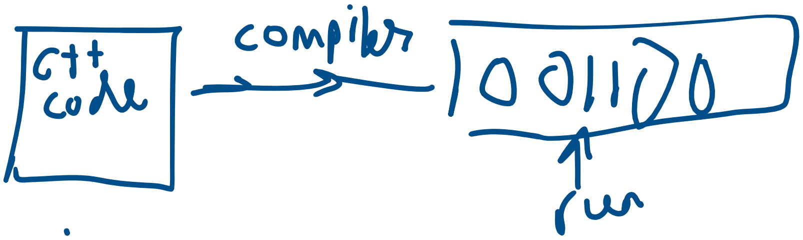
JsIntro

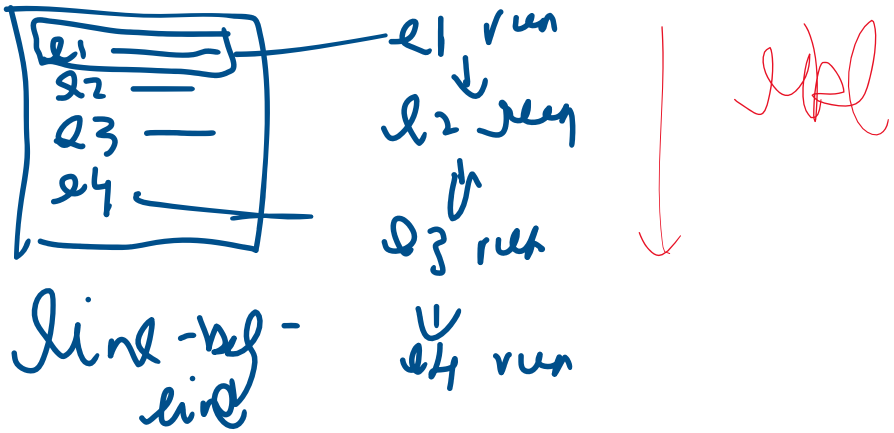
Friday, July 28, 2023 9:51 AM

JS

1 compiled X



interpreted



line-by-line

2

Statically typed and dynamically typed

languages — JS ↑

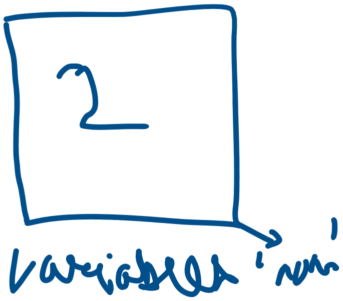
types mention ✓

C++: int x=10;

let x=10;
let y="Hi"

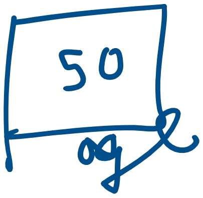
~~;~~ [; c++ ✓

; JS ✓ (optional)



variables

let age = 50



age = age+1

age+1

const pi=3.1

pi= 3.2

ERROR

at js: 19 : 4

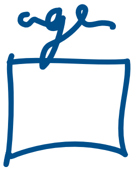
line char

line char

age + 1

50 + 1 = 51

age



modern

[let → reassign ✓

const → ✗

var → old

let

var

```
let x = 10;
let x = 100;
let x = "hi"
```

✗

redeclare ✗

″ ✓

```
f ( ) {
    ---
    --
    if (CONDN) {
        int x = 10;
        var y = 2
    }
```

→ x is out of scope ; y is IN scope

```
}
```

is out

→ y is OUT Of scope

var

function scoped

f ( ) {
var
if(...){
var
}
}

let

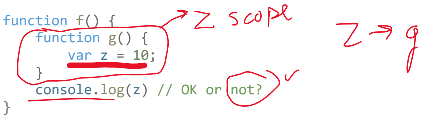
→ block scoped

if(...)
{
let x= 10
}

let → block OUT

var → func ( ) OUT

function



```
function f() {
    function g() {
        var z = 10;
    }
    console.log(z) // OK or not?
}
```

```
function f() {
    function g() {
        var z = 10;
    }
    if (10 < 8) {
        var z2 = 11;
    }
    console.log(z2);  // ok
    console.log(z) // NOT OK
}
```

→ z scope

z2 scope

User input

"hi" + "bye" → "hi bye" concat

"GM..." + "Dev" → "GM...dev"

inp → prompt
take
out → alert
show
for the user

Strings → " .... " > ok
' .... '

2^3 → 2 ** 3 = 8

5 % 3 → 2 (remainder)

'hi' + 'hi' → 'hihi' ✓

'hi' * 3 → 'hihihi' ✗

Py ✓ JS ✗

→ NaN

not a number

'hi' * 3

Num × num
not a

→ NaN

JS datatypes

C++ →

int
float
char ..

JS →

✓ boolean
✓ string "hi"
✗ char "a"
✗ integer
✓ Number
null
undef
✗ float

C++
'a' → char

let adult = true;

boolean

10
10.1 → number
-2

strings → special

+ ✓

✓ − / * %

strings → (+) ✓

× −, /, *, %

'hi' * 3 → NaN

typeof NaN ? → number

Number type → 10

→ 10.5

→ NaN

→ Infinity

−Infinity

value

Infinity > any number

int inf = 1000000 C++

Infinity > Inf → False

Inf == Inf ✓

== equality

"hi" + "hi" → hihi

"2" + "2" → "22"

special concat

special
concat

"22"

2 + "2" → "22"

"2" + 2 →

"2" == 2 → true

2+2 types same,
op ✓

$\boxed{4}$

"2" + 2 types not same
↓
convert

operator

operand

special

"2" + 2

same

↓

"2"

↓

"22"

no special

"2" + 2

↓

2

"2" - 2

↓

"2" - "2"

2

4 X

"2" - "2"

X

"2" - 2

↓

2

2 - 2

= 0

type coercion

"2" == 2

2 == 2 → true

"2" == 2

↓

2

true

don't convert

"2" === 2

triple

false

is type same?   NO

↓ yes

val same?

↓

yes → true

== vs ===

val ✓

type ✓

val ✓