ng(1),

15 Feb 26
Sunday

(14 Feb → Rearfetching)

⇒ log levels

- They revolve around console object that indicate the severity or urgency of events

Why exist
↳ separate informational msg (system working normally) from warning or errors

↳ They control the volume of logs
↳ prioritize issues based on urgency & impact

common log levels

- Trace : [Extremely detailed] → Debugging internals

Date
Page No.

console.time(label) / console.timeEnd

- Debug: [Dev focused] => Tracking variable state
- Info: [General updates, checkpoints] => System startup or login
- Warn [Potential issues, recoverable problems] => Deprecated API usage
- Error [Failures that need attention] => Exceptions, failed req

console.time()
- measures how long a block of code takes to execute

1_000_000
- writing numbers with the numeric separator ("_") does not affect performance at all.
- it treats the same as it as 1000000

- Fatal/Critical [Severe issues] => Database unavailable

console.count()

=> variables

console.table()
- handy way to display tabular data in browser's console

console.group()
- organize related log msg into a collapsible group

console.group(label)
- starts a new group with a optional label

console.groupEnd()
- ends the current group

* software 101 rule:
declare every thing as a "const", if needed then change it as "let"
more not changing things keeps code bug-free

var → no block scope

let / var → block scope

Date
Page No.

[51:00 -]

→ datatypes

→ Primitive Data Types

① Number
Ex: 42 / 3.14
Note: Includes integers/floats/ NaN/Infinity/-Infinity

② BigInt
Ex: 1234.....n
Note: For very large integers Beyond Number.MAX_SAFE_INTE

③ String
Ex: "Hello"
Note: Text enclosed in quotes (' / " / or backticks `)

④ Boolean
Ex: true / false

⑤ Undefined
Ex let x; → undefined
Note: Declared but not assigned

⑥ Null
let y = null; [Explicitly no value]

⑦ Symbol
const id = Symbol("id");
• unique identifiers, often used as object keys
• immutable

* null quirk
typeof(null) returns object
even though null is a primitive
• It is historical bug that stuck around for compatibility reasons

What is Rune?
Rune → special item which gives upgrade

Historic → letter or symbol from old Germany

→ Non-Primitive

① Object → collection of key-value pairs

const person = {
name: "Hyt",
age: 20;
}

• Arrays / functions / classes & other structures are all obj under hood.

Date
Page No.

* typeof (NaN)
  ↳ returns "Number"

* == : checks value only
  === : checks value and type

→ copying object

- To copy objects in JS we use the spread operator
  {...objectname}
- problems arises when the object contains another obj which does not include deep copy

```
const person = {
    name: "Alice";
    age: 25,
    add: {
    street: "123 main";
    zip: "10001".
    }
};
```

→ solutions of deep copies

structuredClone()

- Introduced in 2022 part of modern JS [ECMA]
- It is the built way to create deep copies of objects, arrays, & other complex data structures.
- Works with:
  object / Array / Map / Set / Date / RegExp / ArrayBuffer

◆ Key points

- Deep clone: Nested obj/Array
- Better than JSON tricks
- Limitations
  ↳ cannot clone functions / DOM nodes / class instances
- Errors:
  ↳ Throws DataCloneError

→ legacy Bug (1:30:00)
typeof null === 'object'

- both object & null share the same tag (0)
- type of null → object instead of "null"
- To check for null, you should use
  value === null

→ for checking Array use
.isArray()

→ primitives: copy by value
→ Non-primitives: copy by reference

- primitive ⇒ independent copies
- Non-primitive ⇒ shared references

→

Q what is EPSILON?

Number.EPSILON

- It is a constant that represents the difference b/w 1 & the smallest floating-point number greater than 1.

→ Key points

- used to deal with floating-point precision issues
- value approx
  2.22...318e-16

→ parseInt()
converts string into integer

→ parseFloat()
converts string into floating-point number

→ 0xA3 (hexadecimal literal)

- prefix 0x means number is written in base 16

→ Interview

0.1 + 0.2 === 0.3

↳ return false

- use Math.abs(a - b)

[1:55:00 - 2:30:00]

Date
Page No.

~~16 Feb 2025~~ Rewatch (14 Feb)
~~Monday~~

Java-Script Essentials 1

⇒ String

- length (property)
- charAt (index) → method
- at (index) [using -ve also]
- toUpperCase ( )
- toLowerCase ( )
- index ( )
- include ( )
- slice ( )
- split ( )
- join ( )
- trim ( )
- padStart ( )
- padEnd ( )

* Interview

```
console.log (void datatype);
```

void : It is often used when you want to discard a value & gurrantee **undefined**

Ex

```
console.log (void 0);
```

↳ returns undefined

```
console.log ( void "hyt");
```

→ To demolish a value in JS

```
let x = 42;
x = null;
```

↳ clears the value but variable still exists

```
x = undefined
```

↳ similar effect

Affects of null & undefined in memory

- undefined :
  default state of variable is declared but not assigned

- null :
  Explicity assignment of dev

→ Memory Effect

- Both are primitives value :
  Don't hold references to objs, so ligtweight in memory

- undefined → no extra memory beyond variable itself

- null → also just a marker, no extra memory cost

* no measurable difference in memory usage b/w them

Q Why we prefer null over undefined

- setting object reference to null helps garbage collection free memory sooner.

⇒ Conditionals