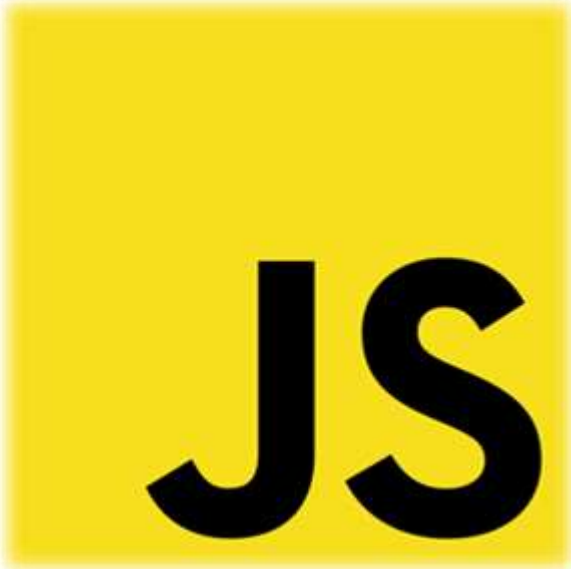

JavaScript

Módulo 9

1. El objeto nativo window: métodos y propiedades
2. El objeto nativo document: métodos y propiedades
3. El objeto nativo date: métodos y propiedades
4. El objeto nativo history: métodos y propiedades
5. El objeto nativo location: métodos y propiedades
6. Objetos personalizados: objetos literales, funciones constructoras y Clases de objetos.

Objeto nativo window: propiedades y métodos.

El objeto window representa una ventana abierta en un navegador.

En un navegador con pestañas, como Firefox, cada pestaña contine su propio objeto window.

Esto significa que el objeto window no se comparte entre diferentes pestañas de la misma ventana del navegador.

Objeto nativo window: propiedades.

Propiedad	Descripción
closed	Retorna un booleano **true** si una ventana está cerrada.
console	Retorna el objeto **Console** de la ventana.
document	Retorna el objeto **Document** de la ventana.
history	Retorna el objeto **History** de la ventana.
localstorage	Retorna el objeto **localstorage** usado para almacenar datos que pueden ser accedidos por el origen que los creo.

Objeto nativo window: métodos.

Method	Descripción
alert()	Muestra una caja de alerta con un mensaje y un botón que dice OK.
atob()	Método decodifica una cadena codificada en base 64.
blur()	Método elimina el foco de una ventana.
btoa()	Método que codifica una cadena en base-64.
clearInterval()	Método borra un temporizador establecido con el **setInterval()** método.
clearTimeout()	Método borra un temporizador establecido con el setTimeout()método.
close()	Cierra una ventana establecida.

Ejercicios.

1. Crear una función abrirVentana() que abra una ventana, otra función cerrarVentana() que cierre una función y luego otra función checarVentana() que revise si la ventana está abierta o no.

2. Crear un reloj dinámico usando clearInterval() y setInterval().

Objeto nativo window: propiedades y métodos.

Existen muchas propiedades más y muchos métodos más, referencia completa de propiedades y métodos:

https://developer.mozilla.org/es/docs/Web/API/Window

Objeto nativo document: propiedades y métodos.

El objeto **document** representa su página web.
Si desea acceder a cualquier elemento en una página HTML, siempre comienza accediendo al objeto del documento.

Objeto nativo document: métodos (Encontrar elementos HTML).

Método	Descripción
document.**getElementById**(*id*)	Encuentra un elemento por su id.
document.**getElementsByTagName**(*nombre*)	Encuentra un elemento por su etiqueta nombre.
document.**getElementsByClassName**(*nombre*)	Encuentra elementos por el nombre de su clase.

Objeto nativo document: propiedades (cambio de elementos HTML).

Propiedad	Descripción
elemento.**innerHTML**=*nuevo_contenido_HTML*	Cambia el interior de un elemento HTML.
elemento.***attribute*** = *nuevo_valor*	Cambia el valor de atributo de un elemento HTML.
elemento.**style**.*propiedad = nuevo_estilo*	Cambia el estilo de un elemento HTML.
Método (setter)	**Descripción**
elemento.**setAttribute**(*attribute, value*)	Cambia el valor de atributo de un elemento HTML.

Objeto nativo document: propiedades (adición y eliminación de elementos HTML).

Método	Description
document.**createElement**(*elemento*)	Crea un elemento HTML.
document.**removeChild**(*elemento*)	Elimina un elemento HTML.
document.**appendChild**(*elemento*)	Agrega un elemento HTML.
document.**replaceChild**(*nuevo, viejo*)	Reemplaza un elemento HTML.
document.**write**(*texto*)	Escribe dentro de la pantalla de salida de HTML.

Ejercicio.

1. Crear un efecto visual cambiando el color de el titulo de una página web.

Objeto nativo document: método y propiedades.

Existen algunas cuantas propiedades y métodos más del objeto document.

https://developer.mozilla.org/es/docs/Web/API/Document

El objeto nativo date: propiedades y métodos.

El objeto Date nos permite trabajar con fechas:

```
7
8   const fecha = new Date();
9
```

Así creamos un nuevo objeto de Date con la fecha y hora actuales.

Métodos de obtención de fecha.

Method	Description
getFullYear()	Obtiene el **año** a 4 dígitos (aaaa)
getMonth()	Obtiene el **mes** como un número (0-11)
getDate()	Obtiene el **día** como un número (1-31)
getHours()	Obtiene la **hora**. (0-23)
getMinutes()	Obtiene el **minuto** (0-59)
getSeconds()	Obtiene los **segundos** (0-59)
getMilliseconds()	Obtiene los **milisegundos** (0-999)
getTime()	Obtiene el **tiempo** (milisegundos desde el 1ro de enero de 1970)
getDay()	Obtiene la **semana** como un número (0-6)
Date.now()	Obtiene el **tiempo**. ECMAScript 5.

El objeto nativo history: propiedades y métodos.

El objeto window.history contiene el historial de los navegadores.

El objeto history se puede escribir sin el prefijo de window.

Para proteger la privacidad de los usuarios, existen limitaciones sobre cómo JavaScript puede acceder a este objeto.

El objeto nativo history: propiedades y métodos.

history.back() método carga la URL anterior en la lista de historial, es lo mismo que hacer clic en el botón Atrás en el navegador.

history.forward() método carga la siguiente URL en la lista de historial, es lo mismo que hacer clic en el botón Adelante en el navegador.

Ejercicios.

1. Crear un botón atrás de una página web.

2. Crear un botón delante de una página web.

El objeto nativo location: propiedades y métodos.

El objeto window.location se puede utilizar para obtener la dirección de la página actual (URL) y para redirigir el navegador a una nueva página.

El objeto location se puede escribir sin el prefijo de window.

El objeto nativo location: propiedades y métodos.

window.location.href devuelve el href (URL) de la página actual.

window.location.hostname devuelve el nombre de dominio del servidor web.

window.location.pathname devuelve la ruta y el nombre de archivo de la página actual.

window.location.protocol devuelve el protocolo web utilizado (http: o https:).

window.location.assign() carga un nuevo documento.

Ejercicios.

1. Generar una función informeWeb() que de un informe breve de nuestro sitio web.

Objetos personalizados.

Con JavaScript, podemos definir y crear nuestro propios objetos.

Hay diferentes formas de crear nuevos objetos:

- Utilizando un objeto literal.
- Con la palabra reservada new.
- Defina un constructor de objetos y luego cree objetos del tipo construido.
- Usando Object.create().

Objetos literales.

Esta es la forma más fácil de crear un objeto JavaScript. Usando un objeto literal, usted define y crea un objeto en una declaración. Se aceptan saltos de línea.

```
12
13  const persona = {nombre:"Reus", apellido:"Campos", edad: 30};
14
```

Funciones constructoras.

Se considera una buena práctica nombrar funciones constructoras con una primera letra en mayúscula.

```
17  function Persona(nombre, apellido, edad) {
18      this.nombre = nombre;
19      this.apellido = apellido;
20      this.edad = edad;
21  }
```

Clases de objetos.

ECMAScript 2015, también conocido como ES6, introdujo clases de JavaScript.

Las clases de JavaScript son plantillas para objetos de JavaScript.

Utilizamos la palabra reservada **class** para crear una clase.
Siempre debemos agregar un método llamado **constructor()**.

Hagamos un ejemplo.

Ejercicios.

1. Crear la clase Automovil con un método mostrar(), crear 3 objetos de la clase Automovil y mostrarlos en la pantalla del navegador.