Tema 8

JavaScript

Objectius

- Comprendre els conceptes fonamentals de la programació en JavaScript, incloent-hi la sintaxi bàsica i l' estructura d' un programa.
- Dominar el maneig de variables, tipus de dades i operadors a JavaScript per realitzar càlculs i emmagatzemar informació de manera eficient.
- Aprendre les estructures de control a JavaScript, incloent condicionals i bucles, per prendre decisions i repetir accions segons les necessitats del programa.
- Explorar els esdeveniments a JavaScript i com gestionar-los per interactuar amb elements del DOM, permetent la creació d'aplicacions interactives.
- Comprendre la importància del Document Object Model (DOM) i com accedir a elements en una pàgina web utilitzant JavaScript.
- Aprendre a modificar el contingut i els atributs d' elements del DOM, així com crear i eliminar elements dinàmicament per manipular l' estructura de la pàgina web.

Introducció

Després d'haver explorat les bases d'HTML i CSS, ara ens endinsarem en l'emocionant món de JavaScript. Aquest llenguatge de programació, àmpliament utilitzat en el desenvolupament web, ens permetrà donar vida a les nostres pàgines web en agregar interactivitat i dinamisme. Al llarg d'aquesta secció, aprendrem com JavaScript s'integra amb HTML i CSS per crear experiències web més riques i atractives, permetent-nos controlar el comportament dels elements de la pàgina i respondre a les accions de l'usuari.

Índex

8.1 INTRODUCCIÓ A LA PROGRAMACIÓ EN JAVASCRIPT

JavaScript és un llenguatge de programació que exerceix un paper essencial en la creació d'experiències web interactives i dinàmiques. Quan explorem la programació en el context de la informàtica i la web, JavaScript destaca com una de les eines més influents i omnipresents.

Què és JavaScript?

JavaScript, sovint abreujat com a JS, és un llenguatge de programació d' alt nivell, interpretat i **orientat a objectes**. A diferència d'HTML (HyperText Markup Language) i CSS (Cascading Style Sheets), que s'utilitzen principalment per estructurar i donar estil a les pàgines web, **JavaScript s'utilitza per agregar interactivitat i comportaments dinàmics a les pàgines web.**
Tot i que JavaScript comparteix el seu nom amb Java, un altre llenguatge de programació, no tenen gaire en comú en termes de sintaxi o funcionalitat. JavaScript va ser dissenyat específicament per a la web i s'executa al navegador web de l'usuari final.

Importància de JavaScript a la Tripleta HTML+CSS+JS

La tripleta HTML+CSS+JS és el nucli de la programació web moderna. Aquí hi ha la raó per la qual cadascun d' aquests components és crucial:

1. **HTML (HyperText Markup Language)**: Proporciona l'estructura i el contingut d'una pàgina web. Defineix els elements i la seva disposició a la pàgina.

2. **CSS (Cascading Style Sheets)**: Controla la presentació i l'estil de la pàgina web. Determina com es veuran els elements HTML.

3. **JavaScript**: Agrega interactivitat i comportaments a la pàgina. Permet que els elements HTML i CSS responguin a les accions de l' usuari i realitzin operacions complexes a la banda del client.

La combinació d'aquests tres elements permet crear llocs web moderns i atractius que van més enllà de la simple presentació d'informació estàtica. JavaScript exerceix un paper crucial en permetre'ns crear aplicacions web interactives, jocs, formularis dinàmics i moltes altres característiques que milloren l'experiència de l'usuari.

Importància de JavaScript en el Desenvolupament Web

La importància de JavaScript en el desenvolupament web és innegable i es deriva de diverses característiques clau:

1. **Interactivitat de l'Usuari**: JavaScript permet als desenvolupadors crear aplicacions web interactives que responen a les accions de l'usuari. Això inclou formularis que validen dades en temps real, menús desplegables, carrusells d'imatges i molt més.

2. **Manipulació del DOM**: El Document Object Model (DOM) representa l'estructura d'una pàgina web i JavaScript permet als desenvolupadors accedir i manipular dinàmicament el DOM. Això significa que es poden agregar, eliminar o modificar elements HTML i CSS en resposta a esdeveniments o accions de l'usuari.

3. **Comunicació amb el Servidor**: JavaScript facilita les sol·licituds asíncrones al servidor a través de la tecnologia AJAX (Asynchronous JavaScript and XML), la qual cosa permet carregar dades en segon pla sense recarregar la pàgina completa. Això s'utilitza àmpliament en aplicacions web modernes per millorar la velocitat i l'eficiència.

4. **Ampli Ecosistema**: Existeix una vasta col·lecció de biblioteques i marcs de treball (com jQuery, React, Angular i Vue.js) construïts sobre JavaScript que faciliten el desenvolupament web. Aquestes eines acceleren el procés de desenvolupament i ajuden a crear aplicacions web avançades de manera eficient.

El Poder de JavaScript en Acció

Quan dominem JavaScript, podem crear una àmplia varietat d'aplicacions i efectes web:

1. **Validació de Formularis en Temps Real**: Verificar dades ingressades en formularis abans del seu enviament.
2. **Animacions i Transicions Suaus**: Crear animacions i efectes visuals atractius.
3. **Carrusells d' Imatges Interactius**: Desenvolupar components lliscs d' imatges dinàmics.
4. **Jocs en Línia**: Construir jocs web simples i entretinguts.
5. **Actualitzacions de Contingut en Temps Real**: Carregar dades i actualitzar la pàgina sense necessitat de recarregar-la.
6. **Aplicacions Web Complexes**: Crear aplicacions web complexes amb funcionalitat d' usuari avançada.

Atès el curt espai de temps que es disposa en aquest curs, durant la primera part del tema estudiarem conceptes bàsics de programació en concret en javaScript i seguidament explorarem les capacitats de JavaScript pel que fa a **manipulació del DOM** i aprendrem a utilitzar aquest poderós llenguatge per crear experiències web riques i interactives.

8.2 VARIABLES, TIPUS DE DADES I OPERADORS

En aquest segon apartat, ens submergirem en els fonaments de JavaScript, explorant la declaració de variables, els tipus de dades disponibles i com utilitzar operadors aritmètics i lògics. Aquests conceptes són essencials per comprendre com funciona JavaScript i com es poden realitzar operacions i manipulacions de dades en el llenguatge.

Declaració de Variables a JavaScript

Les variables són elements fonamentals en qualsevol llenguatge de programació, i JavaScript no n'és l'excepció. Les variables s' utilitzen per emmagatzemar i gestionar dades en un programa. A JavaScript, podem declarar variables utilitzant les paraules clau **let** i **const**.

```
Usando let
let saldo = 1000;
let missatge = "¡Hola, món!";

Usando const (per a valors constants)
const PI = 3.14159265359;
const URL_API = "https://api.example.com";
```

Tipus de Dades a JavaScript

JavaScript admet diversos tipus de dades que s'utilitzen per representar diferents tipus d'informació. Els tipus de dades més comunes a JavaScript són:

1. **Números (Numbers)**: Representen valors numèrics, ja siguin sencers o decimals. Exemple: **let edat = 25;**

2. **Cadenes de Text (Strings)**: Emmagatzemen text. Exemple: **let nom = "María";**

3. **Booleans (Booleans)**: Representen valors de veritat (**true**) o falsedat (**false**). Exemple: **let activat = true;**

4. **Vectors (Arrays)**: Emmagatzemen col·leccions ordenades de dades. Exemple: **let colors = ["vermell", "verd", "blau"];**

5. **Objectes (Objects)**: Emmagatzemen col·leccions de parells clau-valor. Exemple:

```
let persona = {
 nom: "Juan",
 edat: 30,
 casat: false
};
```

Operadors Aritmètics i Lògics

Els operadors són eines fonamentals en programació per realitzar càlculs i comparacions. A JavaScript, tenim una varietat d'operadors a la nostra disposició:

- **Operadors Aritmètics**: S' utilitzen per realitzar càlculs matemàtics, com suma, resta, multiplicació, divisió, mòdul, increment i decrement. Exemple:

```
let suma = 5 + 3;
let resta = 10 - 4;
let = 6 * 7;
let = 20 / 4;
let modulo = 20 % 4; //Da com a resultat 0 que és la resta de dividir 20 entre 4
let = 20 / 4;
```

- **Operadors de Comparació**: Permeten comparar valors i retornen un valor booleà (**true** o **false**). Exemple:

```
let esMayor = 10 > 5;  true
let esIgual = 5 === "5";  false (comparació estricta)
```

- **Operadors Lògics**: S'utilitzen per combinar expressions booleanes i realitzar operacions lògiques com AND (**&&),** OR (**||**) i NOT(!). Exemple:

```
let condicion1 = true;
let condicion2 = false;

let resultatAND = condicion1 && condicion2; false
let resultadoOR = condicion1 || condicion2; true
```

En aquesta sessió, explorarem més exemples pràctics de com utilitzar aquestes variables, tipus de dades i operadors per realitzar càlculs i prendre decisions en programes JavaScript. Aquests conceptes són la base per construir algoritmes i aplicacions més complexes en sessions posteriors del curs.

&& (AND)		
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**False**

|| (OR)		
True	True	**True**
True	False	**True**
False	True	**True**
False	False	**False**

! (NOT)		
True	True	**True**
True	False	**True**

```
Exemple 1: Declaració de variables
let nom = "Juan";
let edat = 25;
Exemple 2: Tipus de dades
let missatge = "¡Hola, món!";
let saldo = 1000;
let activat = true;
Exemple 3: Operadors aritmètics
let suma = 5 + 3;  Resultat: 8
```

```
let resta = 10 - 4;  Resultat: 6
let = 6 * 7;  Resultat: 42
let = 20 / 4;  Resultat: 5
Exemple 4: Operadors de comparació
let esMayor = edat > 18;  Resultat: true
let esIgual = edat = = = 25;  Resultat: true
let esDiferent = saldo !== 0;  Resultat: true
Exemple 5: Operadors lògics
let condicion1 = true;
let condicion2 = false;
let resultatAND = condicion1 && condicion2;  Resultat: false
let resultadoOR = condicion1 || condicion2;  Resultat: true
Exemple 6: Concatenació de cadenes de text
let nomComplet = nom + " Pérez";  Resultat: "Juan Pérez"
Exemple 7: Modificació de variables
edat = 26;  Nova edat: 26
Exemple 8: Ús de cometes simples i dobles en cadenes
let missatge2 = 'Ella va dir: "Hola"';  Resultat: 'Ella va dir: "Hola"'
Exemple 9: Assignació de variables a altres variables
let x = 5;
let i = x;  i tindrà el valor de 5
Exemple 10: Operador d' increment
let comptador = 0;
comptador ++;  comptador ara és 1
Exemple 11: Operador de decrement
let comptador2 = 10;
comptador2--;  comptador2 ara és 9
Exemple 12: Concatenació de cadenes i números
let numero = 42;
let text = "El número és: " + nombre;  Resultat: "El número és: 42"
Exemple 13: Conversió de tipus de dades
let numeroString = "10";
let numeroEntero = parseInt(numeroString);  Resultat: 10
Exemple 14: Operacions mixtes
let resultadoMixto = 5 + "5";  Resultat: "55"
Exemple 15: Ús d' operadors d' assignació
let quantitat = 5;
quantitat + = 3;  quantitat ara és 8
Exemple 16: Comparació de valors
let comparacion = (edat > 30);  Resultat: false
Exemple 17: Operador ternari
let esMayorEdad = (edat >= 18) ? "Major d'edat" : "Menor d'edat";
Exemple 18: Concatenació de cadenes amb temperat literals (plantilles literals)
let producte = "Manzanas";
```

```
let quantitatProducte = 10;
let missatgeProducte = 'Tienes ${cantidadProducto} ${producto}.';   Resultat: "Tens 10
Illes."
Exemple 19: Ús de NaN (Not-a-Number)
let resultatNaN = 10 / "text";  Resultat: NaN
Exemple 20: Ús de typeof per verificar tipus de dades
let tipusEdad = typeof edat;  Resultat: "number"
```

8.3 ESTRUCTURES DE CONTROL.

En aquesta tercera sessió, explorarem les estructures de control a JavaScript, les declaracions condicionals i les sentències repetitives o bucles, que ens permeten prendre decisions en funció de condicions específiques o realitzar repeticions segons el cas. Al llarg d'aquesta sessió, aprendrem a utilitzar les declaracions **if**, **if anidado** y **else**, **while**, **do... while** i **for**.

Les declaracions condicionals són fonamentals en la programació, ja que ens permeten controlar el flux d' execució d' un programa en funció d' una condició. A JavaScript, utilitzem les següents estructures condicionals:

if: La declaració **if** permet executar un bloc de codi si una condició és veritable.

```
let edat = 18;

if (edat >= 18) {
 consoli.log("Ets major d'edat.");
}
```

If anidado: La declaració **else if** s'utilitza per avaluar una condició addicional si la primera condició **if** és falsa.

```
let hora = 15;

if (hora < 12) {
 console.log("Buenos días.");
} else if (hora < 18) {
 consoli.log("Bones tardes.");
} else {
 console.log("Bones nits.");
}
```

else: La declaració **else** s' executa si és false la condició de l' if.

```
let saldo = 0;

if (saldo > 0) {
 consoli.log("Tens saldo positiu.");
} else {
 consoli.log("No tens saldo.");
}
```

Exemples Pràctics

Ara, farem alguns exercicis pràctics per aplicar el que hem après sobre declaracions condicionals:

Exercici 1: Verificació de Contrasenya

```
let contrasenya = "secreta";
let contrasenyaUsuari = "secreta123";

if (contrasenyaUsuari === contrasenya) {
 console.log("Contrasenya correcta.");
} else {
 console.log("Contrasenya incorrecta.");
}
```

Exercici 2: Classificació de nombres

```
let numero = 15;

if (numero > 0) {
 console.log("El número és positiu.");
} else if (numero < 0) {
 consoli.log("El nombre és negatiu.");
} else {
 console.log("El número és igual a zero.");
}
```

Exercici 3: Determinació de Dia de la Setmana

```
let dia = "dimarts";

if (dia === "dilluns") {
 console.log("Avui és el primer dia laborable de la setmana.");
} else if (dia === "viernes") {
 console.log("Divendres! Cap de la setmana laborable.");
} else {
 consoli.log("Estem en meitat de la setmana laborable.");
}
```

Bucles a JavaScript

Els bucles són fonamentals en la programació, ja que ens permeten realitzar tasques repetitives de manera eficient. A JavaScript, utilitzem les següents declaracions de bucles:

for: La declaració **for** s' utilitza per executar un bloc de codi un nombre específic de vegades. Generalment, s'utilitza quan coneixem la quantitat exacta d'iteracions que desitgem.

```
for (let i = 0; i < 5; i++) {
 console.log("Iteració " + i);
}
```

while: La declaració **while** executa un bloc de codi mentre una condició sigui veritable. Es fa servir quan no sabem quantes vegades s'executarà el bucle per endavant.

```
let comptador = 0;

while (comptador < 3) {
 console.log("Iteració " + comptador);
comptador ++;
}
```

do... while: La declaració **do... while** és similar a **while**, però garanteix que el bloc de codi s'executarà almenys una vegada abans de verificar la condició.

```
let intents = 0;

do {
 console.log("Intento #" + intents);
intents++;
} while (intents < 3);
```

Exemples Pràctics amb Bucles

Aquí tens alguns exercicis pràctics per aplicar els conceptes de bucles a JavaScript:

Exercici 1: Imprimir Números Pares

```
for (let i = 0; i <= 10; i += 2) {
console.log(i);
}
```

Exercici 2: Sumar Números de l' 1 al 10

```
let suma = 0;

for (let i = 1; i <= 10; i++) {
suma += i;
}
console.log("La suma dels números de l'1 al 10 és: " + suma);
```

Exercici 3: Endevinar un número

```
const numeroAleatori = Math.floor(Math.random() * 10) + 1;
let intents = 0;
let adivinanza;

do {
 endevinalla = prompt("Adivina el número (entre 1 i 10):");
```

```
intents++;
    if (parseInt(adivinanza) === numeroAleatorio) {
 console.log("¡Correcte! El número era " + numeroAleatori);
        break;
 } else {
 console.log("Intento #" + intents + ": Intenta de nou.");
}
} while (intents < 3);

if (intents === 3) {
 console.log("Agotaste tus intentos. El número era " + numeroAleatori);
}
```

8.4 FUNCIONS I ÀMBIT DE VARIABLES EN JAVASCRIPT

En aquesta secció, explorarem el concepte de funcions a JavaScript i com afecten l'àmbit (scope) de les variables. Les funcions són blocs de codi reutilitzables que realitzen tasques específiques i són fonamentals en la programació JavaScript.

Funcions a JavaScript

Una funció és un bloc de codi que es pot anomenar i executar en qualsevol part d' un programa. Les funcions ajuden a organitzar i reutilitzar el codi de manera eficient. Per definir una funció a JavaScript, pots utilitzar la següent sintaxi:

```
function nombreDeLaFuncion(parametro1, parametro2) {
    Codi de la funció
    return resultat;
}
```

- **nomDeLaFuncion**: És el nom que tries per a la funció.
- **parametre1**, **parametre2**: Són els valors que la funció pot rebre com a entrada (arguments).
- **return resultat**: La funció pot retornar un resultat opcional.

```
function saludar(nombre) {
    return "Hola, " + nom + "!";
}
let missatge = saludar("Juan");
console.log(missatge);  Resultat: "Hola, Juan!"
```

Àmbit de Variables (Scope)

L' àmbit es refereix a la visibilitat i accessibilitat de les variables en un programa. A JavaScript, hi ha dos tipus principals d' àmbit de variables:

Àmbit Global: Les variables globals es declaren fora de qualsevol funció i són accessibles des de qualsevol part del programa.

```
let variableGlobal = "Sóc global";

function funcion1() {
 console.log(variableGlobal);  Puc accedir a la variable global
}
```

Àmbit Local o de Funció: Les variables locals es declaren dins d' una funció i només són accessibles des d' aquesta funció.

```
function funcion2() {
    let variableLocal = "Sóc local";
 console.log(variableLocal);  Puc accedir a la variable local
}
```

Funcions Anònimes i Expressions de Funcions

A més de declarar funcions amb un nom, també pots crear funcions anònimes i expressions de funcions. Aquestes s' utilitzen comunament com a arguments d' altres funcions o per definir funcions dins d' objectes.

Funció Anònima:

```
let sumar = function (a, b) {
    return a + b;
};

let resultat = sumar( 5, 3);  Cridant a la funció anònima
```

Expressió de Funció:

```
let multiplicar = function multiplica(a, b) {
    return a * b;
};

let producte = multiplicar( 4, 6);  Cridant a l' expressió de funció
```

En resum, les funcions són una part essencial de JavaScript que et permet modular i reutilitzar el teu codi de manera efectiva. Comprenent l'àmbit de les variables i com funcionen les funcions, estaràs més ben preparat per escriure programes JavaScript més estructurats i mantenibles.

8.5 . ARRAYS I OBJECTES EN JAVASCRIPT

En aquesta secció, explorarem els conceptes d'arrels i objectes a JavaScript. Aquestes són estructures de dades fonamentals que s' utilitzen per emmagatzemar i organitzar informació de manera eficient.

Arrays a JavaScript

Un array és una col·lecció ordenada d'elements. Cada element en un arrelament té un índex numèric que comença en 0. Els arrays poden contenir elements de qualsevol tipus, incloent números, cadenes, objectes o altres arrels. Per crear un array a JavaScript, pots utilitzar la següent sintaxi:

```
let miArray = [element1, element2, element3];
```

Exemple d' un Array:

```
let colors = ["vermell", "verd", "blau"];
```

Per accedir a elements individuals d'un arrelament, pots utilitzar el seu índex:

```
console.log(colors[0]);  Resultat: "vermell"
```

Objectes a JavaScript

Un objecte és una estructura de dades que emmagatzema parells clau-valor. Cada valor està associat a una clau única que actua com a identificador. Els objectes són útils per representar entitats del món real i les seves propietats. Per crear un objecte a JavaScript, pots utilitzar la següent sintaxi:

```
let miObjeto = {
clau1: valor1,
clau2: valor2,
clau3: valor3
};
```

Exemple d' un objecte.

```
let persona = {
 nom: "Juan",
 edat: 30,
 casat: false
};
```

Per accedir a les propietats d'un objecte, pots utilitzar la notació de punt o la notació de corxets:

```
console.log(persona.nom);  Resultat: "Juan"
console.log(persona["edat"]);  Resultat: 30
```

Arrays d' Objectes

Els arrays i objectes es poden combinar per crear estructures de dades més complexes. Per exemple, pots tenir un arrelament d'objectes que representin elements similars:

```
let estudiants = [
 { nom: "María", edat: 25 },
 { nom: "Carlos", edat: 22 },
 { nom: "Ana", edat: 30 }
```

```
];
```

Iteració a través d' Arrays i Objectes

Pots utilitzar bucles com **for** o **forEach** per iterar a través dels elements d'un arrelament o les propietats d'un objecte:

Iteració a través d' un Array:

```
for (let i = 0; i < colors.length; i++) {
console.log(colors[i]);
}
```

Iteració a través d' un Objecte:

```
for (let clave in persona) {
 console.log(clau + ": " + persona[clau]);
}
```

Els arrays i objectes són estructures de dades cabdals a JavaScript que et permeten emmagatzemar i organitzar informació de manera efectiva. Els arrays són ideals per a col·leccions d'elements similars, mentre que els objectes són útils per representar entitats i les seves propietats. Comprendre com treballar amb aquestes estructures de dades és essencial per al desenvolupament web a JavaScript.

8.6 . ESDEVENIMENTS I GESTIÓ D' ERRORS EN JAVASCRIPT.

En aquesta secció, abordarem els conceptes d'esdeveniments i la gestió d'errors en javascript. els esdeveniments són interaccions de l'usuari o esdeveniments del navegador que desencadenen accions en la teva aplicació web. La gestió d'errors és fonamental per identificar, manejar i solucionar problemes en el teu codi.

Esdeveniments a JavaScript

Els esdeveniments són accions que ocorren en una pàgina web, com fer clic en un botó, moure el mouse, escriure en un camp de text, etc. JavaScript permet la captura i manipulació d'aquests esdeveniments per crear aplicacions web interactives.

Per agregar un esdeveniment a un element HTML, primer has de seleccionar aquest element i després assignar-li una funció que s'executarà quan ocorri l'esdeveniment.

Exemple d' Esdeveniment Click:

```
Selecció de l'element amb id "miBoton"
let boton = document.getElementById("miBoton");

Agregar un esdeveniment de clic al botó
```

```
boton.addEventListener("click", function() {
 alerta("¡Hiciste clic al botó!");
});
```

Heus aquí una taula amb els principals esdeveniments en javascript.

Esdeveniment	Descripció
click	Es dispara quan es fa clic en un element.
mouseover	Passa quan el cursor del mouse entra en un element.
mouseout	Passa quan el cursor del mouse surt d'un element.
keydown	Es dispara quan una tecla del teclat es pressiona.
keyup	Es dispara quan se solta una tecla del teclat.
submit	Es dispara quan s'envia un formulari.
change	Passa quan el valor d'un element d'entrada canvia (com un camp de text o una casella de verificació).
load	Es dispara quan es completa la càrrega d' un recurs, com una imatge o un arxiu de script.
DOMContentLoaded	Passa quan el document HTML s'ha carregat completament i està llest per ser manipulat mitjançant JavaScript.
resize	Es dispara quan es canvia la mida de la finestra del navegador.
scroll	Passa quan l'usuari desplaça la pàgina web.
focus	Es dispara quan un element obté el focus (com un camp d'entrada).
blur	Es dispara quan un element perd el focus.

Gestió d' Errors en JavaScript

La gestió d'errors és crucial per identificar i manejar problemes que puguin sorgir en el teu codi. JavaScript proporciona blocs **try... catch** per capturar i manejar excepcions.

Exemple de Captura d' Errors:

```
try {
    Codi que pot generar un error
    let resultat = 10 / 0;
} catch (error) {
    Maneig de l' error
 console.log("Es va produir un error: " + error.message);
}
```

La captura d'errors permet que la teva aplicació continuï executant-se fins i tot si es produeix un error, en lloc d'aturar-se per complet.

Maneig d' Errades Personalitzades:

Pots crear les teves pròpies excepcions personalitzades utilitzant l'objecte **Error** i llançar-les en el teu codi quan sigui necessari:

```
function dividir(a, b) {
    if (b === 0) {
        throw new Error("No es pot dividir per zero.");
}
    return a / b;
}

try {
    let resultat = dividir( 10, 0);
} catch (error) {
 console.log("Error: " + error.message);
}
```

Esdeveniments i Gestió d' Errors en Pràctica:

Pots combinar esdeveniments i gestió d'errors per crear aplicacions web més robustes i amb millor capacitat de resposta. Per exemple, pots manejar errors en formularis web per proporcionar retroalimentació a l'usuari sobre entrades invàlides.

```
let formulari = document.getElementById("miFormulari");

formulari.addEventListener("submit", function(event) {
    try {
        Validació del formulari
        if (formulari.nom.value === "") {
            throw new Error("El camp de nom no pot estar buit.");
}
        Resta del procés de tramesa del formulari
 } catch (error) {
 event.preventDefault();  Evitar l' enviament del formulari
 alert("Error: " + error.message);
}
});
```

En resum, els esdeveniments i la gestió d'errors són aspectes essencials en el desenvolupament web amb JavaScript. Els esdeveniments permeten que la teva aplicació respongui a les accions de l'usuari, mentre que la gestió d'errors t'ajuda a identificar i manejar problemes en el teu codi de manera efectiva, millorant la confiabilitat i la usabilitat de les teves aplicacions web.

8.7. INTRODUCCIÓ AL DOCUMENT OBJECT MODEL (DOM) EN JAVASCRIPT

El Document Object Model, o DOM, és una representació programàtica de l' estructura d' un document HTML o XML. En altres paraules, el DOM permet als desenvolupadors accedir, manipular i modificar els elements i contingut d'una pàgina web utilitzant JavaScript.

Estructura del DOM

El DOM organitza l' estructura del document en una jerarquia de nodes. Els nodes representen parts del document, com a elements HTML, atributs i text. Els nodes s'organitzen en un arbre, amb un node arrel anomenat "document". Aquí hi ha una breu descripció d' alguns tipus de nodes comuns:

1. **Elements**: Representen etiquetes HTML, com ** **, **<p>**, **<h1>**, etc.
2. **Atributs**: Són els valors associats als atributs dels elements, com **id**, **class**, **src**, etc.
3. **Text**: Representa el contingut de text dins d' un element HTML.
4. **Nodes Pare i Fills**: Els elements poden tenir nodes fills (elements, atributs o text) i un node pare (l'element que conté els fills).

Manipulació del DOM a JavaScript

JavaScript et permet accedir i manipular el DOM d'una pàgina web. Pots seleccionar elements, canviar el seu contingut, modificar atributs i respondre a esdeveniments de l'usuari. Aquí en tens alguns exemples:

Seleccionar Elements:

```
 Seleccionar un element pel seu id→ getElementById
let miElemento = document.getElementById("miId");

 Seleccionar diversos elements per la seva classe→ getElementsByClassName
let elementos = document.getElementsByClassName("miClase");

 Seleccionar elements per la seva etiqueta→ getElementsByTagName
let parrafos = document.getElementsByTagName("p");
```

Canviar el Contingut d'un Element: (`innerHTML`)

```
let miParrafo = document.getElementById("parrafo");
miParrafo.innerHTML = "Nou contingut del paràgraf";
```

Modificar Atributs:

```
let miImagen = document.getElementById("imatge");
miImagen.src = "nova-imatge.jpg";
```

Respondre a Esdeveniments: `addEventListener`

```
let miBoton = document.getElementById("boton");
miBoton.addEventListener("click", function() {
 alerta("Hiciste clic al botó");
});
```

Crear Nous Elements: `appendChild`

```
let nouParrafo = document.createElement("p");
nouParrafo.innerHTML = "Aquest és un nou paràgraf";
document.body.appendChild(nouParrafo);
```

Eliminar Elements: `removeChild`

```
let elementoAEliminar = document.getElementById("elemento");
elementAEliminar.parentNode.removeChild(elementAEliminar);
```

Beneficis del DOM

- **Interactivitat**: Permet crear aplicacions web interactives que responguin a les accions de l' usuari.
- **Accessibilitat**: Facilita la manipulació i actualització de contingut web en temps real.
- **Dinamisme**: Permet modificar l' aparença i el contingut d' una pàgina sense necessitat de recarregar-la.

DOM és una part essencial del desenvolupament web a JavaScript. Proporciona una interfície per accedir i modificar elements HTML, la qual cosa permet crear experiències interactives i dinàmiques en aplicacions web. Comprendre com funciona el DOM és fonamental per a qualsevol desenvolupador web front-end.

Exemple 1: Canviar el Text d' un Paràgraf

Atès el següent codi HTML

```
<! DOCTYPE html>
<html>
<head>
    <title>Ejercicio 1</title>
</head>
<body>
    <p id="miParrafo">Aquest és un paràgraf d'exemple.</p>
    <button id="miBoton">Canviar Text</button>

    <script src="script1.js"></script>
</body>
```

```
</html>
```

scrip1.js

```
Seleccionar el paràgraf i el botó
let miParrafo = document.getElementById("miParrafo");
let miBoton = document.getElementById("miBoton");

Agregar un esdeveniment de clic al botó
miBoton.addEventListener("click", function() {
    Canviar el text del paràgraf
 miParrafo.innerHTML = "Text canviat!";
});
```

Quan facis clic al botó, el text del paràgraf es canviarà a "Text canviat!".

Exercici 2: Crear Elements Dinàmicament

En aquest exercici, crearàs un nou element de llista (li) i l'afegiràs a una llista (ul) quan es faci clic en un botó.

HTML:

```
<! DOCTYPE html>
<html>
<head>
    <title>Ejercicio 2</title>
</head>
<body>
    <ul id="miLista">
        <li>Element 1</li>
        <li>Element 2</li>
    </ul>
    <button id="miBoton">Agregar Elemento</button>

    <script src="script2.js"></script>
</body>
</html>
```

script2.js

```
Seleccionar la llista i el botó
let miLista = document.getElementById("miLista");
let miBoton = document.getElementById("miBoton");

Agregar un esdeveniment de clic al botó
miBoton.addEventListener("click", function() {
    Crear un nou element de llista
```

```
    let nouElement = document.createElement("li");
 nouElemento.innerHTML = "Nou Element";

    Agregar el nou element a la llista
miLista.appendChild(nouElement);
});
```

Exercici 3: Canviar el Color de Fons

En aquest exercici, canviaràs el color de fons de la pàgina web quan es faci clic en un botó.

HTML:

```
<! DOCTYPE html>
<html>
<head>
    <title>Ejercicio 3</title>
</head>
<body>
    <button id="miBoton">Canviar Color</button>

    <script src="script3.js"></script>
</body>
</html>
```

script3.js

```
Seleccionar el botó
let miBoton = document.getElementById("miBoton");

Agregar un esdeveniment de clic al botó
miBoton.addEventListener("click", function() {
    Canviar el color de fons de la pàgina
document.body.style.backgroundColor = getRandomColor();
});

Funció per obtenir un color aleatori
function getRandomColor() {
    letters = "0123456789ABCDEF";
    let color = "#";
    for (let i = 0; i < 6; i++) {
 color += letters[Math.floor(Math.random() * 16)];
}
    return color;
}
```

8.8. MANIPULACIÓ DEL DOM.

Modificació de Contingut i Atributs d' Elements

La modificació del contingut i els atributs d' elements és fonamental per a l' actualització dinàmica d' una pàgina web.

Exemple 1: Canvi de l' Atribut SRC d' una Imatge

```
Seleccionar una imatge pel seu id
let miImagen = document.getElementById("miImagen");

Canviar l' atribut src de la imatge
miImagen.src = "nova-imatge.jpg";
```

Exemple 2: Canvi de Text d' un Encapçalat.

Suposem que tens el següent encapçalat h1 a la teva pàgina HTML:

```
<h1 id="miTitulo">Títol Original</h1>
```

Pots utilitzar JavaScript per canviar el text de l'encapçalat quan es carrega la pàgina:

```
Seleccionar l' encarregat h1 pel seu id
let miTitulo = document.getElementById("miTitulo");

Canviar el text de l' encarregat
miTitulo.textContent = "Nou Títol";
```

Exemple 3: Canvi d' Estil d' un Paràgraf

Suposem que tens el següent paràgraf a la teva pàgina HTML:

```
<p id="miParrafo">Aquest és un paràgraf d'exemple.</p>
```

Pots utilitzar JavaScript per canviar el contingut del paràgraf de la següent manera:

```
Seleccionar un paràgraf pel seu id
let miParrafo = document.getElementById("miParrafo");

Modificar el text del paràgraf
miParrafo.textContent = "Text modificat del paràgraf";
```

Ara, el contingut del paràgraf s'ha canviat a "Text modificat del paràgraf".

Exemple4: Canvi d'Enllaç d'una Imatge

Suposem que tens una imatge a la teva pàgina HTML amb un enllaç original:

```
<img id="miImagen" src="imatge1.jpg" alt="Imatge Original">
```

Pots utilitzar JavaScript per canviar l'enllaç (src) de la imatge quan es fa clic en un botó:

```
Seleccionar la imatge pel seu id
let miImagen = document.getElementById("miImagen");

Seleccionar el botó pel seu id
let miBoton = document.getElementById("miBoton");

Agregar un esdeveniment de clic al botó
miBoton.addEventListener("click", function() {
    Canviar l'enllaç (src) de la imatge
 miImagen.src = "imatge2.jpg";
    Canviar l'atribut "alt" de la imatge
 miImagen.alt = "Nova Imatge";
});
```

En aquest exemple, quan facis clic al botó, l'enllaç de la imatge es canviarà a "imatge2.jpg" i l'atribut "alt" es canviarà a "Nova Imatge". Això pot ser útil per canviar dinàmicament el contingut d'una imatge en resposta a esdeveniments de l'usuari, com un botó de "Canviar Imatge".

Creació i Eliminació d' Elements en el DOM

JavaScript permet crear nous elements HTML i agregar-los al DOM, així com eliminar elements existents.

Em remeto als exemples del punt anterior.

Gestió d' Esdeveniments en Elements en el DOM

Exemple 1: Esdeveniment Click en un Botó

Suposem que tens un botó a la teva pàgina HTML i desitges mostrar un missatge quan l'usuari hi faci clic.

HTML:

```
<button id="miBoton">Haz clic</button>
```

JavaScript

```
Seleccionar el botó pel seu id
let miBoton = document.getElementById("miBoton");

Agregar un esdeveniment de clic al botó
miBoton.addEventListener("click", function() {
 alerta("Hiciste clic al botó");
});
```

Quan l'usuari faci clic al botó, apareixerà un quadre d'alerta amb el missatge "Fes clic al botó".

Exemple 2: Esdeveniment Mouseover en una Imatge

Suposem que tens una imatge i desitges canviar la seva mida quan l'usuari col·loca el mouse sobre ella.

HTML:

```
<img id="miImagen" src="imatge.jpg" alt="Imatge">
```

JavaScript:

```
Seleccionar la imatge pel seu id
let miImagen = document.getElementById("miImagen");

Agregar un esdeveniment de mouseover a la imatge
miImagen.addEventListener("mouseover", function() {
    Canviar l' ample de la imatge al doble
 miImagen.style.width = "400px";
});
```

Quan l'usuari col·loqui el mouse sobre la imatge, aquesta s'ampliarà al doble de la seva mida original.

Exemple 3: Esdeveniment Keydown en un Camp de Text

Suposem que tens un camp de text i desitges mostrar un missatge quan l'usuari pressioni una tecla al camp.

HTML:

```
<input id="miInput" type="text" placeholder="Escriu alguna cosa">
```

JavaScript:

```
Seleccionar el camp de text pel seu id
let miInput = document.getElementById("miInput");

Agregar un esdeveniment de keydown al camp de text
miInput.addEventListener("keydown", function(event) {
 alert("Pressioneu la tecla: " + event.key);
});
```

Quan l'usuari escrigui alguna cosa en el camp de text i pressioni una tecla, apareixerà un quadre d'alerta que mostrarà la tecla pressionada.