JSON i JavaScript

JSON i JS

- **Exemple 1: Crear un objecte JSON simple i accedir a les seues propietats**

```
// Pas 1: Definir un objecte JSON
var persona = {
  nom: "Juan",
  edat: 30,
  ciutat: "Nova York"
};

// Pas 2: Accedir a les propietats de l'objecte JSON
console.log(persona.nom); // Juan
console.log(persona.edat);   // 30
console.log(persona.ciutat); // Nova York
```

→ Creem un objecte JSON senzill anomenat **persona** amb tres propietats: **nom**, **edat** i **ciutat**.
→ Utilitzem la notació de punts (**persona.nom**) per accedir a les propietats de l'objecte i mostrar els seus valors utilitzant **console.log()**.

- **Exemple 2: Convertir un objecte JavaScript en JSON i viceversa**

```
// Pas 1: Definir un objecte JavaScript
var persona = {
  nom: "Juan",
  edat: 30,
  ciutat: "Nova York"
};

// Pas 2: Convertir l'objecte JavaScript a JSON
var jsonString = JSON.stringify(persona);
console.log(jsonString); // {"nom":"Juan","edat":30,"ciutat":"Nova York"}

// Pas 3: Convertir el JSON de tornada a un objecte JavaScript
var objecte = JSON.parse(jsonString);
console.log(objecte); // { nom: "Juan", edat: 30, ciutat: "Nova York" }
```

→ **JSON.stringify()** per convertir l'objecte JavaScript **persona** a una cadena JSON.
→ **JSON.parse()** per convertir la cadena JSON **jsonString** en un objecte JavaScript **objecte**.
→ Això ens permet convertir fàcilment entre formats d'objectes JavaScript i JSON.

- **Exemple 3: Iterar sobre un array d'objectes JSON**

```
// Pas 1: Definir un array d'objectes JSON
var persones = [
  { nom: "Juan", edat: 30 },
  { nom: "María", edat: 25 },
  { nom: "Pedro", edat: 35 }
];

// Pas 2: Iterar sobre l'array i accedir a les propietats de cada objecte
persones.forEach(function(persona) {
  console.log(persona.nom + " té " + persona.edat + " anys.");
});
```

→ Bucle **forEach** per recórrer cada objecte de l'array **persones**.
→ Dins del bucle, accedim a les propietats de cada objecte (**persona.nom** i **persona.edat**) i les mostrem utilitzant **console.log()**.

- **Exemple 4: Filtrar dades d'un array d'objectes JSON**

```
// Pas 1: Definir un array d'objectes JSON
var persones = [
  { nom: "Juan", edat: 30 },
  { nom: "María", edat: 25 },
  { nom: "Pedro", edat: 35 }
];

// Pas 2: Filtrar persones majors de 30 anys
var personesMajors = persones.filter(function(persona) {
  return persona.edat > 30;
});

console.log(personesMajors); // [{ nom: "Juan", edat: 30 }, { nom: "Pedro", edat: 35 }]
```

→ Utilitzem el mètode **filter** per filtrar els elements de l'array **persones** segons una condició. A l'exemple, volem les persones amb una edat superior a 30 anys.
→ El resultat es guarda a la variable **personesMajors** i es mostra per consola.

- **Exemple 5: Modificar dades dins d'un objecte JSON existent**

```
// Pas 1: Definir un objecte JSON
var persona = {
  nom: "Juan",
  edat: 30,
  ciutat: "Nova York"
};

// Pas 2: Modificar la propietat "edat"
persona.edat = 31;

console.log(persona); // { nom: "Juan", edat: 31, ciutat: "Nova York" }
```

→ Accedim directament a la propietat **edat** de l'objecte **persona** i li assignem un nou valor.
→ Mostrem l'objecte actualitzat per consola.

- **Exemple 6: Afegir i eliminar propietats d'un objecte JSON**

```
// Pas 1: Definir un objecte JSON
var persona = {
  nom: "Juan",
  edat: 30,
  ciutat: "Nova York"
};

// Pas 2: Afegir una nova propietat a l'objecte
persona.professio = "Enginyer";

// Pas 3: Eliminar una propietat de l'objecte
delete persona.ciutat;

console.log(persona); // { nom: "Juan", edat: 30, professio: "Enginyer" }
```

→ Afegim una nova propietat (**professio**) a l'objecte **persona** i després eliminem la propietat **ciutat** utilitzant l'operador **delete**.

- **Exemple 7: Encadenar objectes JSON**

```
// Pas 1: Definir un objecte JSON amb objectes encadenats
var persona = {
  nom: "Juan",
  edat: 30,
  direccio: {
    carrer: "123 Carrer Principal",
    ciutat: "Nova York",
    codi_postal: "10001"
  }
};

// Pas 2: Accedir a propietats d'objectes encadenats
console.log(persona.direccio.carrer); // 123 Carrer Principal
```

→ Definim un objecte JSON **persona** amb una propietat anomenada **direccio**, que és a la vegada un objecte JSON amb les propietats **carrer**, **ciutat** i **codi_postal**.
→ Podem accedir a les propietats d'aquest objecte encadenat utilitzant la notació de punts (**persona.direccio.carrer**).

- **Exemple 8: Validar i manipular dades JSON provinents d'una API**

```
// Pas 1: Simular dades JSON d'una API
var dadesJSON = '{"nom": "Juan", "edat": 30, "professio": null}';

// Pas 2: Validar i manipular les dades
try {
  var dades = JSON.parse(dadesJSON);

  // Verificar si la propietat "professio" està definida
  if (dades.professio !== null) {
    console.log("Professió:", dades.professio);
  } else {
    console.log("La professió no està definida.");
  }
} catch (error) {
  console.error("Error en analitzar les dades JSON:", error);
}
```

→ Simulem dades JSON d'una API i les analitzem utilitzant **JSON.parse()**.
→ Verifiquem si la propietat **professio** està definida i la mostrem per consola.
→ Si hi ha algun error en l'anàlisi de les dades JSON, capturem l'error amb **try...catch**.

- **Exemple 9: Ordenar un array d'objectes JSON**

```
// Pas 1: Definir un array d'objectes JSON
var persones = [
  { nom: "Juan", edat: 30 },
  { nom: "María", edat: 25 },
  { nom: "Pedro", edat: 35 }
];

// Pas 2: Ordenar l'array per edat de forma ascendent
persones.sort(function(a, b) {
  return a.edat - b.edat;
});

console.log(persones); // [{ nom: "María", edat: 25 }, { nom: "Juan", edat: 30 }, { nom:
"Pedro", edat: 35 }]
```

→ Mètode **sort** per ordenar l'array **persones** segons la propietat **edat**.
→ Comparem les edats utilitzant una funció de comparació i retornem un valor negatiu, zero o positiu segons si la primera edat és més xicoteta, igual o més gran que la segona.

- **Exemple 10: Combinar múltiples objectes JSON en un sol objecte**

```
// Pas 1: Definir diversos objectes JSON
var dades1 = { a: 1, b: 2 };
var dades2 = { c: 3, d: 4 };
var dades3 = { e: 5, f: 6 };

// Pas 2: Combinar els objectes en un sol objecte
var dadesCombinades = Object.assign({}, dades1, dades2, dades3);

console.log(dadesCombinades); // { a: 1, b: 2, c: 3, d: 4, e: 5, f: 6 }
```

→ **Object.assign()** per combinar múltiples objectes (**dades1**, **dades2**, **dades3**) en un sol objecte **dadesCombinades**.
→ El primer paràmetre buit **{}** especifica l'objecte de destinació on es combinaran les propietats dels altres objectes.