Contents

1 Function

A JavaScript function is a block of code designed to perform a particular task.

```
function echo() {
  console.log("Hello, JavaScript!");
}
```

Listing 1: function

1.1 Calling functions

The function that makes the call is known as the calling function.

```
function echo() {
  console.log("Hello, JavaScript!");
}

echo(); // Calling functions
"Hello, JavaScript!"
```

Listing 2: Calling functions

1.2 Arrow functions

An arrow function expression (previously, and now incorrectly known as **fat arrow function**) has a shorter syntax compared to function expressions and does not have its own this, arguments, super, or new.target. Arrow functions are always anonymous.

```
const echo = () => console.log("Hello, JavaScript!");
```

Listing 3: Calling functions

1.3 Named functions

function with name

2 Date

JavaScript Date objects represent a single moment in time in a platform-independent format. Date objects contain a Number that represents milliseconds since 1 January 1970 UTC.

2.1 Times and Days

2.1.1 Times

1 second	1000 milli seconds
1 mintue	60 seconds
1 hour	60 minutes
1 day	24 hours
1 week	7 days
1 month	28 - 31 days
1 year	365-366 days

Table 1: Times

1 second	1000
1 mintue	60 * 1000
1 hour	60 * 60 * 1000
1 day	24 * 60 * 60 * 1000
1 week	7 * 24 * 60 * 60 * 1000
1 month	(28 - 31) * 24 * 60 * 60 * 1000
1 year	(365-366) * 24 * 60 * 60 * 1000

Table 2: Milli seconds in times

2.1.2 Days in months

month	days
January	31 days
February	28 days in a common year and 29 days in leap years
March	31 days
April	30 days
May	31 days
June	30 days
July	31 days
August	31 days
September	30 days
October	31 days
November	30 days
December	31 days

2.1.3 AM: Anti Meridiem, PM: Post Meridiem

AM and PM both are Latin words, used in 12-hour clock system to represent Before Noon and After Noon respectively. They are also represented as A.M. and P.M.

AM expand as Anti Meridiem which means "before midday" and PM expand as Post Meridiem which means "after midday".

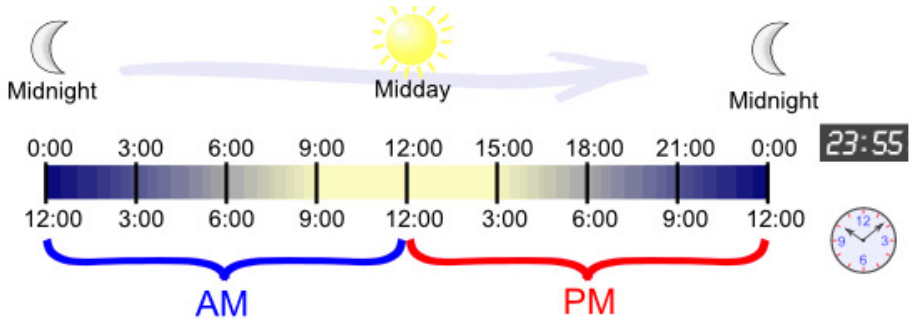

Figure 1: AM PM diagram

2.2 Create date instance

```
const date = new Date();
```

Listing 4: Date instance

2.3 Date string type methods

```
const date = new Date();
console.log(date);
// Output - type: date object
// Sun Mar 20 2022 23:40:17 GMT+0530 (India Standard Time)
```

Listing 5: Output of Date

```
const date = new Date();
const dateToStringMethod = date.toString();
console.log(dateToStringMethod);
// Output - type: string
'Sun Mar 20 2022 23:40:17 GMT+0530 (India Standard Time)'
```

Listing 6: Date to String

```
const date = new Date();
const dateToDateStringMethod = date.toDateString();
console.log(dateToDateStringMethod);
// Output - type: string
'Sun Mar 20 2022'
```

Listing 7: Date to Date String

```
const date = new Date();
const dateToLocalStringMethod = date.toLocaleString();
console.log(dateToLocalStringMethod);
// Output - type: string
'3/20/2022, 11:40:17 PM'
```

Listing 8: Date to Local String

```
const date = new Date();
const dateToUTCStringMethod = date.toUTCString();
console.log(dateToUTCStringMethod);
// Output - type: string
'Sun, 20 Mar 2022 18:10:17 GMT'
```

Listing 9: Date to UTC String

2.4 Date methods

Month **name** and its **index/value**

Index/Value	Month
0	January
1	February
2	March
3	April
4	May
5	June
6	July
7	August
8	September
9	October
10	November
11	December

Days **name** and its **index/value**

Index/Value	Day
0	Monday
1	Tuesday
2	Wednesday
3	Thursday
4	Friday
5	Saturday
6	Sunday

```
// DATE: YYYY-MM-DD: 2022-03-20
const date = new Date();
```

```
const currentDate = date.getDate();
console.log(currentDate);
// Output - type: number
// current date of month
20

const year = date.getFullYear();
console.log(year);
// Output - type: number
// current year
2022

const month = date.getMonth();
console.log(month);
// Output - type: number
// current month
// month start with 0
2

const day = date.getDay();
console.log(day);
// Output - type: number
// current day
// day start with 0
6
```

Listing 10: Date methods - I

```
// Mon Mar 21 2022 12:45:51 GMT+0530 (India Standard Time)
const date = new Date();
const hours = date.getHours();
console.log(hours);
// Output - type: number
// current hour/hours
12

const mintues = date.getMinutes();
console.log(mintues);
// Output - type: number
// current mintues
45

const seconds = date.getSeconds();
console.log(seconds);
// Output - type: number
// current seconds
```

```
51

const milliSeconds = date.getMilliseconds();
console.log(milliSeconds);
// Output - type: number
// current milli seconds
236

const time = date.getTime();
console.log(time);
// Output - type: number
// current time
1647846951236

const timeZoneOffset = date.getTimezoneOffset();
console.log(timeZoneOffset);
// Output - type: number
// current time zone offset
-330
```

Listing 11: Date methods - II

2.5 Date methods - UTC

UTC - Universal Coordinated Time

```
// DATE: YYYY-MM-DD: 2022-03-20
const date = new Date();
const currentUTCDate = date.getUTCDate();
console.log(currentUTCDate);
// Output - type: number
// current date of month
20

const yearUTC = date.getUTCFullYear();
console.log(yearUTC);
// Output - type: number
// current year
2022

const monthUTC = date.getUTCMonth();
console.log(monthUTC);
// Output - type: number
// current month
// month start with 0
2
```

```
21
22 const dayUTC = date.getUTCDay();
23 console.log(dayUTC);
24 // Output - type: number
25 // current day
26 // day start with 0
27 6
```

Listing 12: UTC Date methods - I

```
1  // Mon Mar 21 2022 16:47:01 GMT+0530 (India Standard Time)
2  const date = new Date();
3  const hoursUTC = date.getUTCHours();
4  console.log(hoursUTC);
5  // Output - type: number
6  // current hour(s)
7  11
8
9  const mintuesUTC = date.getUTCMinutes();
10 console.log(mintuesUTC);
11 // Output - type: number
12 // current mintue(s)
13 17
14
15 const secondsUTC = date.getUTCSeconds();
16 console.log(secondsUTC);
17 // Output - type: number
18 // current second(s)
19 1
20
21 const milliSecondsUTC = date.getUTCMilliseconds();
22 console.log(milliSecondsUTC);
23 // Output - type: number
24 // current milli second(s)
25 666
```

Listing 13: UTC Date methods - II

2.6 Date methods - static

2.6.1 now()

A Number representing the milliseconds elapsed since the UNIX epoch

```
1   Date.now();
2   // Output - type: number
3   // 1647864341072
```

Listing 14: Date methods - static now

2.7 Date methods - Set

2.7.1 setFullYear

year required

month optional

date optional

```
const date = new Date();
// Mon Mar 21 2022 17:06:30 GMT+0530 (India Standard Time)

// set year(s)
date.setFullYear(2025)
// Fri Mar 21 2025 17:06:30 GMT+0530 (India Standard Time)
```

Listing 15: Date methods - full year

2.7.2 setMonth

month required

date optional

set month: 0(January) - 11(December)

```
const date = new Date();
// Mon Mar 21 2022 17:10:44 GMT+0530 (India Standard Time)

// set month(s)
date.setMonth(1)
// Mon Feb 21 2022 17:10:44 GMT+0530 (India Standard Time)
```

Listing 16: Date methods - month(s)

2.7.3 setDate

date required

set date: 1 - 31
feb month - 28, 29(leap year)

```
const date = new Date();
// Mon Mar 21 2022 17:11:44 GMT+0530 (India Standard Time)

// set date(s)
date.setDate(20)
// Sun Mar 20 2022 17:11:44 GMT+0530 (India Standard Time)
```

Listing 17: Date methods - date(s)

2.7.4 setHours

hours required

mintues optional

seconds optional

milli seconds optional

set hours: 0 - 23

```
  const date = new Date();
  // Mon Mar 21 2022 17:19:51 GMT+0530 (India Standard Time)

  // set hour(s)
  date.setHours(5)
  // Mon Mar 21 2022 05:19:51 GMT+0530 (India Standard Time)
```

Listing 18: Date methods - hour(s)

2.7.5 setMinutes

mintues required

seconds optional

milli seconds optional

set mintues: 0 - 59

```
  const date = new Date();
  // Mon Mar 21 2022 17:23:53 GMT+0530 (India Standard Time)

  // set mintue(s)
  date.setMinutes(10)
  // Mon Mar 21 2022 17:10:53 GMT+0530 (India Standard Time)
```

Listing 19: Date methods - mintue(s)

2.7.6 setSeconds

seconds required

milli seconds optional

set seconds: 0 - 59

```
const date = new Date();
// Mon Mar 21 2022 17:25:50 GMT+0530 (India Standard Time)

// set second(s)
date.setSeconds(10)
// Mon Mar 21 2022 17:25:10 GMT+0530 (India Standard Time)
```

Listing 20: Date methods - second(s)

2.7.7 setMilliSeconds

milli seconds required

set milli seconds: 0 - 999

```
const date = new Date();
// date.getTime() - 1647863993085

// set milli second(s)
date.setMilliseconds(500)
// 1647863993500
```

Listing 21: Date methods - milli second(s)

2.7.8 setTime

times required

set milli seconds since January 1, 1970

```
const date = new Date();
// Mon Mar 21 2022 17:32:27 GMT+0530 (India Standard Time)
// date.getTime() - 1647863993085

// set time
date.setTime(1000)
// 1000
// Thu Jan 01 1970 05:30:01 GMT+0530 (India Standard Time)
```

Listing 22: Date methods - time

References

[1] momentjs: **Parse**, **validate**, **manipulate**, and **display** *dates* and *times* in JavaScript. [bundle details]

[2] date-fns: date-fns provides the most comprehensive, yet simple and consistent toolset for manipulating **JavaScript dates** in a **browser** & **Node.js**. [bundle details]

[3] Day.js: Fast 2kB alternative to Moment.js with the same modern API [bundle details]

[4] luxon: A powerful, modern, and friendly wrapper for JavaScript dates and times. [bundle details]

Trends - date-fns vs dayjs vs luxon vs moment
`https://www.npmtrends.com/date-fns-vs-luxon-vs-moment-vs-dayjs`