Students notes

Backend vs Frontend

BE example:

- https://jsonplaceholder.typicode.com/
- BE App returns something similar to this: https://jsonplaceholder.typicode.com/photos?_start=0&_limit=5

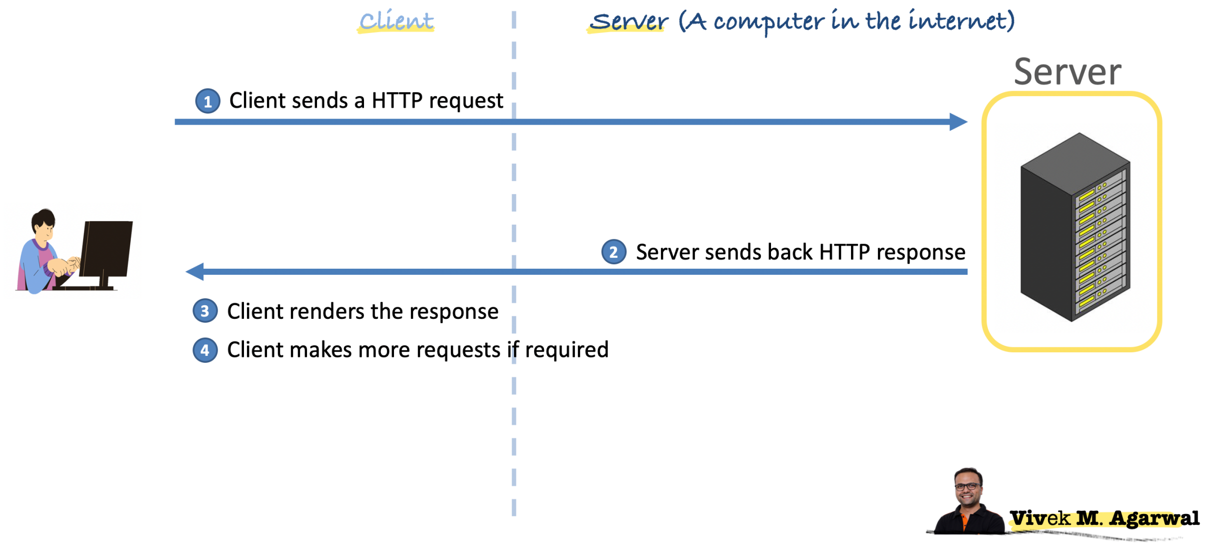
FE example:

- https://codepen.io/drupalastic/pen/rNpXKxN?editors=1010

🔑 Student Activity

- Read the documentation - https://jsonplaceholder.typicode.com/
- Try to load all the users in your browser [simple JSON object - only the code that comes from the backend server]

How web works Client/server architecture


Client
Server (A computer in the internet)
Server
1 Client sends a HTTP request
2 Server sends back HTTP response
3 Client renders the response
4 Client makes more requests if required
Vivek M. Agarwal

The client requests a service

- we open a browser (client)
- we hit a url for example: www.google.com (Uniform Resource Location)
  - the client sends a message to the server asks for a resource
  - Resources can be - web pages, images, video files, fonts, stylesheets
  - This message is formatted based on a protocol called `HTTP`
  - In other words, HTTP (Hypertext transfer protocol) is a standard structure (or protocol) that clients and servers use to communicate over internet.
  - With an `HTTP request`, the client communicates to the server, what it is looking for

The server provides the service

- The server listens to the message
- It figures out what the client is asking
- It sends a message back to the client
- This message is called an `HTTP response`

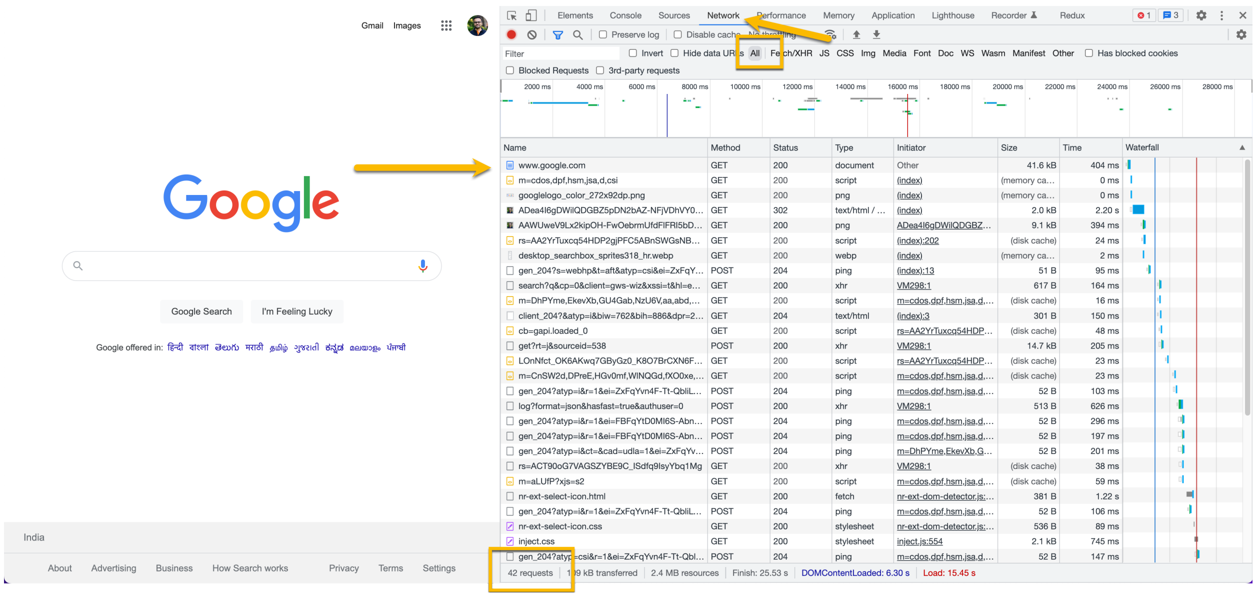
The client gets back a message

- for example, the response can be an html page

```
<!DOCTYPE html>
<html>
...
    <link rel="stylesheet" href="styles.css">
...
  <img class="footer-sub__logo" src="<https://unsplash.co
m/assets/core/logo-black-df2168ed0c378fa5506b1816e75eb379d
06cfcd0af01e07a2eb813ae9b5d7405.svg>">
...
</html>
```

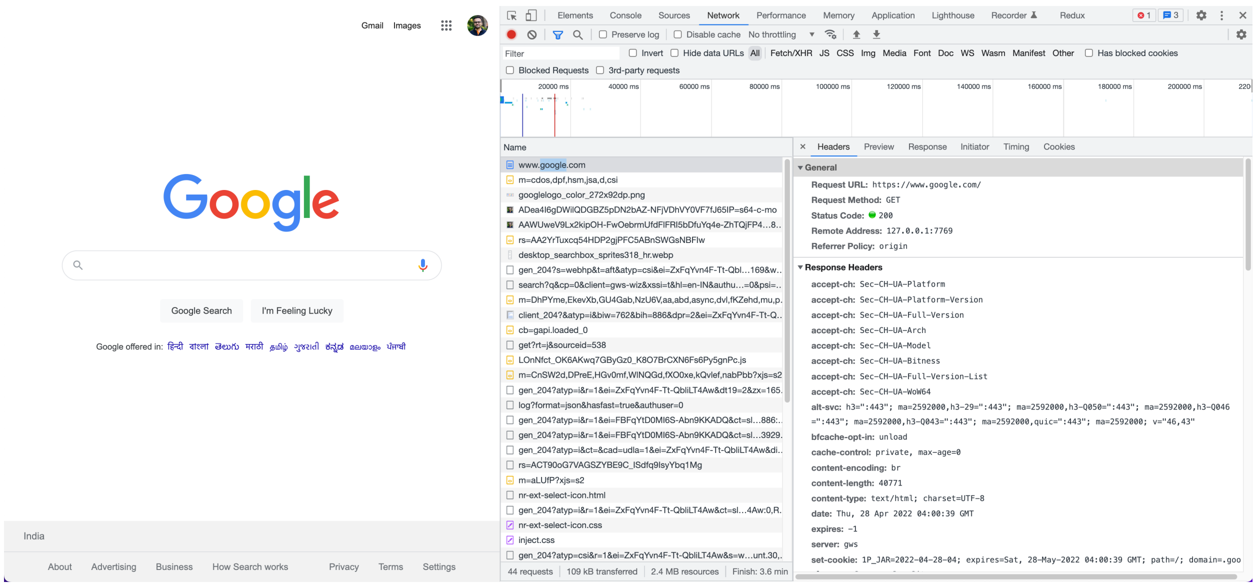
- browser constructs a DOM in case its an HTML document (Document Object Model)
- browser discovers references to other resources in the html document like images, stylesheets, font etc.
- For each resource the browser sends separate HTTP requests to same or other servers to fetch that resource
- These requests can be parallel
- Once the browser has all the necessary resources, it renders the HTML document (or displays it)

Request and Response in Action



Name	Method	Status	Type	Initiator	Size	Time
www.google.com	GET	200	document	Other	41.6 kB	404 ms
m=cdos,dpf,hsm,jsa,d,csi	GET	200	script	(index)	(memory ca…	0 ms
googlelogo_color_272x92dp.png	GET	200	png	(index)	(memory ca…	0 ms
ADea4I6gDWilQDGBZ5pDN2bAZ-NFjVDhVY0…	GET	302	text/html / …	(index)	2.0 kB	2.20 s
AAWUweV9Lx2kipOH-FwOebrmUfdFIFRI5bD…	GET	200	png	ADea4I6gDWilQDGBZ…	9.1 kB	394 ms
rs=AA2YrTuxcq54HDP2gjPFC5ABnSWGsNB…	GET	200	script	(index):202	(disk cache)	24 ms
desktop_searchbox_sprites318_hr.webp	GET	200	webp	(index)	(memory ca…	2 ms
gen_204?s=webhp&t=aft&atyp=csi&ei=ZxFqY…	POST	204	ping	(index):13	51 B	95 ms
search?q&cp=0&client=gws-wiz&xssi=t&hl=e…	GET	200	xhr	VM298:1	617 B	164 ms
m=DhPYme,EkevXb,GU4Gab,NzU6V,aa,abd,…	GET	200	script	m=cdos,dpf,hsm,jsa,d,…	(disk cache)	16 ms
client_204?&atyp=i&biw=762&bih=886&dpr=2…	GET	204	text/html	(index):3	301 B	150 ms
cb=gapi.loaded_0	GET	200	script	rs=AA2YrTuxcq54HDP…	(disk cache)	48 ms
get?rt=j&sourceid=538	POST	200	xhr	VM298:1	14.7 kB	205 ms
LOnNfct_OK6AKwq7GByGz0_K8O7BrCXN6F…	GET	200	script	rs=AA2YrTuxcq54HDP…	(disk cache)	23 ms
m=CnSW2d,DPreE,HGv0mf,WINQGd,fXO0xe,…	GET	200	script	m=cdos,dpf,hsm,jsa,d,…	(disk cache)	23 ms
gen_204?atyp=i&r=1&ei=ZxFqYvn4F-Tt-QbliL…	POST	204	ping	m=cdos,dpf,hsm,jsa,d,…	52 B	103 ms
log?format=json&hasfast=true&authuser=0	POST	200	xhr	VM298:1	513 B	626 ms
gen_204?atyp=i&r=1&ei=FBFqYtD0MI6S-Abn…	POST	204	ping	m=cdos,dpf,hsm,jsa,d,…	52 B	296 ms
gen_204?atyp=i&r=1&ei=FBFqYtD0MI6S-Abn…	POST	204	ping	m=cdos,dpf,hsm,jsa,d,…	52 B	197 ms
gen_204?atyp=i&ct=&cad=udla=1&ei=ZxFqYv…	POST	204	ping	m=DhPYme,EkevXb,G…	52 B	201 ms
rs=ACT90oG7VAGSZYBE9C_ISdfq9lsyYbq1Mg	GET	200	xhr	VM298:1	(disk cache)	38 ms
m=aLUfP?xjs=s2	GET	200	script	m=cdos,dpf,hsm,jsa,d,…	(disk cache)	59 ms
nr-ext-select-icon.html	GET	200	fetch	nr-ext-dom-detector.js:…	381 B	1.22 s
gen_204?atyp=i&r=1&ei=ZxFqYvn4F-Tt-QbliL…	POST	204	ping	m=cdos,dpf,hsm,jsa,d,…	52 B	106 ms
nr-ext-select-icon.css	GET	200	stylesheet	nr-ext-dom-detector.js:…	536 B	89 ms
inject.css	GET	200	stylesheet	inject.js:554	2.1 kB	745 ms
gen_204?atyp=csi&r=1&ei=ZxFqYvn4F-Tt-Qbl…	POST	204	ping	m=cdos,dpf,hsm,jsa,d,…	52 B	147 ms

42 requests | 109 kB transferred | 2.4 MB resources | Finish: 25.53 s | DOMContentLoaded: 6.30 s | Load: 15.45 s



Name
- www.google.com
- m=cdos,dpf,hsm,jsa,d,csi
- googlelogo_color_272x92dp.png
- ADea4I6gDWilQDGBZ5pDN2bAZ-NFjVDhVY0VF7fJ65Pf=s64-c-mo
- AAWUweV9Lx2kipOH-FwOebrmUfdFIFRI5bDfuYq4e-ZhTQjFP4…8..
- rs=AA2YrTuxcq54HDP2gjPFC5ABnSWGsNBFIw
- desktop_searchbox_sprites318_hr.webp
- gen_204?s=webhp&t=aft&atyp=csi&ei=ZxFqYvn4F-Tt-Qbl…169&w..
- search?q&cp=0&client=gws-wiz&xssi=t&hl=en-IN&authu…=0&psi=..
- m=DhPYme,EkevXb,GU4Gab,NzU6V,aa,abd,async,dvl,fKZehd,mu,p.
- client_204?&atyp=i&biw=762&bih=886&dpr=2&ei=ZxFqYvn4F-Tt-Q..
- cb=gapi.loaded_0
- get?rt=j&sourceid=538
- LOnNfct_OK6AKwq7GByGz0_K8O7BrCXN6Fs6Py5gnPc.js
- m=CnSW2d,DPreE,HGv0mf,WINQGd,fXO0xe,kQvlef,nabPbb?xjs=s2
- gen_204?atyp=i&r=1&ei=ZxFqYvn4F-Tt-QbliLT4Aw&dt19=2&zx=165.
- log?format=json&hasfast=true&authuser=0
- gen_204?atyp=i&r=1&ei=FBFqYtD0MI6S-Abn9KKADQ&ct=sl…886:..
- gen_204?atyp=i&r=1&ei=FBFqYtD0MI6S-Abn9KKADQ&ct=sl…3929.
- gen_204?atyp=i&ct=&cad=udla=1&ei=ZxFqYvn4F-Tt-QbliLT4Aw&di..
- rs=ACT90oG7VAGSZYBE9C_ISdfq9lsyYbq1Mg
- m=aLUfP?xjs=s2
- nr-ext-select-icon.html
- gen_204?atyp=i&r=1&ei=ZxFqYvn4F-Tt-QbliLT4Aw&ct=sl…4Aw:0,R.
- nr-ext-select-icon.css
- inject.css
- gen_204?atyp=csi&r=1&ei=ZxFqYvn4F-Tt-QbliLT4Aw&s=w…unt.30,.

44 requests | 109 kB transferred | 2.4 MB resources | Finish: 3.6 min

General

```
Request URL: https://www.google.com/
Request Method: GET
Status Code: 200
Remote Address: 127.0.0.1:7769
Referrer Policy: origin
```

Response Headers

```
accept-ch: Sec-CH-UA-Platform
accept-ch: Sec-CH-UA-Platform-Version
accept-ch: Sec-CH-UA-Full-Version
accept-ch: Sec-CH-UA-Arch
accept-ch: Sec-CH-UA-Model
accept-ch: Sec-CH-UA-Bitness
accept-ch: Sec-CH-UA-Full-Version-List
accept-ch: Sec-CH-UA-WoW64
alt-svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046
=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
bfcache-opt-in: unload
cache-control: private, max-age=0
content-encoding: br
content-length: 40771
content-type: text/html; charset=UTF-8
date: Thu, 28 Apr 2022 04:00:39 GMT
expires: -1
server: gws
set-cookie: 1P_JAR=2022-04-28-04; expires=Sat, 28-May-2022 04:00:39 GMT; path=/; domain=.goo
```

🔑 Student Activity

- Visit: https://www.amazon.in/Apple-iPhone-15-Plus-256/dp/B0CHWV3L2R/ref=sr_1_10?crid=3IYIGCZNHLQ98
- Inspect and change the price
- Where did you change the price - client or the server

Intro to JS

How does the computer work?

Computers function through a series of processes involving hardware and software. The primary components include the central processing unit (CPU), memory, storage, and input/output devices. When a computer is turned on, the CPU executes instructions from the operating system, applications, and user inputs, manipulating data stored in memory.

What is a computer program?

A computer program is a set of instructions written in a programming language to perform specific tasks. To create and execute these programs, developers use a coding environment. For JavaScript, the coding environment includes a JavaScript engine. In web development, browsers handle the JavaScript engine, and each browser has its own engine.

JavaScript Compilation Process

1. **Compiler/Parser:**
    - Checks each line of code for syntax errors.
    - Enforces rules; errors result in the code not running.
2. **Converting to Machine Code:**
    - If the code passes checks, it is converted to machine code.
3. **Different JavaScript Engines:**
    - Browsers use distinct engines (e.g., V8 in Chrome, SpiderMonkey in Firefox).
    - Mobile phones may have different JavaScript engines, affecting code execution speed.

   ![JavaScript Engines](insert link to images showing different engines)

What is a scripting language?

> *Scripting language is like a sub-category of programming languages which will connect one language to another which is crucial for developing websites. Difference between scripting and programming languages is the role of a compiler. Programming languages will be compiled in one go whereas scripting languages will be interpreted line by line.*

What is a framework in general?

> *Frameworks provide developers with the basic foundation necessary for applications. This saves developers the effort of starting any application from scratch.*

> ***Open flipkart and by the end of this unit, you can build the website like flipkart website.***

First we will discuss HTML and CSS, and then we will dive into the basics of JS. Finally, we'll combine all these topics and make a fully functional website.

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment
that runs and executes JavaScript code outside a web browser. Node.js lets developers
use JavaScript to write command-line tools and for server-side scripting. Consequently, Node.js represents a
`"JavaScript everywhere"` paradigm, unifying web
application development around a single programming language, rather than a

different
languages for server-side and client-side scripts.

Coding Platform

https://replit.com/

Running Javascript Code

```
node filename.js
```

Variables

What is a variable?

A variable is a storage container with an assigned name that holds data. It represents a value that can vary or change during program execution.

Purpose of Variables

Variables serve as storage locations with assigned names, holding data for future use. The syntax for declaring and using variables in JavaScript includes keywords like `let`, `const`, and `var`.

Variable Declaration Examples

```
// Using Var
var a;          // Declaring a variable.
a = 10;         // Assigning a value to the variable.
console.log(a); // Running the code.
```

```javascript
// Shorthand for declaration and assignment
var a = 10;
console.log(a);
```

Data Types

1. **Number:**
    - Represents numeric values.
2. **String:**
    - Represents textual data. Always enclosed in quotes (`""` or `''`).

    ```javascript
    var name = "gohan";
    console.log(name);
    ```

Checking Data Types

```javascript
var x = 12;
console.log(typeof(x)); // Output: number

var y = "12";
console.log(typeof(y)); // Output: string
```

Understanding these concepts is fundamental for anyone diving into JavaScript and programming in general.

- variables rules
    - A variable name cannot start with a digit
        - ex 123
    - a variable cannot start with any symbol
        - ex @#
            - exception for $name and _name
    - variable names should be self-explanatory.
    - A variable name must start with a letter or an underscore character (_)
    - A variable name can only contain alpha-numeric characters and underscores (`a-z, A-Z`, `0-9`, and `_`)
        - ex- abc12 or ab12c.
    - Variable names are case-sensitive (age, Age and AGE are three different variables)

- variable **Declaration, Assigning, and Printing**.

```
var x; //declaration

x=10; //assigning

console.log(x); //printing to console/shell.

//I can also do

var x=10; //This is basically declaration and assigning togethe
```

Camel Case

Each word, except the first, starts with a capital letter:

myVariableName = "John"

Snake Case

Each word is separated by an underscore character:

my_variable_name = "John"

Mathematical Operators

common operators **Addition (+), Subtraction (-), Multiplication (*), and Division (/)**.

few examples.

```
var a=4;
var b=7;

console.log("The sum is ", a+b);
console.log("The Difference is ", a-b);
console.log("The Multiplication is ", a*b);
console.log("The Division is ", a/b);
```

Modulo Operator (%)

- Explain the **Remainder** by actually dividing something, take an example from the slide.

Exponentiation Operator (**)

- This basically calculates the **Power** of something.

```
var a=2;
var b=3;

console.log(a**b); // 8

var x=2;
var y=2;
var z=x*y;

console.log(z); // 4
```

▼ concatenation

- string + number= string
- number +number= number
- string + string = string.

String Concatenation (Joining)

```
var a="Hello";
var b="World";

var x=a+b;
console.log(x); // HelloWorld

console.log(a+" "+b) //Hello World
```

explain about the bracket. ((()+())

Boolean Datatype

- It can only contain **true** or **false.**

```
var x=true;

console.log(x); // true

console.log(typeof(x));  // boolean
```

Relational Operators or Comparison Operators

- There are 8 types of relational operators.

S.No	Operator	Description
1	>	Greater than
2	≥	Greater than or equal to
3	<	Less than
4	≤	Less than or equal to

- These take Two inputs and print the output as Boolean.

1. **> (Greater than)**
    - It will give me **true** if the **first value is strictly greater than the second one**.

- It will give you **false** when the **first value is less than or equal to the second value**.

```
// check if Goku height is greater than vegeta or not.

var goku_height= 6;
var vegeta_height= 5;
console.log(shiva_height>jai_height); // true ;
```

2. **≥ (Greater than equal to)**

- It will give you **true** if the first value is **greater than or equal to the second value**.
- It will give me **false** if the first value is **less than the second value**.

```
// case 1
6>=5 // true
4>=4 // true

//case 2
3>=10 // false

//Goku Example
var goku_marks=35;
var passing_marks=35;
var is_passed=goku_marks>=passing_marks;

console.log(is_passed); // true
```

- **inform them when to use > and when to use ≥.**

3. **< (Less than)**

- It will give you **true** if the first value is **strictly less than the second value.**
- It will give you **false** if the first value is **greater than or equal to the second value.**

```
console.log(5<5); // false
console.log(3<5); //true
console.log(15<8); // false
```

4. **≤ (Less than equal to)**

- It will give you **true** if the first value is **strictly less than or equal to the second value.**
- It will give you **false** if the first value is **greater than the second value.**

```
// case 1
3<=4 // true
5<=5 // true

//case 2
5<=1 // false
```

S.no	operator	Description
5	==	Double Equal to
6	!=	Not Equal to
7	===	Triple equal to
8	!==	Not double equal to

- **They can be applied to both numbers and strings.**
- **the output will be Boolean**

  == (Double Equal to)

  - It gives **true** if the **first value is equal to the second value.**
  - It gives a **false** if the **first value is not equal to the second value.**

```
console.log(2==2); // true
console.log("chunnu"=="chunnu"); // true
console.log(2==5); // false
console.log("Chunnu"=="chunnu"); //false
```

  != (Not Equal to)

  - Opposite of ==.
  - It gives **true** if both the values are **not equal**.
  - It gives **false** if both the values are **equal**.

```
console.log(2!=2); // false
console.log("chunnu"!="chunnu"); // false
console.log(2!=5); // true
console.log("Chunnu"!="Chunnu"); // true
```

  ===(Triple Equal to)

  - It gives **true** if the **first value is equal to the second value and also their datatypes are the same.**

- It gives a **false** if the **first value is not equal to the second value or their datatypes are different.**
- **If we check string “2” and number 2 with “==”, it will give true as the output, and to overcome this we use “===”.**
- **It also checks the data types of the operands.**

```
// if i use ==

console.log("2"==2); // true (not checking the datatype, only

// if i use ===

console.log("2"===2); // false (also checking the datatype al
```

!== (Not Double Equal to)

- Opposite of ===.
- It gives **true** if both the values or datatypes are **not equal**.
- It gives **false** if both the values and datatypes are **equal**.

```

// if i use !==

console.log("2"!==2); // true (also checking the datatype alo

consoele.log("3"!=="2") // true;

console.log("2"!=="2")// false
```