

Les objets en Javascript :

1. Introduction sur l'objet Date :

L'objet Date en JavaScript permet de manipuler des dates et des heures. Il est utilisé pour récupérer la date actuelle, effectuer des calculs sur les dates, formater l'affichage, etc.

On peut créer un objet Date avec :

```
let date = new Date(); // Date et heure actuelles  
console.log(date);
```

2. Les différentes manières de créer un objet Date

On peut créer une date de plusieurs façons :

a. new Date()

Crée un objet Date avec la date et l'heure actuelles.

```
let maintenant = new Date();  
console.log(maintenant);
```

b. new Date(année, mois, jour, heures, minutes, secondes, millisecondes)

Permet de créer une date spécifique.

⚠️ Les mois commencent à 0 (janvier = 0, février = 1, etc.).

```
let date1 = new Date(2024, 9, 10); // 10 octobre 2024  
console.log(date1);
```

c. new Date(timestamp)

Crée une date à partir d'un **timestamp** (nombre de millisecondes depuis le 1er janvier 1970).

```
let date2 = new Date(1700000000000);  
console.log(date2);
```

d. new Date(dateString)

Crée une date à partir d'une chaîne de caractères au format YYYY-MM-DD ou ISO.

```
let date3 = new Date("2024-10-10T10:30:00");
console.log(date3);
```

3. Récupérer des informations d'une date

JavaScript fournit des méthodes pour récupérer les composants d'une date :

Méthode	Description	Exemple
getFullYear()	Année complète (ex: 2024)	date.getFullYear();
getMonth()	Mois (0-11, janvier = 0)	date.getMonth();
getDate()	Jour du mois (1-31)	date.getDate();
getDay()	Jour de la semaine (0-6, dimanche = 0)	date.getDay();
getHours()	Heure (0-23)	date.getHours();
getMinutes()	Minutes (0-59)	date.getMinutes();
getSeconds()	Secondes (0-59)	date.getSeconds();
getMilliseconds() ()	Millisecondes (0-999)	date.getMilliseconds() ;

Exemple :

```
let date = new Date();
console.log("Année :", date.getFullYear());
console.log("Mois :", date.getMonth()); // Attention, janvier = 0 !
console.log("Jour :", date.getDate());
console.log("Jour de la semaine :", date.getDay());
console.log("Heure :", date.getHours());
console.log("Minutes :", date.getMinutes());
console.log("Secondes :", date.getSeconds());
```

4. Modifier une date

On peut modifier une date avec des méthodes set :

Méthode	Description	Exemple
setFullYear(année, mois, jour)	Modifie l'année	date.setFullYear(2025);
setMonth(mois)	Modifie le mois (0-11)	date.setMonth(11);
setDate(jour)	Modifie le jour du mois	date.setDate(15);
setHours(heures)	Modifie l'heure	date.setHours(14);

Exemple :

```
let date = new Date();
date.setFullYear(2025);
date.setMonth(11); // Décembre
date.setDate(25); // 25 décembre
console.log(date);
```

5. Comparer des dates

On peut comparer des dates avec les opérateurs `>`, `<`, `>=`, `<=` et `getTime()`.

```
let date1 = new Date(2024, 9, 10);
let date2 = new Date(2024, 9, 15);

if (date1 < date2) {
  console.log("date1 est avant date2");
} else {
  console.log("date1 est après date2");
}
```

6. Formatage d'une date

a. `toLocaleDateString()`

Permet d'afficher une date dans un format local.

```
let date = new Date();
console.log(date.toLocaleDateString("fr-FR")); // Ex: 10/10/2024
console.log(date.toLocaleDateString("en-US")); // Ex: 10/10/2024
```

b. `toISOString()`

Retourne la date au format ISO.

```
console.log(date.toISOString()); // Ex: 2024-10-10T10:30:00.000Z
```

QCM sur l'objet Date

1. Que retourne `getMonth()` pour le mois de décembre ?

- A) 12
- B) 11
- C) 10
- D) 1

2. Quel est le résultat de `new Date(0)` ?

- A) La date actuelle
- B) 1er janvier 1970
- C) 31 décembre 1969
- D) Erreur

3. Quelle méthode permet d'obtenir l'année d'une date ?

- A) `getYear()`
- B) `getFullYear()`
- C) `getDate()`
- D) `getTime()`

4. Que retourne l'expression suivante ?

```
let date = new Date("2024-02-29");
console.log(date.getMonth());
```

- A) 1
- B) 2

- C) 29
- D) Erreur

5. Quelle méthode permet de récupérer le timestamp en millisecondes d'une date donnée ?

- A) getMilliseconds()
- B) getUnixTime()
- C) getTime()
- D) toTimestamp()

6. Que va afficher le code suivant ?

```
let date1 = new Date(2024, 0, 31);
date1.setDate(32);
console.log(date1);
```

- A) 1er mars 2024
- B) 1er février 2024
- C) Erreur
- D) 32 janvier 2024

7. Que renvoie Date.parse("2024-12-31") ?

- A) Un objet Date
- B) Une chaîne de caractères
- C) Un timestamp en milliseconds
- D) null

8. Comment obtenir l'année à deux chiffres d'une date ?

- A) date.getFullYear()
- B) date.getFullYear() % 100
- C) date.toString().slice(-2)
- D) date.getShortYear()

9. Que se passe-t-il si on passe un nombre négatif à new Date() ?

- A) Erreur
- B) Retourne null
- C) Retourne une date avant 1970
- D) La date actuelle

10. Quelle est la sortie de ce code ?

```
let date = new Date("2024-10-10T12:00:00Z");
console.log(date.toLocaleString("fr-FR"));
```

- A) "10/10/2024, 12:00:00"
- B) "10 octobre 2024, 12:00:00"
- C) "10/10/2024, 14:00:00"
- D) "10 octobre 2024, 14:00:00"

11. Quelle est la sortie de ce code ?

```
let date = new Date("February 30, 2024");
console.log(date);
```

- A) 1er mars 2024
- B) Erreur
- C) 30 février 2024
- D) Invalid Date

12. Que fait setUTCFullYear(2025, 11, 25) sur un objet Date ?

- A) Définit uniquement l'année
- B) Définit année, mois et jour en UTC
- C) Définit l'année en UTC mais pas le reste
- D) Crée une nouvelle date

13. Comment calculer le nombre de jours entre deux dates ?

```
let d1 = new Date("2024-01-01");
let d2 = new Date("2024-02-01");
let diff = d2.getTime() - d1.getTime();
console.log(diff / (1000 * 60 * 60 * 24));
```

- A) 31
- B) 30
- C) 1
- D) Erreur

14. Que va afficher ce code ?

```
let date = new Date();
console.log(date instanceof Date);
```

- A) true
- B) false
- C) Erreur
- D) undefined

15. Que retourne new Date(2024, 12, 1) ?

- A) 1er janvier 2025
- B) 1er décembre 2024
- C) 1er novembre 2024
- D) Erreur

16. Que se passe-t-il si setMonth(-1) est appelé sur une date ?

- A) Erreur
- B) Fixe le mois à janvier
- C) Décrémente l'année et met décembre
- D) Fixe le mois à novembre

17. Quelle méthode permet de récupérer l'heure au format UTC ?

- A) getUTCHours()
- B) getUniversalHours()
- C) getGlobalHours()
- D) getGMTTime()

18. Que retourne Date.now() ?

- A) Une date
- B) Un timestamp en secondes
- C) Un timestamp en millisecondes
- D) undefined

Exercices pratiques :

Exercice 1 :

Vous devez créer un **simulateur de calcul sur les dates** avec un formulaire permettant à l'utilisateur de :

1. **Sélectionner une date de départ** .
2. **Saisir un nombre de jours à ajouter ou soustraire** .
3. **Afficher la nouvelle date résultante au format ISO** .
4. **Calculer la différence entre deux dates sélectionnées.**

Créez un formulaire HTML avec :

- Un champ **input** pour choisir une **date** (`input type="date"`).
- Un champ **input** pour entrer un **nombre de jours** (`input type="number"`).
- Deux boutons :
 - Un bouton pour **ajouter des jours**.
 - Un bouton pour **soustraire des jours**.

- Un second champ **input** pour une **deuxième date** (optionnel, pour calculer la différence entre deux dates).
- Un bouton pour **calculer la différence** entre les deux dates.

Utilisez JavaScript pour gérer les événements et effectuer les calculs :

- Ajouter ou soustraire des jours à la date saisie.
- Afficher la nouvelle date au format **ISO 8601** (`toISOString()`).
- Calculer le nombre de jours entre deux dates.

Affichez les résultats dynamiquement dans la page.

14 Simulateur de Calcul sur les Dates

Sélectionnez une date :

Nombre de jours/mois/années :

Type :

 Ajouter
 Soustraire

Nouvelle date : -

vs Comparer deux dates

Deuxième date :

Différence : -

Calcul de l'âge

Date de naissance :

Âge : -

Conversion de Date

- ISO : -
- Locale : -
- UTC : -