



Introduction

Ba Nguyễn

What is JavaScript?

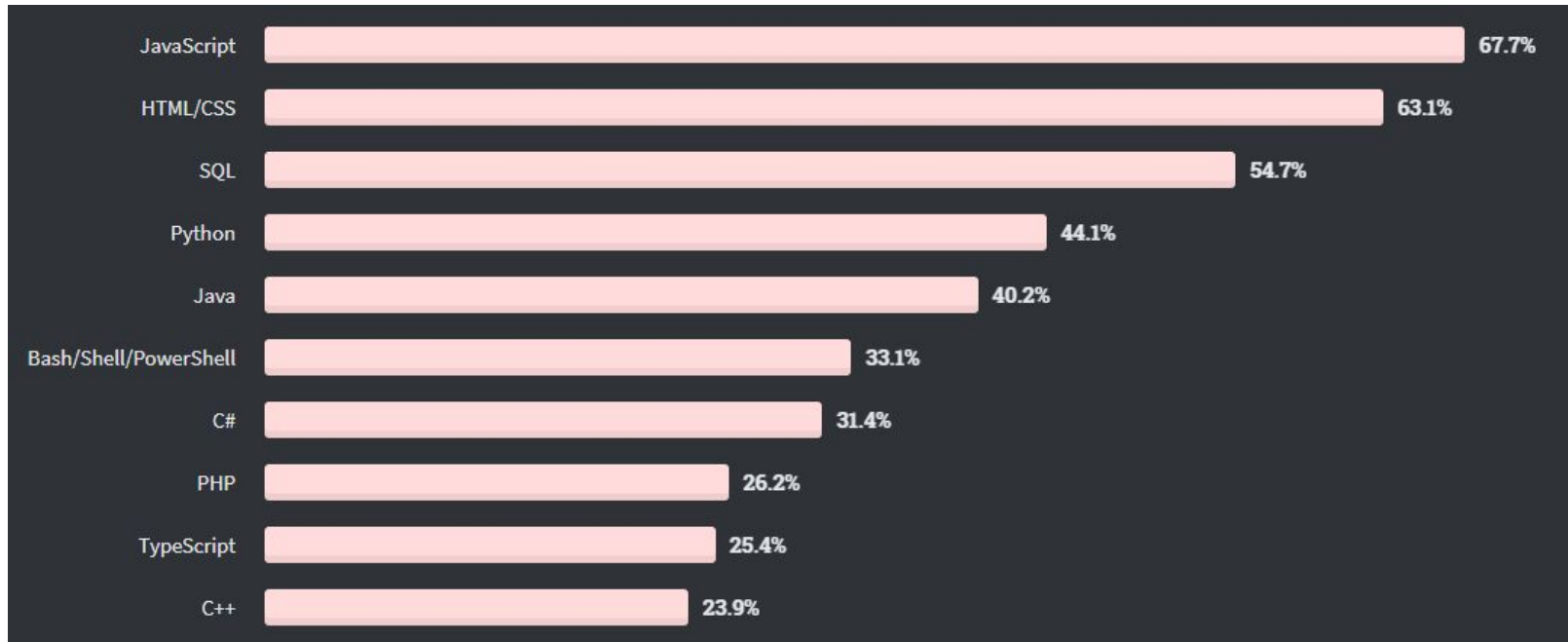
JavaScript là một “ngôn ngữ lập trình kịch bản” (scripting language), ban đầu, nó được tạo ra với mục đích duy nhất - cung cấp tính tương tác cho các trang web.

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 is JavaScript?

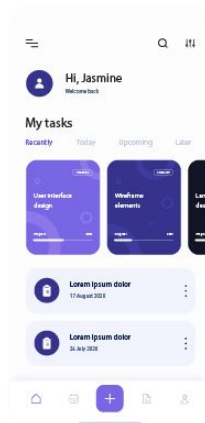


Top 10 ngôn ngữ phổ biến nhất 2020 (stackoverflow)

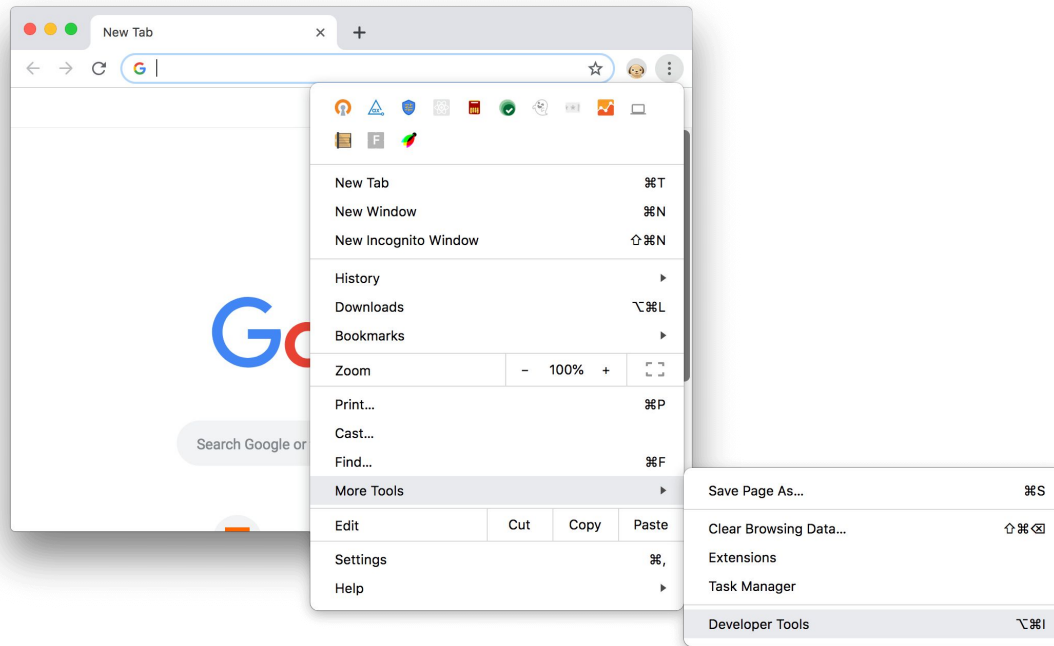
What can you do with JavaScript?

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
- ...



Dev Tools



Interactions



Một số hàm cơ bản để tương tác (hiển thị dữ liệu) trên trình duyệt/console

```
//Hiển thị bảng thông báo  
alert("Nội dung thông báo");
```

```
// Hiển thị ra console  
console.log("Giá trị");
```



Basic

Ba Nguyễn

Hello World



Nhúng mã JavaScript vào trang web

```
<body>
  <h1 onclick="alert('Hello JS')">...</h1>

  <script>
    alert("Hello JS");
  </script>

  <script src="script.js"></script>
</body>
```


Hello World

Statements

```
alert("Hello JavaScript!");  
alert("Mỗi câu lệnh nên kết thúc với ;")
```

Comments

```
// One-line comment  
/* Multi-line comment */
```

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 myName = "Ba Nguyễn";  
let myAge = 29;  
const MY_LOVE = "Bon Bon";
```

```
let a = b = 1;  
let x = 1, y = 2;
```

Variables

let

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

```
let myName = "Ba Nguyễn";  
// Error  
let myName = "Béo Ú";  
// Okey  
myName = "Béo Ú";
```

const

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

```
const MY_LOVE = "Bon Bon";  
// Error  
const MY_LOVE = "Béo Ú";  
// Error  
MY_LOVE = "Béo Ú";
```

Variables Naming Rules & Conventions



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

Quy ước đặt tên biến

- JavaScript sử dụng phong cách **camelCase** cho tên biến, hoặc hàm
- Với hằng số - **const** - sử dụng **UPPERCASE**

Exercise

Tên biến nào không hợp lệ?

```
let 3a = 1;
```

```
let var = 10;
```

```
let my-name = "Ba Nguyễn";
```

```
const job = "Developer";
```

```
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";
```



Data Types

Ba Nguyễn

Data Types

Có 9 kiểu dữ liệu trong JavaScript

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

Primitives

number
bigint
string
boolean
undefined
null
symbol

Reference

object
array
function

Kiểm tra kiểu dữ liệu của một biến/giá trị, sử dụng **typeof**

```
typeof(1); // number
```

Numbers

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

```
let integer = 1;
```

```
let float = 1.5;
```

```
// Một số giá trị số đặc biệt
```

```
Infinity; // 1 / 0
```

```
-Infinity; // -1 / 0
```

```
NaN; // Not a Number
```


Strings

Là một chuỗi ký tự được đặt trong cặp dấu ' ' hoặc " "

```
let firstName = "Nguyễn";
```

```
let lastName = "Ba";
```

```
// Nối chuỗi
```

```
let message = "Hello, " + firstName + " " + lastName;
```

```
console.log(message); // Hello, Nguyễn Ba
```

Booleans, Null, Undefined

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

```
let isOn = true;  
let isOff = false;
```

null và **undefined** là 2 giá trị đặc biệt, **null** đại diện cho một đối tượng *không tồn tại*, còn **undefined** đại diện cho một đối tượng *chưa được gán giá trị*.

```
let favorite = null;  
let number = undefined;
```

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

Object

object là kiểu dữ liệu đặc biệt, cho phép lưu trữ cùng lúc nhiều giá trị trong một biến duy nhất, các dữ liệu được lưu trong **object** có thể thuộc *bất kỳ kiểu nào*

// Khai báo

```
let objName = {  
    propName: value,  
    // ...  
};
```

// Truy cập thuộc tính

objName.propName; // Dot Notation

objName["propName"]; // Bracket Notation

Object

```
let user = {  
  name: "Ba",  
  isHandsome: true,  
};
```

```
console.log(user.name); // Ba  
console.log(user["isHandsome"]); // true
```

Array

array là cấu trúc dữ liệu cơ bản trong JavaScript, **array** cho phép lưu trữ một danh sách các giá trị/đối tượng có *thứ tự* và cung cấp các phương thức đặc biệt để xử lý các giá trị trong nó

// Khai báo

```
let arrName = [value, value, /*...*/];
```

// Truy cập giá trị trong arr

// Thứ tự các mục bắt đầu bằng 0

```
arrName[0]; // Bracket Notation
```

```
arrName[1];
```

Array

```
let sizes = ["xs", "s", "m", "l"];
```

```
console.log(sizes[0]); // xs
```

```
console.log(sizes[1]); // s
```

```
console.log(sizes[2]); // m
```

```
console.log(sizes[3]); // l
```

Function

function là cách thức tổ chức mã cơ bản trong JavaScript, **function** được sử dụng để đóng gói một đoạn mã để xử lý một công việc/tính toán giá trị nào đó, cho phép tái sử dụng đoạn mã ở nhiều nơi trong chương trình

```
// Khai báo
function funcName() {
    // code
} // không cần ;
```

```
// Gọi hàm
funcName();
```

Function

// Khai báo

```
function hello() {  
    alert("Hello!!!");  
    alert("Welcome to JavaScript!!!");  
}
```

// Gọi hàm

```
hello();
```


Local vs Global Variables



Một biến được khai báo bên trong hàm được gọi là **biến local** - chỉ **tồn tại** bên trong hàm đó

Ngược lại các biến khai báo bên ngoài tất cả các **function** (hoặc khối code) được gọi là biến **global** - có thể truy cập ở mọi nơi

Thông thường, nên tránh truy cập trực tiếp tới giá trị của một biến bên ngoài hàm, thay vào đó nên sử dụng **tham số - parameters** và **đối số - arguments**

Local vs Global Variables

```
let x = 1; // Biến global
```

```
function func() {  
    let y = 10; // Biến local  
  
    console.log(x); // 1  
    console.log(y); // 10  
}
```

```
console.log(x); // 1  
console.log(y); // ❌ Lỗi 'y is not defined'
```

Function

function có thể nhận các giá trị đầu vào - **input** - và thay đổi cách nó hoạt động dựa trên giá trị đó, mỗi **function** có thể nhận số lượng giá trị đầu vào bất kỳ.

```
// Khai báo
```

```
function funcName(parameters) {
```

```
    // code
```

```
}
```

```
// Gọi hàm
```

```
funcName(arguments);
```

Function

// Khai báo

```
function hello(name) {  
    alert("Hello " + name + "!!!");  
    alert("Welcome to JavaScript!!!");  
}
```

// Gọi hàm

```
hello("Ba"); // Hello Ba!!!  
hello("Bon"); // Hello Bon!!!
```

Two Types of Function

Các hàm được chia thành 2 loại (kiểu):

1. Thực thi một công việc (tác vụ) nào đó, không quan tâm kết quả
2. Tính toán một giá trị và trả về kết quả

Mặc định, mọi hàm luôn trả về một giá trị sau mỗi câu lệnh gọi hàm là **undefined**. Để chỉ định một giá trị cụ thể là kết quả trả về của hàm, sử dụng câu lệnh **return**

```
function funcName() {  
    return value;  
}
```

```
funcName(); // value
```

Two Types of Function

```
function sum(a, b) {  
    return a + b;  
}
```

```
console.log(sum(1, 2));  
// 1. sum(1, 2) trả về 1 + 2 = 3  
// 2. console.log(3);
```

Hoisting

Trong JavaScript, một hàm có thể được gọi trước khi nó được khai báo trong mã. Bởi khi chạy, các khai báo hàm (và biến với **var**) được đưa lên đầu khối code hoặc tệp lệnh

```
func(); // 👍
```

```
// Khai báo hàm
```

```
function func() {  
    console.log("Hoisting");  
    console.log("Khai báo hàm sẽ được 'câu' lên đầu");  
}
```

```
func(); // 👍
```

Exercises



1. Viết hàm **hello(user)** in ra các thông tin **name**, **birthday**, **email**, **phone** trong object **user** truyền vào theo mẫu sau

User name: ...

Birthday: ...

Email: ...

Phone: ...

2. Viết hàm **multiple(a, b)** tính giá trị của biểu thức **a * b**, trả về kết quả của biểu thức đó
3. Viết hàm **sum(arr, i, j)** tính tổng 2 giá trị tương ứng thứ **i** và **j** trong mảng **arr**, trả về kết quả của biểu thức đó