Introduction to programming

Lecture 1

Aim of the course

To make you feel that coding is a simplest and beautiful thing that could be done in artistic way.

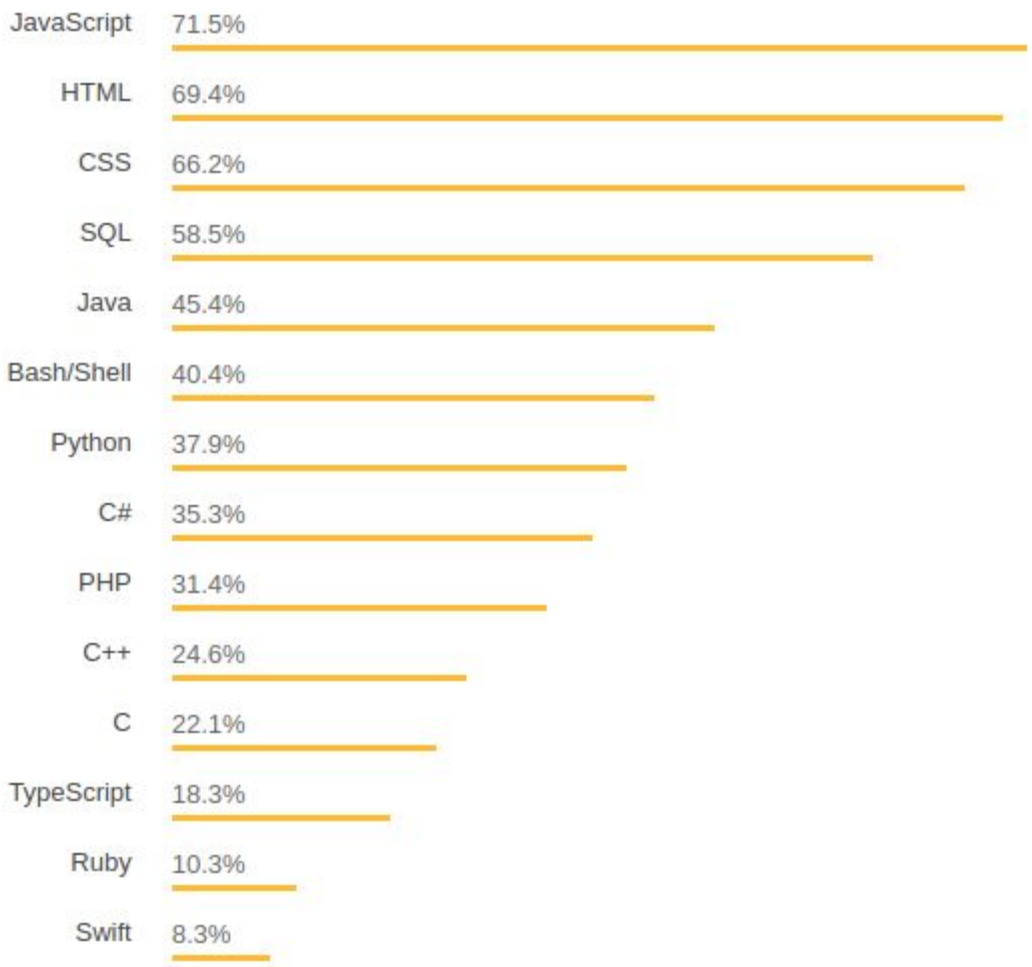
We look majorly at JavaScript, than C/JAVA.

To write a computer program for a problem we have to guess first.

Why JavaScript

Most popular languages, from past 3 years.

Every Browser supports it, From past few year it is also used as a backend language.



Annual Salary

frontend developer | montreal, qc, canada | 1 years experience | bachelors degree (eg ba, bs) | javascript | php | wordpress

Salary is a pre-tax value based on the 2018 Developer Survey results. Learn more about our methodology.

Intuition

It makes you comfortable to code in any language.

Most of the successful software developers don't learn an entire programming language before they start their career.

WHAT DOES A COMPUTER DO

Fundamentally:

1. performs calculations a billion calculations per second!
2. remembers results 100s of gigabytes of storage!

What kinds of calculations?

1. ones that you define as the programmer

Example:

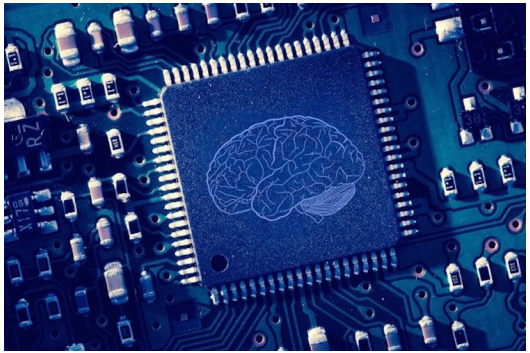
```
a = 2;
b = 3;
c = a + b;
d = c + 1;
```

a,b,c,d are stored on RAM, and the intermediate results are computed and stored on RAM.

computers only know or do what you tell them

Note: Don’t blame them unnecessarily

A Simple Computational Thinking



Treat the computer as your friend brain -who knows only how to do calculations.

#1:
So, you say to your friend that a = 2 and b = 3, and then ask him what is a + b.

#2:
Say all your classmates names and their respective mobile numbers to your friend, and ask him to provide the mobile number of your friend X.

In previous slide, we have seen an simple example on how computation works.

#3:
A calculator on your phone works like that when you add two numbers, We can also consider very complex application like Google Maps which does a lot of complex calculations in order to show you a best route between two places.

High level and low level languages

High level and low level languages - As we said previously computer only know how to do calculations, computer in the sense of CPU, and these CPU manufacturers like intel design them in such a way that these CPU’s understand only the Assembly(low-level language).

Loosely speaking, computers can only execute programs written in low-level languages. Thus, programs written in a high-level language have to be processed before they can run. This extra processing takes some time, which is a small disadvantage of high-level languages.

LDA 2050	A ← 2050
MOV B, A	B ← A
LDA 2052	A ← 2052
ADD B	A ← A+B
STA 3050	A → 3050
LDA 2051	A ← 2051
MOV B, A	B ← A
LDA 2053	A ← 2053
ADC B	A ← A+B+CY
STA 3051	A → 3051
HLT	Stops execution

This is what CPU understands

```
a = 2
b = 3
c = a + b
```

But we write one of these

```
int a = 2;
int b = 3;
int c = a + b;
```

```
var a = 2
var b = 3
var c = a + b
```

Who Translates this?

To print 9 stars in Assembly versus Higher-Level Languages

```
_start:          ;tell linker entry point
   mov  edx,len  ;message length
   mov  ecx,msg  ;message to write
   mov  ebx,1    ;file descriptor (stdout)
   mov  eax,4    ;system call number (sys_write)
   int  0x80     ;call kernel

   mov  edx,9    ;message length
   mov  ecx,s2   ;message to write
   mov  ebx,1    ;file descriptor (stdout)
   mov  eax,4    ;system call number (sys_write)
   int  0x80     ;call kernel

   mov  eax,1    ;system call number (sys_exit)
   int  0x80     ;call kernel

section .data
msg db 'Displaying 9 stars',0xa ;a message
len equ $ - msg  ;length of message
s2 times 9 db '*'
```

JavaScript

```
console.log("*********");
```

JAVA

```
public class MyClass {
    public static void main(String args[]) {
        System.out.println("*********");
    }
}
```

Suppose you have an old computer which you bought in 2008 and still working, and recently you bought a Microsoft surface .

It is clear in above scenario that your new Microsoft Surface has advanced hardware architecture than your old computer.

For suppose Microsoft Surface pro uses Intel i5 processor, where your old computer could uses 8086 processor.

If you install C/C++ compiler on these both systems, the compiler generates different Assembly code based on the CPU architecture.

Advantages with a high-level language are enormous:

1. First, it is much easier to program in a high-level language. Programs written in a high-level language take less time to write, they are shorter and easier to read, and they are more likely to be correct.

2. Second, high-level languages are **portable**, meaning that they can run on different kinds of computers with few or no modifications.

Low-level programs can run on only one kind of computer and have to be rewritten to run on another.

HOW HIGH-LEVEL IS CONVERTED INTO LOW-LEVEL LANGUAGE?

Source Code

First it is just a text file

You write some code in a file, saying hey send a message to another computer or add two number.

Compile the file

It converts the english words you wrote into machine understandable code.

RUN it

Then your machine does the tasks that you written on the file.

Write your code saying, I want to find the highest mark a class got

Not Just error checking, I also generate machine code.

Then you give it to your compiler which tells mistakes in your code.

Then, you have to Run the machine code that compiler has generated

This is standard procedure with C/C++ compilers. Java also does same but little bit different. - Let’s discuss

Different programming languages - Different texts

These are just different ways to speak to a computer(to give work to it)

Java

```
Class Sample{

public static void main(String args[]){
  // beginning or entry point
  int a = 10; int b = 20;
  int c = a + b;
}
}
```

JavaScript

```
var a = 10;
var b = 10;
var c = a + b;
```

PHP

```
$a = 10;
$b = 20;
$c = $a + $b;
```

Treat it as a empty brain. And you just have to give commands to use its brain(cpu).

JavaScript Code

```
var a = 2;
var b = 45;
var c = 23 + 33;
console.log(c);
console.log(d);
```

Java Code

```
Class Sample{

public static void main(String args[]){
  int a = 12;
  System.out.println(a);
  System.out.println(b);

 }
}
```

Do you find any error in above source code?

What difference you found after running these two programs?

Even though there is an error in the code, JavaScript given the output until it encountered the first error, because it does interpretation.

```
56

"error"

"ReferenceError: d is not defined
```

But Java doesn't even generate the intermediate file(.class file) as there is a error, hence you cannot run the code at all.

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        b cannot be resolved to a variable

        at com.fa.Sample.main(Sample.java:9)
```

Printing in JavaScript

Code:

```
var x = 12;
var y = 13;
console.log(x);
console.log(“x”);
```

Output:

```
12
x
```

Code:

```
var x = 12;
var y = 13;
console.log(x+y);
console.log(x>y);
console.log(x<y);
```

Output:

```
15
false
true
```

```
> var %toronto = 2
Uncaught SyntaxError: Unexpected token %
> var $toronto = 2
<· undefined
> var +toronto = 2
Uncaught SyntaxError: Unexpected token +
> var /toronto = 2
Uncaught SyntaxError: Unexpected token /
> var _toronto = 2
<· undefined
> var 23toronto = 2
Uncaught SyntaxError: Invalid or unexpected token
>
```

```
> var a = 2
  var b = 3
  a = 4
<· 4
```

```
> var z
  z = 2
<· 2
```

```
> var t = 2
<· undefined
>
```

```
var r
var f = 0
r = 2
var x = (r == f)
```

```
2
```

```
x
```

```
false
```

```
var z = ( 2 + 3) * 10;
undefined
```

```
z
50
```

```
var z = 2 + 3 * 10;
undefined
```

```
z
32
```

```
3 > 4
```

```
false
```

```
3 > 3
```

```
false
```

```
3 >= 3
```

```
true
```

```
var x = 4
var t = 3.002
var d = 'hello how are u'
var h = [  5,2,3,1,23,4,4 ]
```

```
undefined
```

```
console.log(x)
```

```
4
```

```
undefined
```

```
console.log(h)
```

```
▶ (7) [5, 2, 3, 1, 23, 4, 4]
```

```
undefined
```

```
console.log(h[0])
```

```
5
```

```
undefined
```

```
console.log(h[1])
```

```
2
```

```
var s = [ 4.01, 24.23, 3.21, 5.6, 8 ]
undefined
```

```
s[0]
4.01
```

```
s.length
5
```

```
s[s.length - 1]
8
```

```
s[2 - 1]
24.23
```

```
var temp = 9
undefined
```

```
s[temp - 6]
5.6
```

```
false && false
false
```

```
false && true
false
```

```
true && true
true
```

```
true || true
true
```

```
true || false
true
```

```
false || false
false
```

```
var a1 = 3
var a2 = 4
var a3 = 1
var t = a1 > a2 && a1 > a3
```

```
undefined
```

```
t
```

```
false
```

```
a2 > a1 && a2 > a3
```

```
true
```

a/b

divisor

a%b

remainder.

```
var ft = (10%2 == 0)
undefined
ft
true
```

2) 10 (5
10
0