JavaScript

「console.log("〇〇");」というコードを書くと、()の中の〇〇という文字がコンソールに

注意　シングルクォーテーション（'）かダブルクォーテーション（"）で囲む必要がある

両方が使える。

なかったら、同じ変数名を呼ばれる可能です。

文末にセミコロンをつける

文頭に「//」がある行はコメントになる.

コメントは出力されない、コードの中にメモを残しことができる。

script.js
コード console.log("Hello World"); 結果 Hello World

数値と計算

数値は文字列と違いクォーテーションで囲まない。

結果を出力される。

script.js			
コード			
console.log(1);		**クォーテーションを使ったら**	**＊**
console.log(5+2);	足し算	console.log(”5+2”);	/
console.log(5-2);	引き算	console.log(”5”+”2”);	
console.log(10/2);	割り算		
console.log(9/2);			
console.log(9%2);		余り分しか表示されません	**%**
console.log(3*2);			
結果		結果	
1			
7		5+2	文字列と解釈されそのまま出

3 5 4.5 1 6	52

文字列の連結

プラスの記号を使う。

クォーテーションが必要。

script.js

コード

```
console.log("Hello”＋” World");
console.log("H”＋"C”＋” W");
```

結果

```
Hello World
HC W
```

変数

変数は、データ（値）の入れ物（箱）です。

箱についている名前が「変数名」であり、箱の中に実際の値（文字列や数値など）が

変数は「**let 変数名 = 値**」として定義します。

プログラミングの「=」は「等しい」という意味ではなく、「右辺を左辺に代入する」と

「let」は「これから変数を定義します」という宣言で、その後ろに変数名を書き、値を代

注意　変数を使う原因

1 同じ値を振り返し使える

2 変更に対応しやすい

3 値の意味が分かりやすい

書き方

1 英単語で書く

2 2語以上の場合は大文字で区

× 数字開始

× ローマ字で書く日本語

× 日本語

script.js

コード

```
let name = "cham";
```

script.js

コード

```
let name = "cham";
```

```
console.log(name);
console.log("name");          注意

結果
cham

name      そのまま出力される
```

```
console.log(name + " lala");
let number = 10;
console.log(number + 6);

結果
cham lala
```

16

script.js

コード

```
let text = "my friend";
console.log("LyLy, " + text);
console.log("Hoa, " + text);
console.log("Lan, " + text);
```

結果

```
LyLy, myfriend
Hoa, myfriend
Lan, myfriend
```

変数の更新

一度値を代入した変数に、その後再び値を代入すると、後に代入した値で変数の中身が上
定義する時と違って「**let**」**は必要なく**、「**変数名 = 新しい値**」と書けば値が変更されま

script.js

コード

```
let text = "my friend";
console.log(text);
text = "My LyLy"
console.log(text);
```

プログラムの実行順

結果

```
myfriend
My LyLy
```

上書きした変数の値が出力される。

プロク
後で代
す。

l みh

変数自身を更新する

script.js

コード

```
let number = 3;
console.log(number);

number = (number + 5);
console.log(number);

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

結果

```
3
8                    3+5
21                   8+8+5
```

省略した書き方

script.js	**script.js**
基本	省略
x = x + 10	x += 10
x = x - 10	x -= 10
x = x * 10	x *= 10
x = x / 10	x /= 10
x = x % 10	x %= 10

定数

定数はletの代わりに**const**を用いて定義します。

定数は値を更新することはできません。

script.js

コード

```
const name = "Cham";
console.log(name);
```

結果

Cham　　**更新できない**

let　　変更できる

const　　変更できない

var

いんでんと

テンプレートリテラル

文字列や定数の連結方、同じ＋

バッククォートが必要

script.js

コード

```
const name = "Cham";
const age = 20;
console.log(`Hi, ${name}! I am ${age}.`);
```

結果

Hi, Cham! I am 20.

vs　c(

条件分岐

じょうけんぶんき

プログラミングを学んでいると「ある条件が成り立つときだけある処理を行う」という場
このようなプログラムを条件分岐と言います。

if（条件式）｛処理｝；

script.js

コード

```
const  b  = 12;
if (b > 10){console.log("b は10 よりおおきいです");}
```

結果

b は10よりおおきいです

インデントとは日本語で「**字下げ**」を意味します。
きれいにインデントするとコードが見やすくなります。
下図のようにインデントを入れることでifの処理がどこからか一目でわかります。
tabキーを押すと、インデントすることができます。

script.js

コード

```
const  b  = 12;
if (b > 10){
   console.log("b は10 よりおおきいです”);
}
```

結果

b は10よりおおきいです

条件式の出力

script.js

コード

```
const  b  = 12;
console.log(b >10);
```

結果

true

script.js

コード

```
const  b  = 9;
console.log(b >10);
```

結果

false

大小を比べる演算子

大小を比べる	
a < b	a はbより小さい
a <= b	aの方が小さい または等しい
a > b	a はbより大きい
a >= b	aの方が大きい

script.js

```
const  b  = 9;

console.log(b < 10);
                true

console.log(b <= 9);
                true
console.log(b > 10);
```

または等しい	false

等価演算子

等しいかを比べる	script.js

a == b　　a とbが等しい
 	
a != b　　a とbが異なる
	const b = 9;
	
	console.log(b == 9);
	true

厳密に等しいかを比べる	
a === b　a とbが厳密に等しい a !== b　a とbが厳密に異なる	const name = "Cham"; console.log(name!="Cham"); false

注意　もし==を使ったら、数字と文字列が区別できなく、プログラムを全部呼べる。
プログラムをもっと正しく呼びたいと、===を使う。

» **厳密等価演算子**

script.js	script.js

const b = 9;
 	
console.log(b == 9);
true
 	
console.log(b == "9");
true
	const b = 9;
	
	console.log(b === 9);
	true
	
	console.log(b === "9");
	false

else

もし〇〇なら●●を行う、そうでなければ■■を行う

書き方

```
if(条件式)　{
　　条件が「true」の時の処理
}else{
　　条件が「false」の時の処理
}
```

if を使ったら、複数のif文が必要。
elseを使ったら、1つの条件分岐で同じことを実現できる。

script.js

```
const number = 9;
if(number >10){
  console.log("numberは10よりおおきいです”);
}
if(number ＜＝10){
  console.log("numberは10以下です");
}
```

script.js

```
const number = 9;
if(number>10){
  console.log("numberは10
}else{
  console.log("numberは10
}
```

else if 条件を追加する

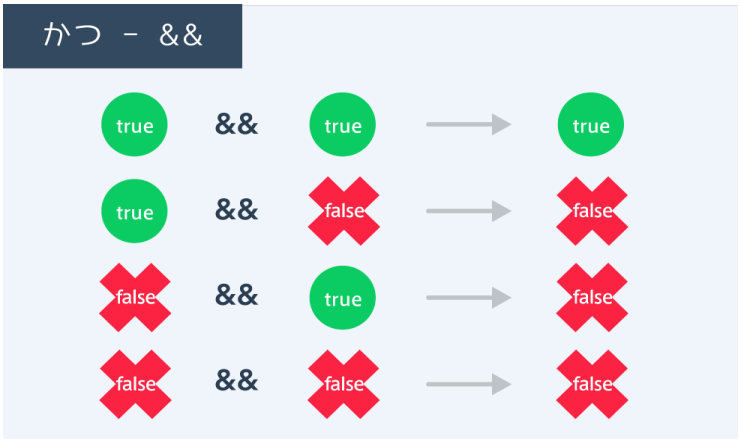
```
if(条件式1）｛
    条件式１が「true」の時の処理
}else if(条件式２）｛
    条件式１が「false」、条件式2が「true」の時の処理
}else{
     どちらの条件式も「false」の時の処理
}
```

script.js

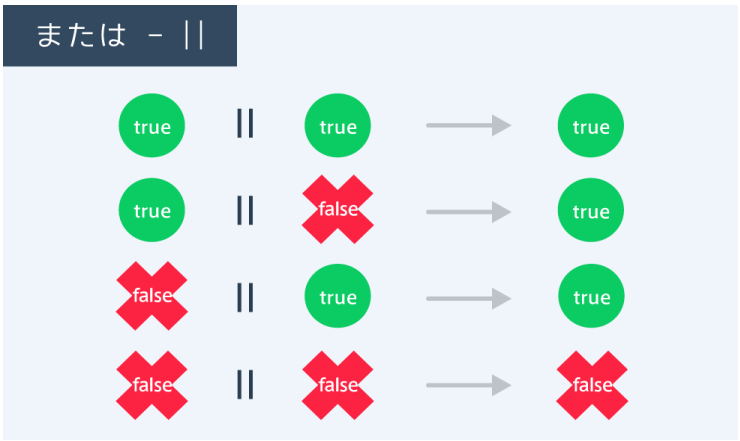
```
const number = 9;
if (number>10){
  console.log("numberは10よりおおきいです”);
} else if(number > 5){
  console.log("numberは5より大きいです");
} else {
  console.log("numberは5以下です”);
}
```

結果
numberは5以下です

かつ 「&&」 「条件1 && 条件2」は「条件1かつ条件2」という意味で、複数の条件が



または 「||」 「条件1 || 条件2」は「条件1または条件2」という意味です。この場合は



script.js

```
const number = 31;
if (number >= 10 && number < 100){
   console.log("numberは2桁です”);
}
```

結果

numberは2桁です

switch文 if文の役割が同じですが、バグしないように使います。

書き方

条件の値：変数の定義

default:　どのcaseにも合致

```
switch(条件の値){
  case 値 1
    「条件の値」が「値１」と等しい時の処理
  break;
  case 値 2
    「条件の値」が「値2」と等しい時の処理
  break;
…
  default:
    処理
    break;
}
```

script.js

```
const color = "赤";
switch(color){
  case "赤":
    console.log("ストップ！");
    break;
  case "黄":
    console.log("要注意");
    break;
…
  default:
    console.log("colorの値が正しくありません");
    break;
}
```

結果

ストップ！

注意点

script

```
const
switch
  cas
    (
  cas
    (
}
```

結果

スト

要注意

:出力される。

アスタリスク
スラッシュ

パーセント

力される

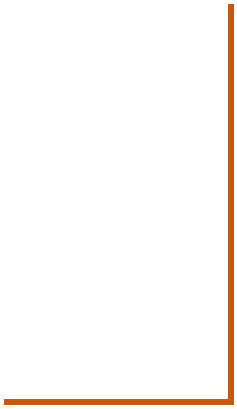
が入っています。

いう意味です。
も入します。

アグルリズム

切る

1 name chamu チャム



に書きされます。
す。

ラムは上から順に実行されるので、
入された値で変数の中身が**更新され**ま

x = number

x = x + 10

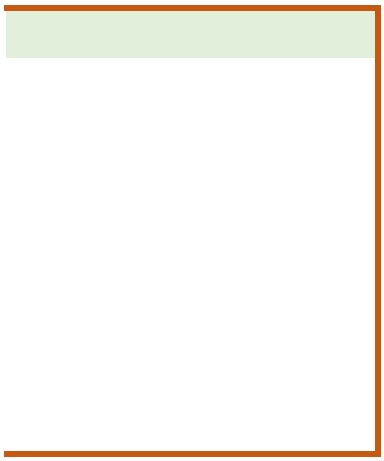
number = number + 10

string　　文字列
integer　　整数
boolean　　真偽値

intent

画面が出てきます。

インデント
tab キーでインデント



、複数の条件のうち1つでもtrueならtrueになります。