educación IT
CODE YOUR FUTURE

JavaScript desde cero

Módulo 2

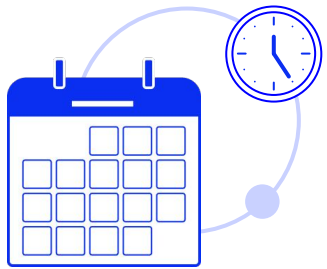
Objeto fecha

Objeto fecha: clase *Date*

La clase **`Date`** es un objeto que forma parte del Core JavaScript. Permite **manipular fechas y horas en los desarrollos**.

Este objeto es *instanciable*. Esto significa que permite "crear múltiples copias" de ese objeto, de acuerdo a la necesidad de la aplicación.

Al ser instanciable, la clase **`Date()`** puede ser utilizada para **crear múltiples fechas y horas diferentes** y, por ejemplo, **calcular períodos de tiempo** entre estas.



Ejemplo

En este ejemplo de código, vemos la clase **`Date()`** instanciada con la fecha indicada en formato ***Año, Mes, Día***.

```
const fechaInicial = new Date(1996, 7, 14);
```

Por diferentes nombres de constantes, **podremos instanciar múltiples veces la clase `Date()`**.

¿Cuál es el objetivo?

Poder calcular la unidad de tiempo que transcurrió entre una fecha y la otra.

```
const fechaInicial = new Date(1996, 7, 14);

const fechaFinal = new Date(2005, 8, 30);
```

Cada fecha instanciada permite utilizar cada parte que compone su unidad de tiempo, de forma individual.

Esto se logra con los diferentes métodos disponibles.

```
const fechaInicial = new Date(1996, 7, 14);

fechaInicial.getDate()
fechaInicial.getDay()
fechaInicial.getMonth()
fechaInicial.getFullYear()
```

Métodos

Los métodos retornarán la unidad de tiempo individual, de la fecha, aunque algunos de ellos tienen particularidades.

Método	Descripción
`.getDate()`	Retorna el día del mes (1-31) de la fecha.
`.getDay()`	Retorna el día de la semana (0-6) de la fecha, donde **0** es domingo y **6** es sábado.
`.getMonth()`	Retorna el mes (0-11) de la fecha, donde **0** es enero y **11** es diciembre.
`.getFullYear()`	Retorna el año (año completo) de la fecha.

Ejemplo: método *getMonth()*

El caso de la fecha instanciada corresponde al día **14 de Julio**, pero el método **`getMonth()`** retorna el número de mes, a través de una estructura interna que asigna el valor **`0`** para el mes de enero, y el valor **`11`** para el mes de diciembre.

Esto a veces causa confusión y por eso, es necesario tener **precaución si se va a obtener y representar un mes con el número en pantalla**.

```
const fechaInicial = new Date(1996, 7, 14);

fechaInicial.getMonth()    // retorna Agosto
```

Si necesitamos mostrar el número de mes en la aplicación, simplemente **se le suma 1 al valor que retorna `getMonth()`**. Así, se resuelve el tema.

```
const fechaInicial = new Date(1996, 7, 14);
      fechaInicial.getMonth() + 1;
```

Ejemplo: método *.toLocaleString()*

Otra alternativa al ejemplo visto en las últimas diapositivas, es instanciar la fecha utilizando el **formato *ISO DATE*** y luego recurrir al método **`.toLocaleString()`**, que nos permite formatear el parámetro de salida de la información. Así, se consigue ver el mes correspondiente en **formato nombre**.

```
const fechaInicial = new Date("1996-07-14");

const opciones = { month: 'long'};
const fechaFormateada = fechaInicial.toLocaleString('es-AR', opciones);
      //retornará 'julio'
```

Este es un ejemplo completo del uso del método **`.toLocaleString()`** para obtener: el valor de cada unidad de tiempo de una fecha, formateado a una configuración regional específica y a un tipo de datos también específico:

```
const fechaInicial = new Date("1996-07-14");

const opcionesMes = { month: 'long' };
const opcionesDia = { weekday: 'long' };
const opcionesFecha = { day: 'numeric' };
const opcionesAnio = { year: 'numeric' };

fechaInicial.toLocaleString('es-AR', opcionesMes);    // julio
fechaInicial.toLocaleString('es-AR', opcionesDia);    // sábado
fechaInicial.toLocaleString('es-AR', opcionesFecha);  // 14
fechaInicial.toLocaleString('es-AR', opcionesAnio);   // 1996
```

Ejemplo: método *.getTime()*

El método **`.getTime()`** retorna la fecha instanciada en **formato milisegundos**.

Para calcular un período de **tiempo entre dos fechas**:

1. Se pueden instanciar, y luego restar la fecha más reciente a la fecha más lejana, mediante la invocación de este método.
2. Luego, a la diferencia retornada en milisegundos, de la divide por ***1000*** (1 segundo = 1000 milisegundos). Finalmente, se multiplica por el valor de tiempo máximo de cada unidad de tiempo, hasta llegar a la unidad de tiempo deseada.

```
let fecha1 = new Date('2023-03-28');
let fecha2 = new Date('2023-04-05');

const diferencia = fecha2.getTime() - fecha1.getTime();
const dias = (diferencia / (1000 * 60 * 60 * 24));

console.log(dias + ' días');
```

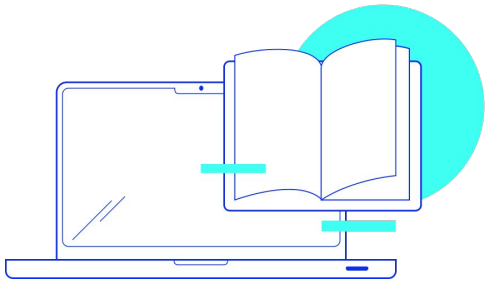
3. La diferencia la dividimos por el **número de milisegundos en un día** (1000 milisegundos por segundo * 60 segundos por minuto * 60 minutos por hora * 24 horas por día), para obtener la **diferencia en días**.

4. Para calcular la **diferencia en semanas**, **meses y años**, podemos usar un enfoque similar pero dividir la diferencia en milisegundos por el número de milisegundos en una semana, mes o año, respectivamente.

```
const diferencia = fecha2.getTime() - fecha1.getTime();
const dias = (diferencia / (1000 * 60 * 60 * 24));
```

Revisión

- Repasar el concepto de **objeto en JavaScript**.
- Implementar la clase **`Date`** con las diferentes posibilidades del **objeto fecha**.
- **Combinar los diferentes métodos** para extraer una unidad de tiempo específica de una fecha.
- Realizar cálculos entre diferentes fechas para **obtener el tiempo transcurrido**.



¡Sigamos trabajando!