**Que es el objeto console (de un ejemplo)**

El objeto console es una característica integrada en los navegadores web y en el entorno de ejecución de JavaScript que proporciona métodos para interactuar con la consola del navegador y mostrar mensajes de depuración, advertencias, errores, etc. Es muy útil para desarrolladores durante el proceso de depuración y prueba de aplicaciones web.

// Ejemplo de uso del objeto console

let nombre = "Juan";

let edad = 30;

let ciudad = "Barcelona";

console.log("Bienvenido a nuestra aplicación.");

console.log("Nombre:", nombre);

console.log("Edad:", edad);

console.warn("¡Atención! Esta es una advertencia.");

console.error("Se ha producido un error en la aplicación.");

console.info("Información adicional sobre el proceso.");

console.table({ Nombre: nombre, Edad: edad, Ciudad: ciudad });

En este ejemplo, se utilizan varios métodos del objeto **console**:

* **console.log()**: Muestra mensajes informativos en la consola.
* **console.warn()**: Muestra mensajes de advertencia en la consola.
* **console.error()**: Muestra mensajes de error en la consola.
* **console.info()**: Muestra mensajes de información en la consola.
* **console.table()**: Muestra datos en forma de tabla en la consola.

**Que es el objeto date (de un ejemplo)**

El objeto Date en JavaScript se utiliza para trabajar con fechas y horas. Proporciona métodos para obtener y establecer diferentes componentes de la fecha y la hora, como el año, el mes, el día, la hora, el minuto, el segundo, etc.

// Ejemplo de uso del objeto Date

let fechaActual = new Date();

// Obtener la fecha y hora actual

console.log("Fecha y hora actual:", fechaActual);

// Obtener el año actual

let anio = fechaActual.getFullYear();

console.log("Año actual:", anio);

// Obtener el mes actual (los meses se indexan desde 0)

let mes = fechaActual.getMonth() + 1;

console.log("Mes actual:", mes);

// Obtener el día del mes

let dia = fechaActual.getDate();

console.log("Día del mes:", dia);

// Obtener el día de la semana (0 para Domingo, 1 para Lunes, ..., 6 para Sábado)

let diaSemana = fechaActual.getDay();

console.log("Día de la semana:", diaSemana);

// Obtener la hora actual

let hora = fechaActual.getHours();

console.log("Hora actual:", hora);

// Obtener el minuto actual

let minuto = fechaActual.getMinutes();

console.log("Minuto actual:", minuto);

// Obtener el segundo actual

let segundo = fechaActual.getSeconds();

console.log("Segundo actual:", segundo);

En este ejemplo, primero se crea un objeto **Date** llamado **fechaActual**, que representa la fecha y hora actuales. Luego, se utilizan varios métodos del objeto **Date** para obtener diferentes componentes de la fecha y la hora, como el año, el mes, el día, la hora, el minuto y el segundo.

**Que es el objeto Math (de un ejemplo)**

El objeto Math en JavaScript proporciona un conjunto de funciones y constantes matemáticas predefinidas que se pueden utilizar para realizar operaciones matemáticas comunes. Estas funciones incluyen funciones trigonométricas, funciones exponenciales, funciones de redondeo, funciones de generación de números aleatorios, entre otras.

// Ejemplo de uso del objeto Math

// Obtener el valor absoluto de un número

let numero1 = -10;

let valorAbsoluto = Math.abs(numero1);

console.log("Valor absoluto de", numero1, ":", valorAbsoluto);

// Redondear un número hacia abajo al entero más cercano

let numero2 = 3.7;

let redondeoAbajo = Math.floor(numero2);

console.log("Redondeo hacia abajo de", numero2, ":", redondeoAbajo);

// Redondear un número hacia arriba al entero más cercano

let numero3 = 3.2;

let redondeoArriba = Math.ceil(numero3);

console.log("Redondeo hacia arriba de", numero3, ":", redondeoArriba);

// Obtener el máximo de dos números

let maximo = Math.max(5, 10);

console.log("Máximo entre 5 y 10:", maximo);

// Obtener el mínimo de dos números

let minimo = Math.min(5, 10);

console.log("Mínimo entre 5 y 10:", minimo);

// Generar un número aleatorio entre 0 (inclusive) y 1 (exclusivo)

let aleatorio = Math.random();

console.log("Número aleatorio:", aleatorio);

En este ejemplo, se utilizan varios métodos del objeto **Math**:

* **Math.abs()**: Devuelve el valor absoluto de un número.
* **Math.floor()**: Redondea un número hacia abajo al entero más cercano.
* **Math.ceil()**: Redondea un número hacia arriba al entero más cercano.
* **Math.max()**: Devuelve el máximo de los argumentos dados.
* **Math.min()**: Devuelve el mínimo de los argumentos dados.
* **Math.random()**: Devuelve un número aleatorio entre 0 (inclusive) y 1 (exclusivo).

**Que es el operador de cortocircuito en javascript (de un ejemplo)**

El operador de cortocircuito en JavaScript es una técnica que se utiliza en expresiones lógicas para mejorar la eficiencia y evitar la evaluación innecesaria de operandos. Cuando se encuentra un operador de cortocircuito, la evaluación se detiene tan pronto como se determina el resultado final de la expresión, sin necesidad de evaluar los operandos restantes.

Los dos operadores de cortocircuito en JavaScript son **&&** (AND lógico) y **||** (OR lógico).

**Operador && (AND lógico)**:

* Si el primer operando es falso, la expresión se evalúa como falsa y se detiene la evaluación. Si el primer operando es verdadero, la expresión se evalúa utilizando el segundo operando.

// Ejemplo de operador de cortocircuito &&

let resultado = false && hacerAlgo(); // Como el primer operando es falso, no se evalúa la función hacerAlgo()

console.log(resultado); // Imprime: false

function hacerAlgo() {

console.log("Haciendo algo...");

return true;

}

En este ejemplo, como el primer operando de **&&** es falso, la función **hacerAlgo()** nunca se llama y no se imprime "Haciendo algo...". Esto muestra cómo el operador de cortocircuito **&&** evita la evaluación innecesaria cuando el primer operando ya determina el resultado final.

**Operador || (OR lógico)**:

* Si el primer operando es verdadero, la expresión se evalúa como verdadera y se detiene la evaluación. Si el primer operando es falso, la expresión se evalúa utilizando el segundo operando.

// Ejemplo de operador de cortocircuito ||

let resultado = true || hacerAlgo(); // Como el primer operando es verdadero, no se evalúa la función hacerAlgo()

console.log(resultado); // Imprime: true

function hacerAlgo() {

console.log("Haciendo algo...");

return false;

}

En este ejemplo, como el primer operando de **||** es verdadero, la función **hacerAlgo()** nunca se llama y no se imprime "Haciendo algo...". Esto demuestra cómo el operador de cortocircuito **||** evita la evaluación innecesaria cuando el primer operando ya determina el resultado final.

**Que es el alert, confirm y pormpot (de un ejemplo de cada una)**

**alert():**

* La función alert() muestra un cuadro de diálogo con un mensaje y un botón de "Aceptar". Es útil para mostrar mensajes de alerta o información importante al usuario.

// Ejemplo de alert

alert("¡Hola! Este es un mensaje de alerta.");

**confirm()**:

* La función **confirm()** muestra un cuadro de diálogo con un mensaje, un botón de "Aceptar" y un botón de "Cancelar". Devuelve **true** si el usuario hace clic en "Aceptar" y **false** si hace clic en "Cancelar".

// Ejemplo de confirm

let respuesta = confirm("¿Estás seguro de que quieres continuar?");

console.log(respuesta); // true si se hace clic en "Aceptar", false si se hace clic en "Cancelar"

**prompt()**:

* La función **prompt()** muestra un cuadro de diálogo con un mensaje, un campo de entrada de texto y botones de "Aceptar" y "Cancelar". Devuelve el valor ingresado por el usuario como una cadena de texto, o **null** si se hace clic en "Cancelar".

// Ejemplo de prompt

let nombre = prompt("Por favor, ingresa tu nombre:");

console.log("Hola, " + nombre + "! Bienvenido.");

**Que son las expresiones regulares (de un ejemplo)**

Las expresiones regulares, también conocidas como regex, son patrones de búsqueda y manipulación de texto utilizados en programación. Permiten realizar búsquedas avanzadas, validaciones y manipulaciones de cadenas de texto de manera muy flexible y poderosa. Las expresiones regulares se componen de caracteres literales y metacaracteres que representan clases de caracteres, repeticiones, posiciones, etc.

// Ejemplo de expresión regular para buscar números de teléfono

let texto = "Mi número de teléfono es 123-456-7890. Llámame.";

let patron = /\d{3}-\d{3}-\d{4}/; // Expresión regular para buscar números de teléfono en formato ###-###-####

let resultado = patron.exec(texto);

if (resultado) {

console.log("Número de teléfono encontrado:", resultado[0]);

} else {

console.log("Número de teléfono no encontrado.");

}

En este ejemplo, la expresión regular **/d{3}-\d{3}-\d{4}/** busca un patrón de tres dígitos seguido de un guion, tres dígitos más seguido de otro guion, y cuatro dígitos más. Cuando se ejecuta esta expresión regular en el texto, encuentra el número de teléfono "123-456-7890".

**Que son las funciones anónimas autoejecutables (de un ejemplo)**

// Ejemplo de función anónima autoejecutable

(function() {

let mensaje = "¡Hola desde la función anónima autoejecutable!";

console.log(mensaje);

})();

En este ejemplo, se define una función anónima entre paréntesis **(function() { ... })** y se agrega un par de paréntesis adicionales **( ... )** al final de la declaración de la función para ejecutarla inmediatamente. Dentro de la función, se imprime un mensaje en la consola.

**Que son los módulos (Import y export) (de un ejemplo)**

Los módulos en JavaScript son una forma de organizar y reutilizar código al dividirlo en archivos separados. Los módulos permiten definir variables, funciones, clases y otros recursos en un archivo y luego exportarlos para que estén disponibles para su uso en otros archivos. Además, los módulos pueden importar recursos de otros archivos para utilizarlos en su propio código. Aquí tienes un ejemplo de cómo se utilizan los módulos con las declaraciones import y export:

Supongamos que tenemos dos archivos: **modulo1.js** y **modulo2.js**.

En **modulo1.js**, definimos una función y la exportamos para que esté disponible para otros módulos:

// modulo1.js

export function saludar(nombre) {

console.log(`¡Hola, ${nombre}!`);

}

En **modulo2.js**, importamos la función **saludar** del módulo **modulo1.js** y la utilizamos:

// modulo2.js

import { saludar } from './modulo1.js';

saludar('Juan');

En este ejemplo, **modulo2.js** importa la función **saludar** del módulo **modulo1.js** utilizando la declaración **import { saludar } from './modulo1.js';**. Luego, invoca la función **saludar('Juan');** para imprimir "¡Hola, Juan!" en la consola.