計算をすることができ、プログラムで処理することを演算といいます。

演算子とは

どのような演算をするかを決めるための記号を「演算子」と呼び、データ型に応じた演算子を利用します。 また、演算子でない方を「被演算子（オペランド）」といいます。

以下の例では「1」「2」がオペランド、「+」が演算子となります。

```
1 + 2
```

二項演算

二項演算は、数の四則演算（加減乗除）などの2つの数から新たな数を作成することをいいます。 プログラムでは変数や値が 2つ登場する演算子のことをいいます。

四則演算子

プログラムの四則演算子は以下の通りです。

- たし算： +
- ひき算： -
- かけ算： *
- わり算： /
- 余り： %

四則演算の例

変数 hp を宣言して計算結果を表示してみましょう。

```
var hp = 3;
console.log(hp + 1);

value = 1;
hp.log(hp - 2);

value = 5;
hp.log(hp * 3);

hp = 4;
console.log(hp / 5);

hp = 6;
console.log(hp % 5);
```

結果

```
4
-1
15
0.8
1
```

単項演算

単項演算子は 1つの変数に対して処理をおこない、その結果をその変数に代入します。

単項演算の例

```
hp = 5;
++hp;
console.log(hp);

hp = 5;
--hp;
console.log(hp);
```

結果

```
6
4
```

複合演算

複合演算は、演算子を組み合わせて指定し、値に代入します。

+=

変数に値を足して再代入します。

```
hp = 10;
hp += 5; // hp に 5 足した結果を hp に代入
console.log(hp);
```

-=

変数から値を引いて再代入します。

```
hp = 10;
hp -= 5; // hp から 5 を引いた結果を hp に代入
console.log(hp);
```

*=

変数に値をかけて再代入します。

```
hp = 10;
hp *= 5; // hp に 5 をかけた結果を hp に代入
console.log(hp);
```

/=

変数を値で割って再代入します。

```
hp = 10;
hp /= 5; // hp を 5 で割った結果を hp に代入
console.log(hp);
```

文字列の連結

文字列の変数は「＋」で連結することができます。

```
文字列変数1 + 文字列変数2
```

文字連結の例

```
hp = 10;
monster_name = 'スライム';
var status_message = monster_name + 'のHPは' + hp;
console.log(status_message);
```

結果

```
スライムのHPは10
```

文字列リテラル

文字連結の演算子 + を使わずにリテラルテンプレートという方式を利用すると見やすくなります。リテラルテンプレートは変数を ${ } で、全体を「`」（バッククオート）で囲んで処理します。

```
`${変数}文字列${変数}`
```

文字列リテラルを使った例です。

```
var status_message = `${monster_name}のHPは${hp}`;
console.log(status_message)
```

結果

```
スライムのHPは10
```

単項演算

```
var message = 'この物語は';
message += 'フィクションです';
console.log(message);
```

結果

```
この物語はフィクションです。
```

文字列と数値の連結

文字列と数値を + するとテキストで連結されます。

```
message+= 2020;
console.log(message);
```

結果

```
この物語はフィクションです。2020
```

プログラムは右辺から計算される

演算式で左辺と右辺がどちらから処理されるか疑問に思うかもかも知れません。

```
var attack = 10;
hp = 50;
hp = hp - attack;
console.log(hp);
```

JavaScript のようなプログラムでは計算順序にルールがあり「右辺から計算する」ようになっています。

```
hp - attack  //右辺で「50 - 10」が計算
```

次に左辺に代入します。

```
hp = 40  // hp に 40 を代入
```

比較演算子

比較演算子は左の値と右の値を比較して true , false を判定し、条件式の分岐で利用します。

下の表は比較演算子で一致すれば true を返します。

==	A == B	A と B が等しい
!=	A != B	A と B が等しくない
>	A > B	A が B より大きい
<	A < B	A が B より小さい
>=	A >= B	A が B より大きいか等しい（以上）
<=	A <= B	A が B より小さいか等しい（以下）
===	A === B	A と B が等しく、データ型も同じ
!==	A !== B	A と B が等しく、データ型も同じときの否定

論理演算

右辺で比較演算子を使った論理型(bool)の結果を、左辺に代入します。

```
is_bool = (値1 比較演算子 値2);
```

() はつけなくても動作しますが、見た目的にわかりやすいのでつけた方がよいでしょう。

論理演算の例

```
var is_bool;

hp = 10;
is_bool = (hp == 20);
console.log(is_bool);

hp = 0;
is_bool = (hp < -10);
console.log(is_bool);

hp = 0;
is_bool = (hp > -10);
console.log(is_bool);

hp = 10;
is_bool = (hp >= 10);
console.log(is_bool);
```

三項演算

三項演算は論理演算の結果(true, false)によって、結果を代入します。

```
値 = (true or false) ? (true のときの値) : (false のときの値)
```

三項演算は最初は難しいですが、理解できると、のちに学習する if や switch などを省略することができます。

```
hp = 10;
var result = (hp <= 20) ? 'ピンチ！' : 'まだ平気';
```

結果

```
ピンチ！
```

練習

価格、個数、値引き、税率の変数を宣言して、値引き後の合計金額（税込）を計算してみましょう。

```
var total_price = 0;  //合計金額
var price = 300;    //価格
var amount = 5;     //個数
var discount = 100;  //値引き
var tax = 0.1;      //税率
```

もし、余分な数字がついてしまった場合は、toFixed() を使ってみましょう。

```
total_price.toFixed();
```