ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

HONORIS UNITED UNIVERSITIES

Développement Web:
JavaScript

Professeur:

Nouhaila MOUSSAMMI

n.moussammi@emsi.ma

2024/2025

Manipulation des Objets

Manipulation des Objects

JS JavaScript

Les enregistrements (object)

- **Créer à un objet**

Un Objet (Enregistrement) est un conteneur de paires **"clé: valeur"**, où chaque clé est une chaîne de caractères et la valeur peut être de tout type, y compris une fonction (appelée méthode). Les objets sont créés avec des accolades **{...}** et permettent de regrouper plusieurs valeurs sous une seule entité, simplifiant ainsi leur gestion.

```
let admin = {
   // La clé nom stocke la valeur 'Ilyass'
   nom: "Ilyass",
   age: 45,
   profession: "Développeur"
 };
```

Dans cet exemple, l'objet admin a les propriétés (clés) nom, age, profession, avec leurs valeurs assignées à l'aide de deux points (:). Les propriétés sont séparées entre elles par des virgules.

Manipulation des Objects

JS JavaScript

Les enregistrements (object)

- **Créer à un objet**

Une autre méthode pour créer un objet est d'utiliser le constructeur **Object**.

```
let admin = new Object();
admin.nom = "Ilyass";
admin.age = 45;
admin.profession = 'Développeur';
```

Manipulation des Objects

JS JavaScript

Les enregistrements (object)

- **Accéder à un objet**

```
console.log(admin.nom);
console.log(admin.profession);
```

```
'Ilyass'
'Développeur'
```

- **Modifier des propriétés**

```
admin.age = 50; // 50
```

- **Ajouter une propriété**

```
admin.mail = 'Ilyass1990@gmail.com';
console.log(admin.mail);
```

```
'Ilyass1990@gmail.com'
```

Manipulation des Objects

JS JavaScript

Les enregistrements (object)

- **Supprimer une propriété**

```
delete admin.age; // L'age a supprimer
```

- **Parcourir un objet par for..in**

```
for (let propriete in admin) {
 console.log(propriete + ": " + admin[propriete]);
}
```

```
'nom: Ilyass'
'age: 50'
'profession: Développeur'
'mail: Ilyass1990@gmail.com'
```

L'utilisation de **for... in** permet de de parcourir les propriétés énumérables d'un objet.

Manipulation des Objects

JS JavaScript

Les enregistrements (object)

- **Parcourir un objet par Object.keys()**

```
console.log(Object.keys(admin));
```

```
[ 'nom', 'age', 'profession', 'mail' ]
```

Object.keys() renvoie un tableau contenant les noms des propriétés (clés ou keys) propres à un objet et qui sont énumérables.

- **Object.values()**

Object.values() renvoie un tableau contenant les **valeurs** des propriétés énumérables de cet objet.

Manipuler les objets

JS JavaScript

Exemple:

```
const utilisateur = {
  nom: "Alice",
  age: 30,
};

// Ajouter
utilisateur.email = "alice@example.com";

// Modifier
utilisateur.age = 31;

// Supprimer
delete utilisateur.nom;

console.log(utilisateur);
```

Accéder aux propriétés

```
console.log(utilisateur["age"]); // Accès par indexation
console.log(utilisateur.email); // Accès par point
```

Parcourir un objet

```
for (let key in utilisateur) {
  console.log(`${key}: ${utilisateur[key]}`);
}
```

Obtenir les clés ou valeurs

```
console.log(Object.keys(utilisateur)); // ['age', 'email']
console.log(Object.values(utilisateur)); // [31, 'alice@example.com']
```

Manipulation des Objects

Les enregistrements (object)

- **Vérifier l'existence d'une propriété**

```
// L'utilisation de in
if ('nom' in admin) {
 console.log('La clé "nom" existe');
}
```

```
'La clé "nom" existe'
```

Manipulation des Objects

Les enregistrements (object)

- **Méthodes d'objets**

```
let admin = {
 nom: "Ilyass",
 role: "Super Admin",
 email: "Ilyass1990@gmail.com",
 isActive: true,
 estDisponible: function() {
   console.log(`L'administrateur est maintenant
${this.isActive ? "actif" : "inactif"}.`);
 }
};

// Appeler la fonction toggleActiveStatus
admin.estDisponible();
```

```
"L'administrateur est maintenant actif."
```

Manipulation des Objects

Les enregistrements (object)

- **Hiérarchie d'objets (objets imbriqués)**

Les objets imbriqués en JavaScript sont des objets qui contiennent d'autres objets à l'intérieur d'eux. Cela permet de structurer des données complexes de manière hiérarchique et de mieux organiser les informations.

```
let admin = {
   nom: "Ilyass",
   role: "Super Admin",
   email: "Ilyass1990@gmail.com",
   profile: {                          // Objet imbriqué pour les détails du profil
     phone: "+212 661686745",
     address: {                        // Objet imbriqué pour l'adresse
       street: "XXX",
       city: "Casablanca",
       country: "Maroc"
     }
   },
```

Manipulation des Objects

JS JavaScript

Les enregistrements (object)

- **Créer à un objet**

La **fonction constructeur** est une autre méthode pour créer des objets en JavaScript. Elle permet de définir un modèle d'objet avec des propriétés et des méthodes associées.

```
function NomObjet(param1, param2, ...) {
 // Initialisation des propriétés
 this.propriete1 = param1;
 this.propriete2 = param2;
};
```

Manipulation des Objects

JS JavaScript

Les enregistrements (object)

- **Création de la fonction constructeur de l'objet voiture**

```
function voiture(marque, modele, annee,couleur,moteur) {
  this.marque = marque;
  this.modele = modele;
  this.annee = annee;
  this.couleur = couleur;
  this.moteur = {
    type: moteur.type,
    puissance: moteur.puissance,
 }
  this.description = function() {
    return `${this.marque} ${this.modele} (${this.annee}), Couleur: ${this.couleur}, du type:
${this.moteur.type}`;
    };
};
```

Manipulation des Objects

JS JavaScript

Les enregistrements (object)

- **Création de deux instances de voiture**

Dans cet exemple, nous avons créé deux voitures, Tesla et Porsche, en utilisant la fonction constructeur voiture et en passant les valeurs appropriées.

```
var Tesla = new voiture("Tesla", "Model 3", 2023, "Noir",{type:"Électrique",puissance:283});
var Porsche = new voiture("Porsche", "Carrera", 2023, "Noir",{type:"Essence", puissance:379});
```

- **Accès aux propriétés et méthodes**

```
console.log(Tesla.modele);
console.log(Porsche.description());
```

```
'Model 3'
'Porsche Carrera (2023), Couleur: Noir, du type: Essence'
```

Manipuler les objets

JS JavaScript

Manipulation des objets natifs

Les tableaux : déclaration

Un tableau JS est un objet qui hérite de l'objet global standard **Array** (héritant de **Object**).

L'objet **Array** est une liste d'éléments indexés dans lesquels on pourra ranger (écrire) et reprendre (lire) des données.

- **Déclaration d'un tableau vide :**

```
let tableau = new Array; // déclaration explicite en instanciant l'objet Array
let tableau = new Array(3); // 3 correspond à la taille du tableau

let tableau = [];        // déclaration de manière littérale avec []
```

- **Déclaration et initialisation :**

```
let tableau = new Array(5, 10, 15, 20);
let tableau = ['A', 'B', 'C'];
```

- **Taille du tableau :** La propriété **length** de l'objet **Array** retourne la taille du tableau :

```
let tableau = ['A', 'B', 'C'];
let nbElements = tableau.length;
console.log(nbElements); // 3
```

Manipuler les objets

JS JavaScript

Manipulation des objets natifs

Accéder aux éléments du tableau

Pour accéder à un élément du tableau, on utilise son indice : **nom_du_tableau[i] = "elément";**

Rappel : Les indices du tableau comment à 0

- **Accéder à un élément du tableau :**

```
let tableau = ['A', 'B', 'C'];
console.log(tableau[1]); // Retourne 'B'
```

- **Récupérer l'indice d'un élément : la méthode indexOf() :**

```
let tableau = ['A', 'B', 'C'];
console.log(tableau.indexOf('C')); // Retourne 2
```

- **indexOf()** a un deuxième paramètre optionnel qui permet aussi de choisir l'indice à partir duquel la recherche débute (Par défaut, ce deuxième paramètre est à 0) :

```
let tableau = ['A', 'B', 'C', 'B'];
console.log(tableau.indexOf('B'));    // Retourne 1, l'indice du premier B trouvé
console.log(tableau.indexOf('B', 2)); // Retourne 3, l'indice du premier B trouvé après l'indice 2
```

Manipuler les objets

JS JavaScript

Manipulation des objets natifs

Parcourir un tableau

On peut parcourir un tableau de différente manières :

```
for (let i = 0; i < monTableau.length; i++)
{
// monTableau[i] permet d'accéder à l'élément courant du tableau
}
//Ou bien
for (const monElement of monTableau)
{
        // monElement permet d'accéder à l'élément courant du tableau
}
```

```
monTableau.forEach ( monElement => {
        // monElement permet d'accéder à l'élément courant du tableau
});

//Ou bien
monTableau.forEach ( function ( monElement)
{
        // monElement permet d'accéder à l'élément courant du tableau
});
```

Exemple :

```
let tableau = ['A', 'B'];
tableau.forEach(function(element) {
  console.log(element);
});
// Retourne :
// 'A';
// 'B';
```

Manipuler les objets

JS JavaScript

Manipulation des objets natifs

Ajouter un élément dans un tableau

- La méthode **push** ajoute un élément à la fin du tableau :

```
let tableau = ['A', 'B'];
tableau.push('C');
console.log(tableau);   // Retourne ['A', 'B', 'C']
```

- La méthode **unshift** ajoute un élément au début du tableau :

```
let tableau = ['A', 'B'];
tableau.unshift(22);
console.log(tableau); // Retourne [22, 'A', 'B'];
```

Modifier un élément du tableau

Pour modifier la valeur d'un élément, on peut directement utiliser son indice :

```
let tableau = ['B', 'B', 'C'];
tableau[0] = 'A';   // Modifie la première case
tableau[4] = 'E';   // ajoute un élément dans l'indice 4
console.log(tableau);  // Retourne ['A', 'B', 'C', empty, 'E']
```

Manipuler les objets

JS JavaScript

Manipulation des objets natifs

Supprimer un élément du tableau

- La méthode **pop()** supprime le dernier élément du tableau :

```
let tableau = ['A', 'B', 'C'];
tableau.pop();
console.log(tableau);  // Retourne ['A', 'B'];
```

- La méthode **shift()** supprime le premier élément du tableau :

```
let tableau = ['A', 'B', 'C'];
tableau.shift();
console.log(tableau); // Retourne ['B', 'C'];
```

- La méthode **splice**() permet de supprimer plusieurs éléments :

Le premier paramètre est l'indice à partir duquel supprimer, et le second est le nombre d'éléments à supprimer.

```
let tableau = ['A', 'B', 'C'];
tableau.splice(1,1);
console.log(tableau); // Retourne ['A', 'C'];
```

Manipuler les objets

JS JavaScript

Manipulation des objets natifs

Trier un tableau

- La méthode **sort()** retourne les éléments par ordre alphabétique (elle doivent être de la même nature) :

```
let tableau = ['E', 'A', 'D'];
tableau.sort();
console.log(tableau); // Retourne ['A', 'D', 'E']
```

- La méthode **reverse()** Inverse l'ordre des éléments (sans tri) :

```
let tableau = ['A', 'B', 'C'];
tableau.reverse();
console.log(tableau); // Retourne ['C', 'B', 'A'];
```

Recherche un élément dans le tableau

- La méthode **findIndex**() retourne l'indice du premier élément du tableau qui remplit une condition :

```
function findC(element) {
    return element === 'C';
  }
  let tableau = ['A', 'B', 'C', 'D', 'C'];
  tableau.findIndex(findC);  // Retourne 2, l'indice du premier C
```

Manipulation des Objects

Strings

- **Definition**

Les **chaînes de caractères** (ou **strings**) en JavaScript sont des séquences de caractères utilisées pour stocker des données textuelles, telles que des textes, des noms ou des adresses. Elles peuvent être délimitées par des guillemets simples ('), doubles (") ou des backticks (`). Les chaînes de caractères font partie des types de données primitifs de JavaScript et sont couramment utilisées dans de nombreux contextes.

```
const string1 = "Hello world";
const string2 = 'It\'s me, your prof';
const string3 = `How are you doing?`;
```

Manipulation des Objects

JS JavaScript

Strings

- **La longueur d'une chaîne de caractères: length**

```
console.log(string1.length); // 11
```

- **Accéder à un caractère: charAt()**

```
console.log(string1.charAt(4)); // 'o'
```

- **index de la première occurrence: indexOf()**

```
console.log(string2.indexOf('your')); // 9
```

- **Extraire une portion de la chaîne: substring(index start,index end)**

```
console.log(string3.substring(0, 9)); // 'How are y'
```

Manipulation des Objects

JS JavaScript

Strings

- **Extraire une portion de la chaîne en acceptant des indices négatifs: slice(index start,index end)**

```
console.log(string3.slice(-10)); // 'ou doing ?'
```

- **Convertir une chaîne en majuscules ou en minuscules: toUpperCase() et toLowerCase()**

```
console.log(string1.toUpperCase()); // 'HELLO WORLD'
console.log(string3.toLowerCase()); // 'how are you doing ?'
```

- **Supprimer les espaces au début et à la fin d'une chaîne: trim()**

```
const string4 = " This message for you ";

console.log(string4.trim());
```

```
'This message for you'
```

Manipulation des Objects

JS JavaScript

Strings

- **Remplacer une portion de chaîne par une autre: replace()**

```
let nouveauString = string1.replace('world', 'students');
console.log(nouveauString); // 'Hello students'
```

- **Divise une chaîne en un tableau de sous-chaînes: split()**

```
console.log(nouveauString.split(' '));
```

```
[ 'Hello', 'students' ]
```

- **Vérifier si une sous-chaîne est présente dans la chaîne: includes()**

```
console.log(nouveauString.includes('javascript')); // false
console.log(nouveauString.includes('Hello')); // true
```

Manipulation des Objects

JS JavaScript

Number

- **Definition**

En JavaScript, un nombre (ou **number**) est l'un des types de données primitifs. JavaScript utilise un seul type pour représenter tous les nombres, qu'ils soient **entiers** ou **décimaux**.

```
var nb = 5; typeof(nb);
var nb = 255.0; typeof(nb);
var nb = -15; typeof(nb);

// Notation scientifique (1 million)
var nb = 1e6; typeof(nb);
```

```
'number'
'number'
'number'
'number'
```

Manipulation des Objects

Number

- **Déterminer si la valeur passée en argument vaut NaN (NotANumber): isNaN()**

```
let num1 = 2000;
let num2 = "2000";
let num3 = "Hello 2000";

console.log(isNaN(num1)); // false
console.log(isNaN(num2)); // false
console.log(isNaN(num3)); // true
```

- **Déterminer si la valeur passée en argument est un entier: isInteger()**

```
let num4 = 2000.5;
console.log(isInteger(num1)); // true
console.log(isInteger(num4)); // false
```

Manipulation des Objects

JS JavaScript

Number

- **Déterminer si la valeur numérique passée en argument est un nombre fini: isFinite()**

```
console.log(isFinite(Infinity)); // false
console.log(isFinite(0)); // true
console.log(isFinite(1 / 0)); // false
```

- **Convertir une chaîne en un entier et un nombre à virgule flottante: parseInt() et parseFloat()**

```
console.log(parseFloat(0.5)); // 0.5
console.log(parseInt(0.5)); // 0
console.log(parseFloat("409")); // 409
console.log(parseInt("5.7")); // 5
```

Manipulation des Objects

JS JavaScript

Number

- **Retourner une chaîne représentant le nombre avec la notation en virgule fixe: toFixed()**

```
let num5 = 56.7854;
console.log(num5.toFixed("3")); // '56.785'
```

- **Convertir une chaîne en un entier et un nombre à virgule flottante: parseInt() et parseFloat()**

```
console.log(num5.toString()); //'56.7854'
```

Manipulation des Objects

Date

- **Definition**

En JavaScript, les dates sont gérées par l'objet **Date**, qui permet de créer, modifier et afficher des dates et des heures. Bien qu'il n'existe pas de type primitif pour les dates, l'objet **Date** offre diverses méthodes pour manipuler ces données sans avoir de propriétés directes.

```
let aujourdHui = new Date();
console.log(aujourdHui);
```

```
2024-11-29T18:50:05.436Z
```

Manipulation des Objects

JS JavaScript

Date

- **Obtenir le jour, le mois, l'année, etc.**

```
let aujourdHui = new Date();

// getDate(): extraire le jour du mois
console.log(aujourdHui.getDate());

// getDay(): extraire le jour de la semaine (0 = dimanche, 1 = lundi, ..., 6 = samedi)
console.log(aujourdHui.getDay());

// getMonth(): extraire le mois
console.log(aujourdHui.getMonth());

// getFullYear(): extraire l'année complète (par exemple, 2024)
console.log(aujourdHui.getFullYear());
```

Manipulation des Objects

JS JavaScript

Date

- **Obtenir le jour, le mois, l'année, etc.**

```
// getTime(): obtenir le nombre de millisecondes depuis le 1er janvier 1970
console.log(aujourdHui.getTime());

// getHours(): extraire l'heure (de 0 à 23)
console.log(aujourdHui.getHours());
```

Manipulation des Objects

JS JavaScript

Date

- **Définir les propriétés de la date**

Il est également possible de modifier certains aspects de l'objet Date après sa création, à l'aide des méthodes commençant par **set**.

```
let date = new Date();

date.setFullYear(2025);
date.setMinutes(30);
console.log(date);
```

```
2025-11-29T19:30:27.802Z
```

Manipulation des Objects

JS JavaScript

Math

- **Definition**

L'objet **Math** en JavaScript est un objet natif contenant des méthodes et des constantes pour effectuer des opérations mathématiques. Il est directement accessible et ne nécessite pas d'instanciation.

- **Calculer la racine carrée: sqrt()**

```
console.log(Math.sqrt(16));   // 4
```

- **Calculer la puissance d'un nombre: pow()**

```
console.log(Math.pow(base, exponent));
```

Manipulation des Objects

JS JavaScript

Math

- **Arrondir un nombre à l'entier inférieur le plus proche: floor()**

```
console.log(Math.floor(4.9)); // 4
```

- **Arrondir un nombre à l'entier le plus proche: round()**

```
console.log(Math.round(4.6)); //5
```

- **Retourne la valeur absolue d'un nombre: abs()**

```
console.log(Math.abs(-12)); // 12
```

- **retourne le plus grand et le plus petit nombre parmi les arguments passés: max() et min()**

```
console.log(Math.max(34, 765, 75));   // 765
console.log(Math.min(56, 23, 124));   // 23
```

Manipulation des Objects

Window

- **Definition**

En JavaScript, l'objet **Window** représente l'onglet (ou la fenêtre) du navigateur dans lequel une page web est chargée, de manière globale. Cela signifie que **window** est implicitement accessible partout dans le code JavaScript, sans nécessiter de déclaration explicite. En utilisant les méthodes disponibles sur l'objet window, il est possible de:

- **Récupérer la taille de la fenêtre: Window.innerWidth et Window.innerHeight**

```
console.log(window.innerWidth);    // Largeur de la fenêtre
console.log(window.innerHeight);   // Hauteur de la fenêtre
```

- **Fournir des informations sur l'URL de la page actuel: window.location.href**

```
console.log(window.location.href);
```

Manipulation des Objects

Window

- **Affiche une boîte de dialogue d'alerte avec un message: alert()**

```
window.alert("Hello world !");
```

- **Affiche une boîte de dialogue avec un message et deux boutons ("OK" et "Annuler"): confirm()**

```
window.confirm("Es-tu sûr que ton âge est supérieur à 18 ?");
```

- **Affiche une boîte de dialogue qui permet de saisir une valeur : prompt()**

```
window.prompt("Et donc quel est ton age ?");
```

Manipulation des Objects

Window

- **Ouvre une nouvelle fenêtre ou un nouvel onglet avec l'URL spécifiée: open()**

```
window.open("https://developer.mozilla.org", "_blank", "width=600,height=400");
```

- **Ferme la fenêtre actuelle: close()**

```
window.close();   // Ferme la fenêtre actuelle
```

Manipuler les objets

JS JavaScript

Manipulation JSON

JSON : définition et caractéristiques

JSON (JavaScript Object Notation) est un format d'échange de données qui est facile à utiliser par les humains et les machines. Ce format est utilisé pour échanger les valeurs entre les applications (clients) et les serveurs. Sa forme complète est en notation d'objet JavaScript.

Caractéristiques de JSON :

- **Facile à utiliser** - l'API JSON offre une mise en œuvre avancée des différents types et structures de données, ce qui aide à simplifier les cas d'utilisation.
- **Performance et rapidité** - JSON consomme très peu de mémoire.
- **Outil gratuit** - la bibliothèque JSON est open source et gratuite.
- **Dépendance** - la bibliothèque JSON ne nécessite pas l'utilisation d'une autre bibliothèque pour le traitement.
- **Compatibilité :**
  - JSON est supporté par les navigateurs ;
  - JSON est pris en charge par tous les principaux framework JavaScript ;
  - JSON permet de transmettre et de sérialiser des données structurées à l'aide d'une connexion réseau ;
  - JSON est compatible avec des langages de programmation modernes.

Manipuler les objets

JS JavaScript

Manipulation JSON

JSON : Syntaxe

Règles générales de la syntaxe JSON

- Les données Json sont écrites entre accolades (braces) ;
- Les données sont représentées sous forme de paires de clé – valeur ;
- Les clés doivent être mises entre guillemets (double quotes) ;
- La clé et la valeur doivent être séparées par deux points (:) ;
- La virgule (,) est utilisée pour séparer les données ;
- Les crochets tiennent les tableaux (brackets) ;
- Les accolades retiennent les objets.

Les types JSON : Number, String (entre guillemets), Boolean, Null, Array, Object

Exemple :

```
{       "nom"     :       "saidi",
        "prenom"  :       "ali",
        "age"             :40,
        "interets"        :null,
        "experience"      : ["CSS","JS","HTML"],
        "addresse"        : {"Rue"  : "Sidi Maarouf","Ville": "Casablanca","codeP" : 10000   }
}
```

Manipuler les objets

JS JavaScript

Manipulation JSON

Manipulation des données JSON

Pour traiter et afficher les données JSON dans les pages web, on a souvent besoin de les convertir en objets Javascript et vice versa.

- **Analyse syntaxique (parse)** : Convertir une chaîne de caractères en un objet natif.
- **Linéarisation (stringification)** : Convertir un objet natif en chaîne de caractères.

En Javascript, les méthodes utilisées sont :

- **JSON.parse** permet de convertir JSON vers un objet javascript.
- **JSON.stringify** permet de convertir des objets javascript vers des données JSON.

Exemples :

```
//Création d'un string JSON
var jsonData = '{"nom":"Saidi", "prenom":"Ali"}';
document.write(typeof(jsonData)+'<br>'); //string

//Convertir JSON vers Javascript
var jsObject = JSON.parse(jsonData);
document.write(typeof(jsObject)+'<br>');//object
document.write(jsObject+'<br>'); //[object object]

document.write(jsObject.nom + "
 "+jsObject.prenom+'<br>'); //Saidi Ali
```

```
//Création d'un objet Javascript
var jsObject = {nom:"Saidi", prenom:"ali"};
document.write(typeof(jsObject)+'<br>');//Object

//Convertir javascript vers JSON
var jsonString = JSON.stringify(jsObject);
document.write(typeof(jsonString)+'<br>');//string

document.write(jsonString);//{"nom":"Saidi","prenom
":"ali"}
//remarquer la présence des "" dans les clés
```

Manipuler les objets

JS JavaScript

Manipulation JSON

A. Conversion d’un objet en JSON

```
const data = {
  nom: "Alice",
  age: 30,
  langues: ["Français", "Anglais"],
};

const jsonData = JSON.stringify(data);
console.log(jsonData); // {"nom":"Alice","age":30,"langues":["Français","Anglais"]}
```

Manipuler les objets

JS JavaScript

Manipulation JSON

B. Conversion d’une chaîne JSON en objet

```
const jsonString = '{"nom":"Bob","age":25,"langues":["Espagnol","Italien"]}';

const objet = JSON.parse(jsonString);
console.log(objet.nom); // Bob
```