# Introduction

Ba Nguyễn

# What is JavaScript?

JavaScript là một “ngôn ngữ lập trình kịch bản” (scripting language), ban đầu, nó được tạo ra với mục đích duy nhất - cung cấp tính tương tác cho các trang web.

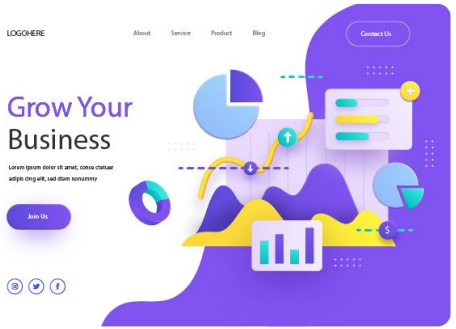
Ngày nay, JavaScript là ngôn ngữ phổ biến và có tốc độ phát triển nhanh nhất, nó được sử dụng trong nhiều mục đích và không còn bị giới hạn trong trình duyệt mà có thể sử dụng ở rất nhiều nền tảng / môi trường khác nhau, như máy chủ, điện thoại, ...

Trong môi trường trình duyệt, các đoạn mã JavaScript có thể được nhúng trong trang HTML và chạy tự động khi trang web được tải

# What can you do with JavaScript?

JavaScript ngày càng mạnh mẽ và có thể làm được rất nhiều thứ như:

- Web / Mobile Apps
- Real-time Networking Apps
- Command-line Tools
- Games
- ...

# Setting

Để bắt đầu **“chill”** cùng JavaScript, bạn cần có một trình duyệt và một Editor

- Cài đặt một trình duyệt nếu bạn chưa có
- Truy cập **code.visualstudio.com** tải và cài đặt trình soạn thảo VS Code

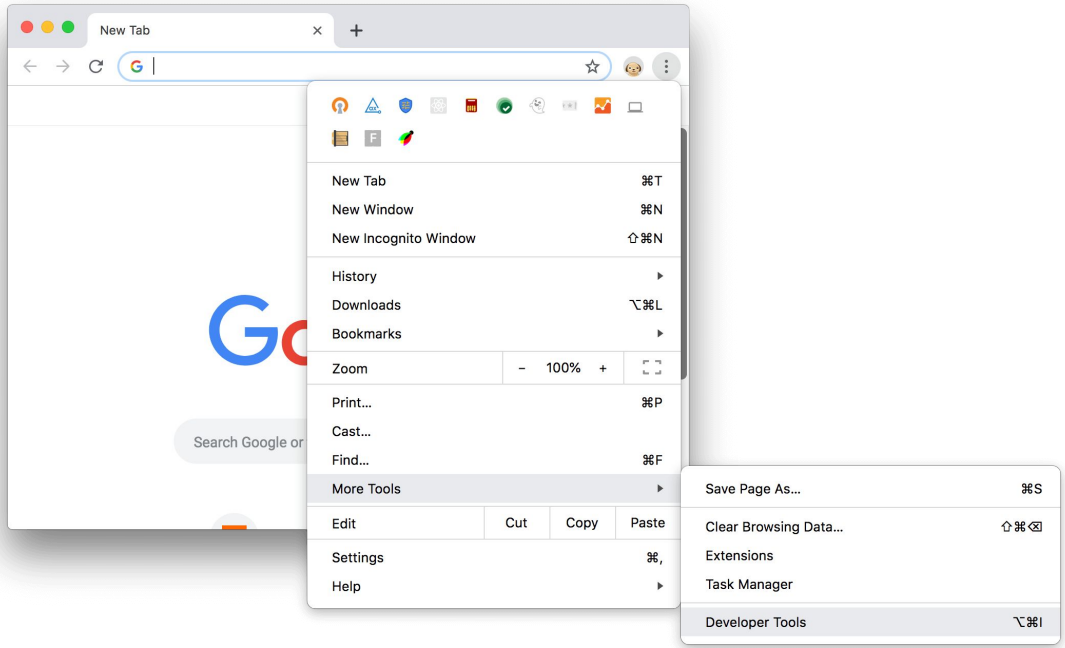
Cài đặt một số Extension hỗ trợ code **“fun”** hơn

- Live Server
- Prettier, ...

Thay đổi một số cài đặt cho VS Code

- Auto save
- Formatter, ...

# Dev Tools



# Basic

Ba Nguyễn

# Hello World

Có 3 cách để nhúng mã JavaScript vào trang web

### Internal Script

```
<html>
  <head></head>
  <body>

  <script>
   alert("Ba đẹp trai 😍")
  </script>
  </body>
</html>
```

### External Script

```
// script.js
alert("Ba đẹp trai 😍")

<!-- index.html -->
<script src="script.js">
</script>
```

### Inline Script

```
<h1
  onclick="alert('Ba đẹp
trai 😍')">
Hello World
</h1>
```

# Exercise

1. Tạo file **html**, nhúng mã JavaScript vào theo cả 3 cách, sử dụng phương thức `alert()` để hiển thị một thông báo khác nhau với mỗi cách

# Statements

### Câu lệnh

```
alert("Ba"); alert("Đẹp trai");
alert("😍😎🤣🥰");
```

💡 Các câu lệnh trong JavaScript nên kết thúc với dấu “;”

### Chú thích

```
// oneline comment
/* multi-line comment
à mây ding, gút chóp!!!
one more line*/
```

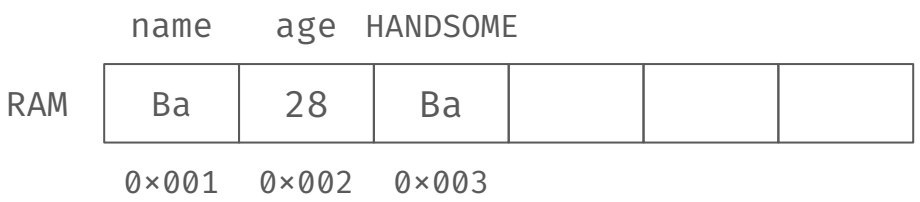
💡 Ghi chú giúp nhớ và hiểu cách mã thực thi

# Variables

Biến là yếu tố cơ bản của bất kỳ ngôn ngữ lập trình nào. Biến là "tên" một vùng nhớ lưu trữ dữ liệu trong bộ nhớ máy tính.

Cú pháp khai báo biến trong JavaScript:

```
let name = "Ba";
let age = 28;
const HANDSOME = true;
var myPower = 99.99;
let a = b = c = "😛";
var a = "😂", b = "😴";
let c;
alert(name);  // Ba
```



# Exercise

1. Khai báo một vài biến, sử dụng phương thức `alert()` hiển thị giá trị trên trình duyệt

# Variables

### let

Biến khai báo với **let** không thể khai báo lại, tuy nhiên, giá trị của biến có thể thay đổi

```
let name = "Ba";
let name = "Oops";  // ❌ Error
name = "Ba handsome 😍";  // 👌 Okey
name = 123;  // 👌 Okey
name = true;  // 👌 Okey
```

### const

Biến khai báo với **const** (hằng số) không thể khai báo lại, và cũng không thể thay đổi giá trị

```
const name = "Ba";
const name = "Oops";  // ❌ Error
// ❌ Error
name = "Ba not handsome 😥";
name = 123;  // ❌ Error
name = true;  // ❌ Error
```

# Variables

**var** là cách khai báo cũ, biến được khai báo với **var** được phép khai báo lại, thay đổi giá trị, và phạm vi **(scope)** của biến cũng khác biệt so với **let** và **const**.

```
var name = "Ba";
var name = "Ba đô la 🤩";  // 👌 Okey
```

** Scope là phạm vi tồn tại của một biến, sẽ được để cập sau

💡 Nên sử dụng **let** và **const** để khai báo biến

# Variables Naming Rules & Conventions

**Các quy tắc đặt tên biến**

- Tên biến **chỉ được chứa** *ký tự*, *số*, hoặc ký tự đặc biệt $ và _
- Tên biến **không được** bắt đầu bằng *một số*
- Tên biến **có phân biệt** *chữ hoa*, *chữ thường*
- Tên biến **không được** trùng với *từ khóa* của JavaScript

**Các quy ước đặt tên biến**

- JavaScript sử dụng phong cách **camelCase** cho tên biến, hoặc phương thức
- Với hằng số mà giá trị được xác định ngay từ đầu, sử dụng **UPPERCASE**

# Exercise

Tên biến nào không hợp lệ?

```
let 3a = 1;
let var = 10;
let my-name = "ahihi";
const lol = "Leaguage of Legends";
const PI = 3.14;
var __ = "$$";
var let = "__";
```

```
let x5 = 55;
let x_X = "OmG";
let $y__ = 1;
const ___ = "GOOD";
var PI = 3.14;
let FIRSTNAME = "Ba";
const birthday = "24.05.1992";
let LAstNaME = "Nguyễn";
```

# Data Types

Có 8 kiểu dữ liệu trong JavaScript

1. number
2. bigint
3. string
4. boolean
5. undefined
6. null
7. symbol
8. object

💡 Để kiểm tra kiểu dữ liệu của một biến, sử dụng `typeof(name)` hoặc `typeof name`

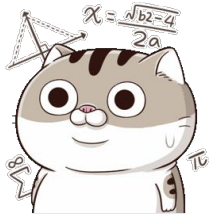
# Exercise

1. Sử dụng phương thức `alert()` in ra kiểu giá trị của các biến đã khai báo

# Numbers

Bao gồm cả số nguyên và số thực, giới hạn -2^53 + 1 đến 2^53 - 1 😛

```
let int = 1;
let float = 1.2345;
let complex = 1e10;
let binary = 0b01;
let hex = 0×ff;
```



Các giá trị đặc biệt trong kiểu **“number”**

```
Infinity;  // 1 / 0
-Infinity;  // -1 / 0
NaN;  // Not a Number
```

# Strings

Là một chuỗi ký tự được đặt trong cặp dấu ‘’, “” hoặc ``

```
let firstName = "Ba";
let lastName = "Nguyễn";
let message = 'I\'m very "handsome"';  // escape string
let greeting = `Hi, I'm ${firstName} ${lastName}`;  // template literal
```

💡 Template literal (` `) là cú pháp mới, cho phép “nhúng” giá trị của một biến, biểu thức, hoặc thậm chí một phương thức vào chuỗi sử dụng cú pháp ${ }, thay thế cho cách cổ điển:

```
let greeting = "Hi, I'm " + firstName + " " + lastName;
```

💡 Một số ký tự cần “escape” trong chuỗi như \' , \" , \\ , \n , ...

# Booleans, Null, Undefined

Kiểu **“boolean”** hay logic chỉ bao gồm hai giá trị **true** hoặc **false**

```
let isPretty = true;
let isUgly = false;
```

**null** và **undefined** là 2 giá trị đặc biệt, chúng thuộc về hai kiểu riêng **“null”** và **“undefined”**, **null** có nghĩa là một cái gì đó **không tồn tại**, còn **undefined** là cái gì đó **có tồn tại**, nhưng **giá trị không được xác định**.

💡 Khi khai báo một biến mà không gán giá trị, biến sẽ có giá trị là **undefined**

# Symbol, Object

**“object”** là kiểu dữ liệu đặc biệt, các biến với kiểu dữ liệu khác chỉ lưu trữ được một giá trị tại một thời điểm, còn **object** cho phép lưu cùng lúc nhiều giá trị trong một biến duy nhất, các dữ liệu được lưu trong **object** có thể thuộc bất kỳ kiểu nào

```
let ba = {
    name: "Ba",
    age: 28,
    greeting() { alert(this.name + "đẹp trai quá 😍") }
};
```

**“symbol”** được dùng để tạo ra các ID duy nhất, thường dùng với **object**

```
let ba = Symbol("ba");
```

# Interactions

JavaScript cung cấp sẵn một số phương thức để tương tác (nhập, hiển thị dữ liệu) trên trình duyệt:

```
alert("Hello");
let name = prompt("Enter your name"); // string / null
let love = confirm(`Do you love me, ${ name }? 😉` ); //  true / false
console.log("I appreared in console");
console.log(name);
console.log(answer);
```

# Exercise

1. Viết chương trình cho phép nhập vào một vài thông tin sử dụng các phương thức `confirm()`, `prompt()` VD: **name**, **age**, ... Sử dụng `alert()` để in ra các thông tin vừa nhập. Nối chuỗi sử dụng template literal
2. Thay đổi chương trình, sử dụng `console.log()` để hiển thị ra **console**

# Type Conversions

JavaScript hỗ trợ **chuyển đổi kiểu dữ liệu tự động** khi thực hiện tính toán các giá trị, và cũng cung cấp thêm một số phương thức để chuyển đổi từ kiểu dữ liệu này sang kiểu dữ liệu khác

### String

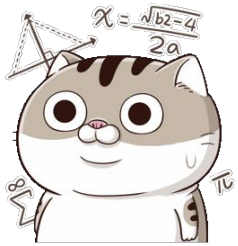
```
String(123); // "123"
String(true); // "true"
String(null); // "null"
String(undefined); // "undefined"
String(); // ""
String("Ami bụng bự"); // "Ami bụng bự"  <----------------
```

# Type Conversions

### Number

```
Number("123.456"); // 123.456
+"123.456"; // 123.456
+"-123.456"; // -123.456
```

Một số giá trị đặc biệt khi chuyển về kiểu **number**



```
+undefined; // NaN
+null; // 0
+true; // 1
+false; // 0
+"  123   "; // Khoảng trắng ở đầu và cuối chuỗi được loại bỏ → 123
+"123abc"; // Chuỗi chứa ký tự → NaN
+""; // 0
```

# Type Conversions

### Boolean

```
Boolean(123); // true
Boolean("Ba đẹp trai 😎"); // absolutely
!"Ba xấu trai?"; // false --- phủ định
!!"Ba đẹp trai"; // absolutely --- phủ định của phủ định 😄
```

Một số giá trị đặc biệt khi chuyển về kiểu **boolean**

```
"", 0, null, undefined, NaN; // → false
// All other values → true
```

# Operators

**Operand - toán hạng**

Là các giá trị tham gia vào phép tính, VD 1 + 2 → 1 và 2 là toán hạng

**Urany - toán tử đơn**

Là các toán tử đi kèm với 1 toán hạng, VD `+"abc"`, `typeof "abc"`

**Binary - toán tử đôi**

Là các toán tử đi kèm với 2 toán hạng, VD: 1 + 2, 2 - 1, ...

💡 JavaScript hỗ trợ **chuyển đổi kiểu dữ liệu tự động** khi thực hiện phép tính

# Operators

**Basic operator** `+ - * / % **`

Chia lấy phần dư `%`

```
5 % 3; // 2              "6" % 2; // 0            1 % true; // 0
```

Lũy thừa `**`

```
2 ** 2; // 4             2 ** "2"; // 4           2 ** false; // 1
```

💡 Mọi phép tính (trừ phép `+` chuỗi) với `NaN` đều cho kết quả là `NaN`

```
5 - NaN / 2; // NaN      1 * 2 * undefined; // NaN
```

# Exercise

1. Viết chương trình cho phép nhập 2 số a, b, in ra kết quả tất cả phép tính cơ bản với 2 số. Lưu ý chuyển đổi kiểu dữ liệu khi nhập vào

# Homework

1. Viết chương trình cho phép nhập vào chiều dài, chiều rộng của hình chữ nhật, tính và in ra chu vi, diện tích của hình chữ nhật
2. Viết chương trình cho phép nhập vào bán kính hình tròn, tính và in ra chu vi, diện tích của hình tròn
3. Viết chương trình cho phép nhập vào hệ số a, b của phương trình bậc nhất ax + b = 0, tính và in ra nghiệm của phương trình
4. Viết chương trình cho phép nhập vào một số là đơn vị cm, tính và in ra giá trị tương ứng ở các đơn vị mm, m, km
5. Viết chương trình cho phép nhập vào một số là nhiệt độ có đơn vị Celsius, in ra nhiệt độ ở đơn vị Fahrenheit và Kevin tương ứng
6. Viết chương trình cho phép nhập một số phút tính từ 0h, tính và in ra giờ/phút tương ứng

# Operators

**Concat String +**

Đối với kiểu dữ liệu **String** phép + chuyển đổi kiểu dữ liệu của toán hạng về kiểu **string** và thực hiện nối chuỗi

```
1 + "2"; // "12"          1 + 2 + "3"; // "33"     "1" + 2 + true; // "12true"
```

Phép nối chuỗi chỉ hoạt động **duy nhất với toán tử +** , với những toán tử khác, mọi kiểu dữ liệu được chuyển về kiểu **number**

```
6 - "2" + null - true; // 3          "5" / 3 + 2 / "1"; // 4
```

💡 Quy tắc thực hiện biểu thức theo thứ tự từ trái qua phải, dựa theo độ ưu tiên toán tử. Các toán tử có độ ưu tiên khác nhau, quyết định phép tính nào sẽ được thực hiện trước

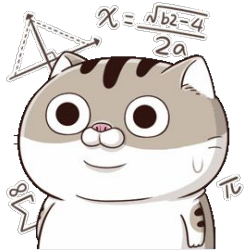
# Operators

### Gán =

```
let a = 3; // a = 3
let b = 1 + 2 + 3; // b = 6
let c = (a = b + 5); // b = 6, a = 11, c = 11
```

### Gán kết hợp

```
a += b; // a = a + b
a /= b; // a = a / b
a %= b + 5; // a = a % (b + 5)
a *= a + 2; // a = a * (a + 2);
```



# Operators

**Increment, Decrement (tự tăng/giảm)**

`++` và `--` là hai toán tử đặc biệt, nó thực hiện phép tính **tăng/giảm** giá trị của biến đi 1, hai toán tử này có thể đặt ở trước (prefix) hoặc sau (postfix), khi nằm trên một câu lệnh riêng biệt thì không có sự khác nhau. VD: `let a = 1;`

```
a++; // a = 2        a--; // a = 1        ++a; // a = 2        --a; // a = 1
```

Tuy nhiên, khi đặt trong một biểu thức, `a++` tăng `a` lên 1, nhưng trả về **giá trị trước khi tăng**, `++a` cũng tăng `a` lên 1, nhưng trả về **giá trị sau khi tăng**

```
let a = 1;
let b = a++ + 2; // a = 2, b = 3 (vì a++ trả về 1)
let c = ++a + 2; // a = 3, c = 5 (vì ++a trả về 3)
let d = a++ + ++a - a-- - --a; // d = 3 + 5 - 5 - 3 = 0, a = 3
```

# Exercise

Tính giá trị các biểu thức

```
let a = 1, b = (a % 2) * 2,
c = (a++ - b--), d = "0";
```

1. `a + b + c + d;`
2. `a - b + c - d;`
3. `a-- + b-- * c / d;`
4. `++a - +b * c + d;`
5. `d + ++a + --b % c;`
6. `a-- - +d++ - ++c + b--;`
7. `a++ - b-- + ++c + --d;`

```
let a = 1, b = (a * 2) / 2,
c = (a-- + b++), d = "-0";
```

1. `a - b - c - d;`
2. `a + b - c + d;`
3. `a++ - b++ / c * d;`
4. `--a + -b / c - d;`
5. `d - --a - ++b * c;`
6. `a++ + -d-- + --c - b++;`
7. `a-- + b++ - --c - ++d;`

# Operators

**Toán tử so sánh** `==` `!=` `>` `>=` `<` `<=` `===` `!==`

Kết quả của các phép so sánh là một giá trị Boolean

`==` và `!=` (và cả `>` `>=` `<` `<=`) tự động chuyển đổi kiểu dữ liệu của 2 toán hạng về cùng một kiểu và thực hiện so sánh

```
2 == "2"; // true
2 != "2"; // false
1 > null; // true
```

`===` và `!==` (strict comparison) so sánh cả kiểu giá trị của dữ liệu

```
2 === 2; // "number" == "number" và 2 == 2 → true
2 === "2"; // “number” == “string” → false
2 !== "2"; // “number” != “string” → true
2 !== 2; // "number" != "number" → false
```

TechMaster
Học là có việc

# Operators

**So sánh chuỗi**

JavaScript sử dụng bảng mã Unicode, khi so sánh 2 chuỗi, nó thực hiện so sánh từng ký tự dựa theo thứ tự trong bảng mã (Unicode table)

```
"a" > "A"; // true          "A" > "Z"; // false          "Ba" == "Ba"; // true
```

**null, undefined, NaN**

💡 Mọi biểu thức so sánh với NaN đều cho kết quả là False

**null** và **undefined** là 2 trường hợp đặc biệt

```
null == 0; // false          null <= 0; // true          null >= 0; // true
null == undefined; // true          null === undefined; // false
```

# Operators

**Toán tử logic** `||` **- or**, `&&` **- and**, `!` **- not**

`||` - **or** tìm giá trị `true` đầu tiên trong biểu thức (khi chuyển về kiểu **boolean**) và trả về giá trị đó, nếu không có thì trả về giá trị cuối cùng trong biểu thức

```
true || false; // true
0 || 1; // 1
0 || false || ""; // ""
"abc" || true; // "abc"
let age = 16;
age > 18 || alert("Movies are only for 18+"); // alert
```

# Operators

**Toán tử logic** `||` - **or**, `&&` - **and**, `!` - **not**

`&&` - **and** tìm giá trị `false` đầu tiên trong biểu thức (khi chuyển về kiểu **boolean**) và trả về giá trị đó, nếu không có thì trả về giá trị cuối cùng trong biểu thức

```
true && false; // false        0 && 1; // 0
0 && false && ""; // 0         "abc" && true; // true
let age = 16;
age > 18 && alert("Movies are only for 18+"); // not alert
```

`!` - **not** chuyển giá trị về kiểu **boolean** và phủ định nó

```
!""; // true                    !"123"; // false
!!false; // false               !!"xxx"; // true
```

# Exercise

Tính giá trị các biểu thức

```
let a = true, b = !a;
let c = !a && !!b || 0;
```

1. `a && b && c;`
2. `a || b || c;`
3. `a && !b || !!c;`
4. `!(a || !b) && c;`
5. `!!(a && !!b) || !c;`


```
let a = false, b = !!a;
let c = a || !b && 0;
```

1. `a && b && c;`
2. `a || b || c;`
3. `!a || b && !c;`
4. `!!(!a && b) || c;`
5. `!(!a || !b) || c;`

# Control Statements

Ba Nguyễn

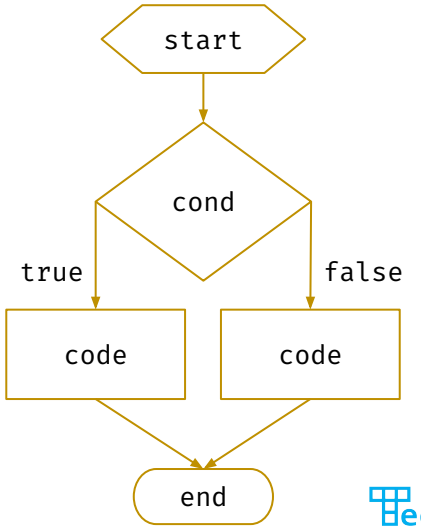
# If else

Thực thi các câu lệnh theo điều kiện chỉ định, điều kiện có thể là một giá trị, biểu thức, hàm, ... và được tự động chuyển đổi về kiểu **boolean**. Mệnh đề **if** chỉ định điều kiện và code bên trong chỉ được thực thi nếu điều kiện đúng, ngược lại, mệnh đề **else** sẽ được thực thi

Cú pháp:

```
if (condition) {
    // run if
condition is true
} else {
    // run if
condition is false
}
```

```
if (condition) {
    // condition
} else if (other condition) {
    // other condition
} else if (another condition) {
    // another condition
} else {
    // if all condition false
}
```



# Exercise

1. Viết chương trình cho phép nhập vào một số, kiểm tra số đó là chẵn hay lẻ và in ra màn hình
2. Viết chương trình cho phép nhập vào 2 số, kiểm tra và in ra số lớn hơn
3. Viết chương trình cho phép nhập vào một tháng, in ra mùa tương ứng trong năm

# Homework

1. Viết chương trình cho phép nhập ba số, kiểm tra và in ra số lớn nhất
2. VCT cho phép nhập một số, kiểm tra và in ra số đó có chia hết cho 5 và 11 hay không
3. VCT cho phép nhập một năm, kiểm tra và in ra năm đó có phải năm nhuận hay không
4. VCT cho phép nhập một ký tự, kiểm tra và in ra ký tự đó có thuộc bảng ký tự alphabe (a-zA-Z) hay không
5. VCT cho phép nhập một ký tự, kiểm tra và in ra ký tự đó là nguyên hay phụ âm (tiếng Anh)
6. VCT cho phép nhập một ký tự, kiểm tra và in ra ký tự đó là chữ thường hay chữ in hoa
7. VCT cho phép nhập ba hệ số a, b, c, của phương trình bậc 2 $ax^2 + bx + c = 0$, tính và in ra nghiệm phương trình
8. VCT quy đổi điểm hệ số 10, sang điểm hệ chữ cho sinh viên, với điểm =< 10 là A, < 8.5 là B, < 7.0 là C, < 5.5 là D, < 4.0 là F

# ?

Toán tử ? (toán tử hỏi, toán tử ba ngôi) là cú pháp rút gọn của **if else**, thường dùng khi muốn gán một giá trị theo điều kiện

Cú pháp:

```
condition ? value1 :
value2;

condition
? value1
: otherCondition
? value2
: anotherCondition
? value3 : value4;
```

```
let yearOld = +prompt("How old are u?");
let age =
    yearOld < 15
        ? "Kid"
        : yearOld < 18
        ? "Teen"
        : yearOld < 50
        ? "Adult"
        : "Old";
```

# Exercise

1. Viết chương trình cho phép nhập vào một số, kiểm tra số đó là chẵn hay lẻ và in ra màn hình, sử dụng toán tử ?
2. Viết chương trình cho phép nhập vào 2 số, kiểm tra và in ra số lớn hơn, sử dụng toán tử ?
3. Viết chương trình cho phép nhập vào một tháng, in ra mùa tương ứng trong năm, sử dụng toán tử ?

# Switch case

Là cấu trúc rẽ nhánh khác, tuy nhiên, mỗi **case** là một giá trị duy nhất (với **if else** có thể là một khoảng giá trị), và nó so sánh giá trị với kiểu nghiêm ngặt ===, với **switch case**, *giá trị cần so sánh* và *giá trị mỗi case* phải có *cùng kiểu dữ liệu*

Có thể gộp nhiều case, bất kỳ một case nào là **true** thì code sẽ được thực thi. Khi tất cả các case đều **false**, code trong **default** sẽ được thực thi.

Mỗi case nên kết thúc bởi câu lệnh **break;** để thoát ra khỏi switch case, nếu không, khi một case được thực thi mà không có **break;** *tất cả các case phía dưới* sẽ được thực thi mà không cần kiểm tra điều kiện

Cú pháp:

```
let yearOld = +prompt();

switch (yearOld) {
    case 15:
        // code
        break;
    // ... many case
    default:
        // code
}
```