Luyện Tập number, string và for

kiểu dữ liệu number

Kiểu dữ liệu number đại diện cho việc tính toán trong ngôn ngữ lập trình, nó rất gần gũi với toán học của chúng ta, nếu bạn yêu toán thì rất tuyệt vời còn nếu bạn không yêu toán thì cũng không cần quá lo lắng vì bạn chỉ cần các kiến thức rất cơ bản về toán để chinh phục lập trình mà thôi

Các loại number khác nhau

Trong một số ngôn ngữ khác thì kiểu dữ liệu number lại được chia ra thành các kiểu dữ liệu khác nhau:

1. interger: Dùng để lưu trữ các số nguyên, bao gồm nguyên dương và nguyên âm
2. float: Dùng để lưu trữ các số có phần thập phân với độ chính xác thấp hơn double
3. double: Dùng để lưu trữ các số có phần thập phân với độ chính xác cao hơn float

> Ngôn ngữ javascript chỉ có 1 kiểu dữ liệu chung là number

```
console.log(typeof 10);
console.log(typeof 10.5);
```

Converting to number data types

Bạn sẽ gặp các trường hợp kiểu dữ liệu không phải number như người dùng nhập vào input chẳng hạn.

Bạn sau đấy mong muốn convert nó sang kiểu dữ liệu number để tính toán, nên sử dụng: **Number() constructor**

```
let myNumber = "74";
myNumber += 3;
```

yêu cầu: đọc hiểu các hàm khác để vận dụng khi cần thiết

Cụ thể: parseInt, dấu + phía trước value... **theo link**: https://www.freecodecamp.org/news/how-to-convert-a-string-to-a-number-in-javascript/

Kiểu dữ liệu string

Kiểu dữ liệu string được dùng để lưu trữ các thông tin dưới dạng **text**, có thể là tên người dùng, tiêu đề bài viết, mô tả bài viết

Declaring strings

Khi khai báo chuỗi có đầy đủ nháy đơn('') hoặc nháy kép("") hoặc backticks(``). Nếu không bạn sẽ nhận được lỗi không mong muốn. Dưới đây là một số lỗi khi khai báo chuỗi:

```
const badString1 = This is a test;
const badString2 = 'This is a test;
const badString3 = This is a test';
```

Single quotes, double quotes, and backticks

string được tạo từ backticks có một số khác biệt như sau:

1. you can embed JavaScript in them
2. you can declare template literals over multiple lines

Embedding JavaScript

You can use `${}` only with template literals, not normal strings. You can concatenate normal strings using the + operator:

```
const greeting = "Hello";
const name = "Chris";
console.log(greeting + ", " + name); // "Hello, Chris
```

However, template literals usually give you more readable code:

```
const greeting = "Hello";
const name = "Chris";
console.log(`${greeting}, ${name}`); // "Hello, Chris"
```

Including expressions in strings

You can include JavaScript expressions in template literals, as well as just variables, and the results will be included in the result

```
    const song = "Fight the Youth";
    const score = 9;
    const highestScore = 10;
    const output = `I like the song ${song}. I gave it a score of ${
    (score / highestScore) * 100
    }%.`;
    console.log(output); // "I like the song Fight the Youth. I gave it a score of
90%."
```

Multiline strings

Template literals respect the line breaks in the source code, so you can write strings that span multiple lines like this:

```
    const newline = `One day you finally knew
    what you had to do, and began,`;
    console.log(newline);
```

To have the equivalent output using a normal string you'd have to include line break characters (\n) in the string:

```
    const newline = "One day you finally knew\nwhat you had to do, and began,";
    console.log(newline);
```

Including quotes in strings (kí tự đặc biệt)

Since we use quotes to indicate the start and end of strings, how can we include actual quotes in strings? We know that this won't work:

```
    const badQuotes = "She said "I think so!"";
```

One common option is to use one of the other characters to declare the string:

```
const goodQuotes1 = 'She said "I think so!"';
const goodQuotes2 = `She said "I'm not going in there!"`;
```

Another option is to escape the problem quotation mark. Escaping characters means that we do something to them to make sure they are recognized as text, not part of the code. In JavaScript, we do this by putting a backslash just before the character. Try this:

```
const bigmouth = 'I\'ve got no right to take my place…';
console.log(bigmouth);
```

Numbers vs. strings

What happens when we try to concatenate a string and a number? Let's try it in our console:

```
const name = "Front ";
const number = 242;
console.log(name + number); // "Front 242"
```

Tìm hiểu vòng lặp Loop

1. Bình thường vợ là người đi chợ nấu cơm, nhưng tuần này là ngày mùng 8/3 nên chồng đi chợ nấu cơm từ đầu tháng và hết 8/3 tặng vợ, vậy thì đây là công việc hàng ngày chồng làm:

1/3 -> đi chợ, nấu cơm 2/3 -> đi chợ, nấu cơm 3/3 -> đi chợ, nấu cơm 4/3 -> đi chợ, nấu cơm 5/3 -> đi chợ, nấu cơm 6/3 -> đi chợ, nấu cơm 7/3 -> đi chợ, nấu cơm 8/3 -> đi chợ, nấu cơm

-> nếu xem ngày là 1 biến trong lập trình và ngày thay đổi theo --> Nếu đưa vào mô hình lập trình với for anh sẽ làm như sau:

for

Cấu trúc vòng lặp for tổng quát như sau:

```
for (expression1; expression2; expression3) {
    statements
}
```

Bất kì biểu thức nào cũng có thể để trống, bạn có thể xem các cách biểu diễn for khác nhau như sau:

```
// vòng lặp đầy đủ biểu thức
for (var i = 1; i <= 8; i++) {
    console.log('i', i);
}

// vòng lặp bỏ điều kiện dừng
for (var i = 1; ; i++) {
    if(i > 8) {
        break;
    }
    console.log('i', i);
}

// vòng lặp bỏ cả 2
var i = 1;
for (;;) {
    if(i > 8) {
        break;
    }
    console.log('i', i);
    i++;
}
```

while

````
Tương tự for, cấu trúc vòng lặp while như sau:

```

    while (expression) {
        statements
    }
````
````

```
    var count = 1;
    while (count < 5) {
        console.log('count', count);
        count++;
    }

```

do while

Nó sẽ chạy 1 lần kể cả điều kiện sai vì nó chạy trước khi kiểm tra điều kiện

Cấu trúc do while như sau:

```

    do {
        statements
    } while (expression)

    var count = 11;
    do {
        console.log('count', count);
    } while (count < 10);

```