TypeScript

入門2

Getting Started

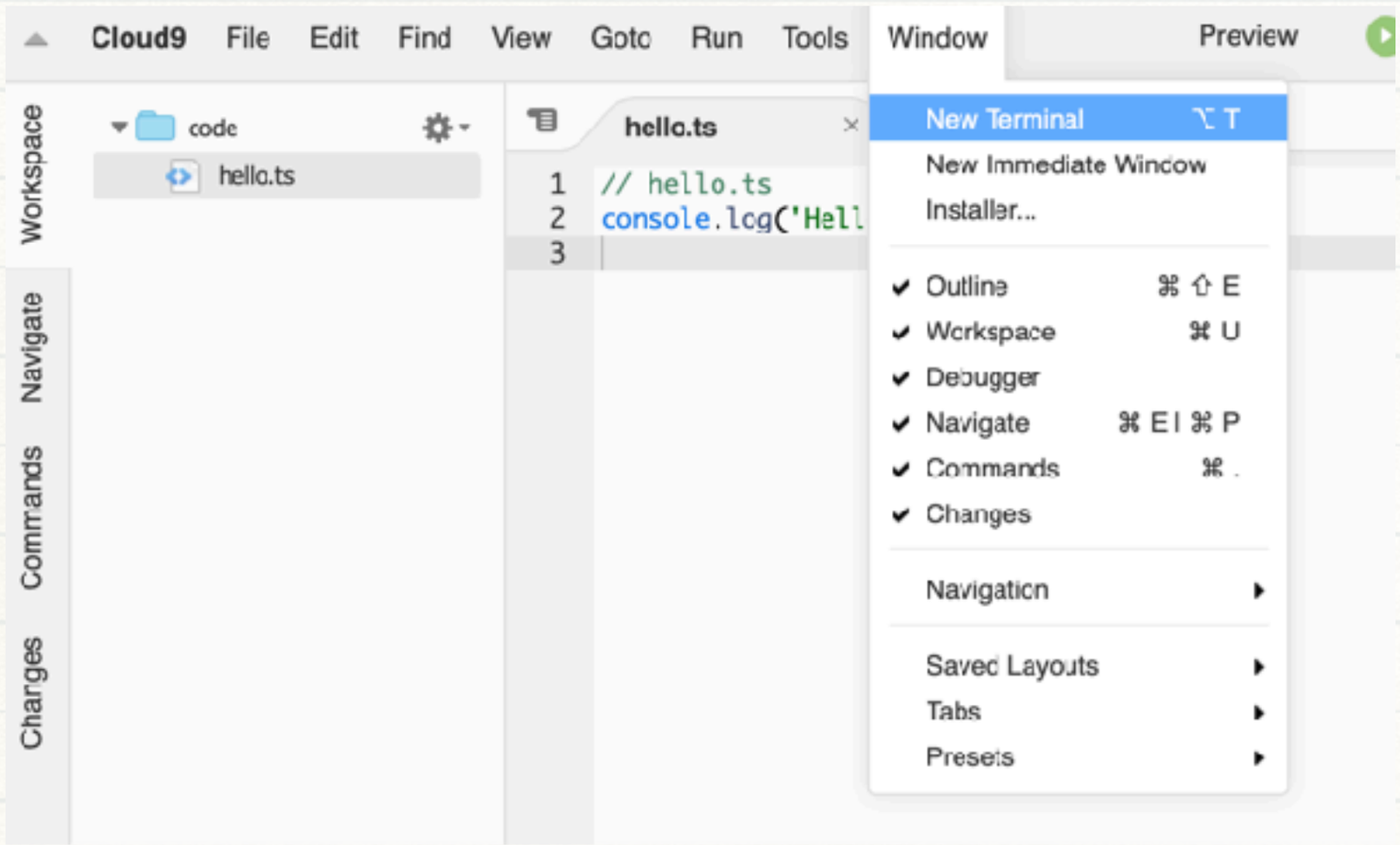
Hello, world!

Hello, world!

hello.ts
拡張子は .ts

```
// hello.ts
console.log('Hello, world!');
```

Window → New Terminal
実行してください



deno

10 Things I Regret About Node.js - Ryan Dahl

```
deno run filename.ts
    --help
    --reload
```

```
$ deno run hello.ts
Hello, world!
```

```
ls
pwd
cd

mkdir
rmdir
cp
rm
```

```
$ ls
$ pwd
$ cd
$ pwd

$ cd /code
```

```
zip -r file.zip dir
```

Lesson1

function

function

```
function 関数名(引数: 型): 型 {
    ...
    return ...
}
```

```
// ex1.ts
function square(n: number): number {
  return n * n;
}

console.log(square(2));
console.log(square('3'));
console.log(square('a'));
```

```
$ deno check ex1.ts
Check file:///Users/shingo1551/Documents/github/course/ts2/lesson1/ex1.ts
error: TS2345 [ERROR]: Argument of type 'string' is not assignable to parameter of type 'number'.
console.log(square('3'));
                   ~~~
    at file:///Users/shingo1551/Documents/github/course/ts2/lesson1/ex1.ts:7:20

TS2345 [ERROR]: Argument of type 'string' is not assignable to parameter of type 'number'.
console.log(square('a'));
                   ~~~
    at file:///Users/shingo1551/Documents/github/course/ts2/lesson1/ex1.ts:8:20

Found 2 errors.
```

jsでも実行してみましょう

function 型推論、アロー関数

関数の戻り値も、型推論

```
// ex2.ts
function square(n: number) {
  return n * n;
}

console.log(square(2));
```

```
$ deno run ex2.ts
4
```

アロー関数

```
// ex3.ts
const square1 = (n: number) => {
  return n * n;
};

const square2 = (n: number) => n * n;

console.log(square1(2));
console.log(square2(2));
```

```
$ deno run ex3.ts
4
4
```

function 引数

option?の引数

```
// ex4.ts
function foo(bar: number, bas?: string) {
  console.log(bar, bas);
}

foo(123);
foo(123, 'world');
```

```
$ deno run ex4.ts
123 undefined
123 world
```

引数のdefault値

```
// ex5.ts
function foo(bar: number, bas = 'hello') {
  console.log(bar, bas);
}

foo(123);
foo(123, 'world');
```

```
$ deno run ex5.ts
123 hello
123 world
```

function 可変個引数

可変個引数

```ts
// ex6.ts
function sum(name: string, ...v: number[]) {
  let sum = 0;
  for (const x of v)
    sum += x;

  console.log(name, sum);
}

sum('sum:', 2, 4, 6, 8);
```

```
$ deno run ex6.ts
sum: 20
```

Lesson2

string

template

テンプレート文字

`...`

```
// ex1.ts
const a = 'Hello';
const b = 'World!';

const s = `
<html>
  <body>
    <h1>${a}, <span>${b}</span></h1>
    <h1>${a.toUpperCase()}, ${b.toUpperCase()}</h1>
  </body>
</html>
`;

console.log(s);
```

```
$ deno run ex1.ts

<html>
  <body>
    <h1>Hello, <span>World!</span></h1>
    <h1>HELLO, WORLD!</h1>
  </body>
</html>
```

raw, length

String.raw`...`

length

```
// ex2.ts
const path = String.raw`C:\Development\profile\aboutme.html`;

console.log(path, path.length);
```

```
 deno run ex2.ts
C:\Development\profile\aboutme.html 35
```

replace, replaceAll

```
// ex3.ts
const p =
  'The quick brown fox jumps over the lazy dog. If the dog reacted, was it really lazy?';

console.log(p.replace('dog', 'monkey'));
console.log(p.replaceAll('dog', 'monkey'));
```

replace

replaceAll

```
$ deno run ex3.ts
The quick brown fox jumps over the lazy monkey. If the dog reacted, was it really lazy?
The quick brown fox jumps over the lazy monkey. If the monkey reacted, was it really lazy?
```

slice

slice

```
// ex4.ts
const log = console.log;
const str = 'The quick brown fox jumps over the lazy dog.';

log(str.length);
log(str.slice(31));
log(str.slice(4, 19));
log(str.slice(-4));
log(str.slice(-9, -5));
```

```
$ deno run ex4.ts
44
the lazy dog.
quick brown fox
dog.
lazy
```

split

split

```
// ex5.ts
const obama = `
My fellow citizens:

I stand here today humbled by the task before us, grateful for the trust
you have bestowed, mindful of the sacrifices borne by our ancestors.
I thank President Bush for his service to our nation, as well as the
generosity and cooperation he has shown throughout this transition.
`;

console.log(obama);

const lines = obama.split('\n');
console.log(lines);
```

```
$ deno run ex5.ts

My fellow citizens:

I stand here today humbled by the task before us, grateful for the trust
you have bestowed, mindful of the sacrifices borne by our ancestors.
I thank President Bush for his service to our nation, as well as the
generosity and cooperation he has shown throughout this transition.

[
  "",
  "My fellow citizens:",
  "",
  "I stand here today humbled by the task before us, grateful for the trust",
  "you have bestowed, mindful of the sacrifices borne by our ancestors.",
  "I thank President Bush for his service to our nation, as well as the",
  "generosity and cooperation he has shown throughout this transition.",
  ""
]
```

trim

trim

```ts
// ex6.ts
const greeting = '   Hello world!   ';

console.log(greeting);
console.log(greeting.trim());
```

```
$ deno run ex6.ts
  Hello world!
Hello world!
```

JSON.stringify

JSON.stringify

```ts
// ex7.ts
const obj = {
  name: 'TARO YAMADA',
  _credit: '36666666666660',
  expire: '01/25',
  _security: 1234,
};

console.log(JSON.stringify(obj));
console.log(JSON.stringify(obj, null, '  '));

console.log(JSON.stringify(obj, ['name', 'expire']));
console.log(JSON.stringify(obj, (k, v) => (k.startsWith('_') ? undefined : v)));
```

```
$ deno run ex7.ts
{"name":"TARO YAMADA","_credit":"36666666666660","expire":"01/25","_security":1234}
{
  "name": "TARO YAMADA",
  "_credit": "36666666666660",
  "expire": "01/25",
  "_security": 1234
}
{"name":"TARO YAMADA","expire":"01/25"}
{"name":"TARO YAMADA","expire":"01/25"}
```

JSON.parse

JSON.parse

```ts
// ex8.ts
const s = `
{
  "name": "TARO YAMADA",
  "_credit": "366666666666660",
  "expire": "01/25",
  "_security": 1234
}`;

console.log(JSON.parse(s));

console.log(JSON.parse(s, (k, v) => (k.startsWith('_') ? undefined : v)));
```

```
$ deno run ex8.ts
{ name: "TARO YAMADA", _credit: "366666666666660", expire: "01/25", _security: 1234 }
{ name: "TARO YAMADA", expire: "01/25" }
```

Lesson3

Array

length, push, pop, reverse

length
0 から length-1 まで

push

pop

reverse

```
// ex1.ts
const fruits = ['Apple', 'Banana', 'Mango'];
console.log(fruits.length);
console.log(fruits[0], fruits[fruits.length - 1]);

const length = fruits.push('Orange');
console.log(length, fruits);

const last = fruits.pop();
console.log(last, fruits);

fruits.reverse();
console.log(fruits);
```

```
$ deno run ex1.ts
3
Apple Mango
4 [ "Apple", "Banana", "Mango", "Orange" ]
Orange [ "Apple", "Banana", "Mango" ]
[ "Mango", "Banana", "Apple" ]
```

shift, unshift

shift

unshift

```ts
// ex2.ts
const fruits = ['Apple', 'Banana', 'Mango'];
const first = fruits.shift();
console.log(first, fruits);

const newLength = fruits.unshift('Strawberry');
console.log(newLength, fruits);

fruits[9] = 'Kiwi';
console.log(fruits.length, fruits);
```

```
$ deno run ex2.ts
Apple [ "Banana", "Mango" ]
3 [ "Strawberry", "Banana", "Mango" ]
10 [ "Strawberry", "Banana", "Mango", <6 empty items>, "Kiwi" ]
```

Array(length), fill, flat

Array(length)

fill

flat

```
// ex3.ts
const array1 = Array(8);
console.log(array1);

array1.fill(10);
console.log(array1);

const array2 = [0, 1, 2, [3, 4]];
console.log(array2.flat());
```

```
$ deno run ex3.ts
[ <8 empty items> ]
[
  10, 10, 10, 10,
  10, 10, 10, 10
]
[ 0, 1, 2, 3, 4 ]
```

includes, indexOf, join

includes

indexOf

join

```
// ex4.ts
const array3 = [1, 2, 3];
console.log(array3.includes(2));

const pets = ['cat', 'dog', 'bat'];
console.log(pets.includes('cat'));
console.log(pets.includes('ant'));
console.log(pets.indexOf('cat'));
console.log(pets.indexOf('ant'));

console.log(pets.join(', '));
```

```
$ deno run ex4.ts
true
true
false
0
-1
cat, dog, bat
```

concat, [...a, ...b]

concat
[...a, ...b]

```
// ex5.ts
const fruits1 = ['Apple', 'Banana'];
const fruits2 = ['Mango', 'Strawberry'];

const fruits3 = fruits1.concat(fruits2);
const fruits4 = [...fruits1, ...fruits2];

console.log(fruits1, fruits2);
console.log(fruits3, fruits4);
```

```
$ deno run ex5.ts
[ "Apple", "Banana" ] [ "Mango", "Strawberry" ]
[ "Apple", "Banana", "Mango", "Strawberry" ] [ "Apple", "Banana", "Mango",
"Strawberry" ]
```

slice, splice

slice

splice

```
// ex6.ts
const animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];
console.log(animals.slice(2));
console.log(animals);

const removedItem = animals.splice(1, 2);
console.log(removedItem);
console.log(animals);

const months = ['Jan', 'March', 'April', 'May'];
months.splice(1, 0, 'Feb');
console.log(months);
```

```
 deno run ex6.ts
[ "camel", "duck", "elephant" ]
[ "ant", "bison", "camel", "duck", "elephant" ]
[ "bison", "camel" ]
[ "ant", "duck", "elephant" ]
[ "Jan", "Feb", "March", "April", "May" ]
```

sort

sort

```
// ex7.ts
const months = ['Jan', 'Feb', 'Mar', 'Apr', 'May'];
months.sort();
console.log(months);

const array = [1, 30, 4, 21, 100000];
array.sort();
console.log(array);

function cmp(i: number, j: number) {
    return i - j;
}
array.sort(cmp);
console.log(array);

array.sort((i, j) => j - i);
console.log(array);
```

型推論

```
$ deno run ex7.ts
[ "Apr", "Feb", "Jan", "Mar", "May" ]
[ 1, 100000, 21, 30, 4 ]
[ 1, 4, 21, 30, 100000 ]
[ 100000, 30, 21, 4, 1 ]
```

Question

以下の処理を追加してください

- nameでsort
- ageでsort

```ts
// ans1.ts
const persons = [
    { name: 'Yamada', age: 30 },
    { name: 'Suzuki', age: 20 },
    { name: 'Nakamura', age: 35 },
    { name: 'Tanaka', age: 40 },
];
```

```
$ deno run ans1.ts
[
  { name: "Nakamura", age: 35 },
  { name: "Suzuki", age: 20 },
  { name: "Tanaka", age: 40 },
  { name: "Yamada", age: 30 }
]
[
  { name: "Suzuki", age: 20 },
  { name: "Yamada", age: 30 },
  { name: "Nakamura", age: 35 },
  { name: "Tanaka", age: 40 }
]
```

Answer

```ts
// ans1.ts
const persons = [
    { name: 'Yamada', age: 30 },
    { name: 'Suzuki', age: 20 },
    { name: 'Nakamura', age: 35 },
    { name: 'Tanaka', age: 40 },
];

persons.sort((p1, p2) => p1.name < p2.name ? -1 : 1);
console.log(persons);

persons.sort((p1, p2) => p1.age - p2.age);
console.log(persons);
```

find, findIndex, forEach

find

findIndex

forEach

```
// ex8.ts
const array = [5, 12, 8, 130, 44];
console.log(array.find(element => element > 10));

console.log(array.findIndex(element => element > 13));

array.forEach((element) => console.log(element));
```

```
$ deno run ex8.ts
12
3
5
12
8
130
44
```

filer, map

filter

map

```
// ex9.ts
const words = ['spray', 'limit', 'exuberant', 'destruction', 'present'];
const result = words.filter((word) => word.length > 6);
console.log(result);

const array1 = [1, 4, 9, 16];
const array2 = array1.map((x) => x * 2);
console.log(array2);
```

```
$ deno run ex9.ts
[ "exuberant", "destruction", "present" ]
[ 2, 8, 18, 32 ]
```

reduce

reduce

```
// ex10.ts
const array = [1, 2, 3, 4];
let sum = array.reduce((acc, val) => acc + val);
console.log(sum);

sum = array.reduce((acc, val, i) => {
    console.log(i, ':', acc, val);
    return acc + val;
});
console.log(sum);
```

```
$ deno run ex10.ts
10
1 : 1 2
2 : 3 3
3 : 6 4
10
```

Lesson4

fmt

printf, sprintf (Standard Library)

C言語とほぼ同じ printf が使用できる

T: typeof
t: boolean

数字: 文字数

d: 整数

-: 左寄せ

0: 0埋め

x, X: 16進数

o: 8進数

b: 2進数

s: string
j: json

e, E: exponent
f, F: float
g, G: f or e
v: default

```ts
// ex1.ts
import { printf, sprintf } from 'https://deno.land/std/fmt/printf.ts';

const b = true;
printf('%T %t\n', b, b);

const n = 42;
printf('%T: %d\n', n, n);
printf('%d (%6d) (%-6d) (%06d)\n', n, n, n, n);
printf('%x %o %b\n', n, n, n);

const s = 'abc';
printf('%s (%10s) (%-10s)\n', s, s, s);

const p = { name: 'Tanaka', age: 42 };
printf('%j\n', p);

printf('\\%%\n');

const pi = Math.PI;
const s2 = sprintf('%e %f %g %v (%6.2f) (%-6.2f)', pi, pi, pi, pi, pi, pi);
console.log(s2);
```

https://deno.land/std@0.100.0/fmt

```
$ deno run ex1.ts
boolean true
number: 42
42 (    42) (42    ) (000042)
2a 52 101010
abc (       abc) (abc       )
{"name":"Tanaka","age":42}
\%
3.141593e+00 3.141593 3.14159 3.141592653589793 (  3.14) (3.14  )
```

Lesson5

Object指向

JavaScript

JavaScriptのclassの使い方

class

this.name, this.age

new

p.name, p.age

p.print()

ex1.tsでも実行しましょう

```
// ex1.js
class Person {
  print() {
    console.log(`name: ${this.name}, age: ${this.age}`);
  }
}

const p = new Person();
p.name = 'tanaka';
p.age = 35;
p.print();

p.name = 45;
p.age = 'suzuki';
p.print();
```

```
$ deno run ex1.js
name: tanaka, age: 35
name: 45, age: suzuki
```

TypeScript

Javaそっくり

```
// ex2.ts
class Person {
  name: string;
  age: number;

  constructor(name: string, age: number) {
    this.name = name;
    this.age = age;
  }

  print() {
    console.log(`name: ${this.name}, age: ${this.age}`);
  }
}

const p = new Person('tanaka', 35);
p.print();
```

```
$ deno run ex2.ts
name: tanaka, age: 35
```

constructor

~~public: name: string;~~

~~public: age: number;~~

~~this.name = name;~~

~~this.age = age;~~

```
// ex3.ts
class Person {
  constructor(public name = 'tanaka', public age = 20) {}

  print() {
    console.log(`name: ${this.name}, age: ${this.age}`);
  }
}

const p1 = new Person();
p1.print();

const p2 = new Person('suzuki', 35);
p2.print();
```

```
$ deno run ex3.ts
name: tanaka, age: 20
name: suzuki, age: 35
```

private, getter, setter

private

get

set

```
// ex4.ts
class Person {
  constructor(private _name = 'tanaka', private age = 20) {}

  get name() {
    return this._name;
  }

  set name(name: string) {
    this._name = name;
  }

  print() {
    console.log(`name: ${this._name}, age: ${this.age}`);
  }
}

const p = new Person();
p.print();

p.name = 'suzuki';
console.log(p.name);

~~p.age = 45;~~
```

```
$ deno check ex4.ts
Check file:///Users/shingo1551/Documents/github/course/ts2/lesson5/ex4.ts
error: TS2341 [ERROR]: Property 'age' is private and only accessible within class 'Person'.
p.age = 45;
  ~~~
    at file:///Users/shingo1551/Documents/github/course/ts2/lesson5/ex4.ts:24:3
```

継承

extends

super()

super.print()

```
// ex5.ts
class Person {
  constructor(public name: string, public age: number) {}

  print() {
    console.log(`name: ${this.name}, age: ${this.age}`);
  }
}

class Employee extends Person {
  constructor(name: string, age: number, public salary: number) {
    super(name, age);
  }

  print() {
    super.print();
    console.log(`salary: ${this.salary}`);
  }
}

function print(p: Person) {
  p.print();
}

const p1 = new Person('tanaka', 30);
print(p1);

const p2 = new Employee('yamada', 25, 300000);
print(p2);
```

```
$ deno run ex5.ts
name: tanaka, age: 30
name: yamada, age: 25
salary: 300000
```

Lesson6

export, import

export, import

export

```
// ex1.ts
export const PI = 3.14;

export const square = (n: number) => n * n;

export class Person {
  constructor(public name: string, public age: number) {}

  print() {
    console.log(this.name, this.age);
  }
}
```

import

denoでは拡張子 .ts が必要

```
// ex2.ts
import { PI, square, Person } from './ex1.ts';

console.log(PI);

console.log(square(3));

const p = new Person('tanaka', 10);
p.print();
```

```
$ deno run ex2.ts
3.14
9
tanaka 10
```

import * as

* as ...

```
// ex3.ts
import * as ex1 from './ex1.ts';

console.log(ex1.PI);

console.log(ex1.square(3));

const p = new ex1.Person('tanaka', 10);
p.print();
```

```
$ deno run ex3.ts
3.14
9
tanaka 10
```

Appendix

RegExp, string

RegExpで正規表現が使用できます

/... /のみでもRegExp

```
const re1 = new RegExp('ab+c');
const re2 = /ab+c/;
```

RegExp

- test
- exec

string

- match
- matchAll
- replace
- replaceALl
- search
- split

で使用できます

```
// ex1.ts
const log = (re: RegExp, s: string) => console.log(re.test(s), re, s);

//
const re1 = /do+g/;
log(re1, 'hounddog');
log(re1, 'badge');
log(re1, 'hotdog');
log(re1, 'doofus');
log(re1, 'doogie');
log(re1, 'Doogie');

const re2 = /car*t/;
log(re2, 'carted');
log(re2, 'carrot');
log(re2, 'cat');
log(re2, 'carl');
```

```
$ deno run ex1.ts
true /do+g/ hounddog
false /do+g/ badge
true /do+g/ hotdog
false /do+g/ doofus
true /do+g/ doogie
false /do+g/ Doogie
true /car*t/ carted
false /car*t/ carrot
true /car*t/ cat
false /car*t/ carl
```

早見表

文字クラス

文字や数字の区別など、文字の種類を区別します

\	\n	\w	[\b]
.	\r	\W	
\cX	\s	\0	
\d	\S	\xhh	
\D	\t	\uhhhh	
\f	\v	\uhhhhh	

アサーション

行や単語の始まりや終わりを示す境界や、(先読み、後読み、条件式を含む)何らかの方法で一致できることを示す、その他のパターンが含まれます

^	\b
$	\B
x(?=y)	
x(?!y)¥(?<=y)x	
(?<!y)x	

グループと範囲

式にある文字のグループと範囲を示します

(x)	[xyz]
(?:x)	[^xyz]
(?<Name>x)	*Number*
x|y	

数量

一致させる文字や式の数を示します

*	x{*n*,}
+	x{*n*,*m*}
?	
x{*n*}	

Unicodeプロパティエスケープ

Unicodeプロパティエスケープ

大文字と小文字、数学記号、句読点など、Unicode文字のプロパティに基づき区別します

```
\p{UnicodeProperty}
\P{UnicodeProperty}
```

```
// ex9.ts
const sentence = 'A ticket to 大阪 costs ¥2000 👌.';

const reg = /\p{Emoji_Presentation}/gu;
console.log(sentence.match(reg));
```

```
$ deno run ex9.ts
[ "👌" ]
```

Link

TypeScript	https://www.typescriptlang.org
deno	https://deno.land
10 Things I Regret About Node.js	https://youtu.be/M3BM9TB-8yA
Visual Studio Code (VScode)	https://code.visualstudio.com