Fiche Javascript

Date de création	@16 janvier 2024 10:01
Cours	Techno Web Dynamic Front
Type	Fiche de révision
Relue ?	☐
Année	2023.24 L1 SDN
Semestre	L1 S2

Cours :

- navigateur : logiciel côté client qui permet d'afficher des pages web

  l'ordinateur doit être connecté à internet

  l'utilisateur saisit une adresse url et via le protocole http le navigateur se connecte au serveur web
- moteurs de rendu (affiche à l'écran) : gecko, blink, webkit
- javascript :
  - langage de script

    - côté navigateur
    - interpréteurs : v8, spider monkey par exemple
    - usages : client web, applications mobiles, rendu côté serveur, 3d, machine learning, iot
- ECMAScript : ensemble de normes, c'est un standard et un langage de programmation orienté prototype
- comité TC39 : réalisent 5 étapes de validation pour mettre à jour le standard ou non, si l'on veut utiliser le standard à venir avant sa validation on peut utiliser BabelJS
- Javascript est fortement typé
- l'arité d'une fonction est le nombre d'argument qu'elle requiert

Types de valeurs et leur déclaration (il y a également les valeurs indéfinies, les valeurs nulles et les symboles) :

```
//nombre
let a = 5;
let c = 0;
//chaîne de caractères
let b="2";
let txt = "un texte";
let prenom ="";
let nvchaîne = "";
//booléen
let t = true:
//tableau
let g=[1,2,3];
let item=g[0];
//objet
let Personne1 = { name: "John", age: 28 };
//bibliothèque maths
let i = 0;
i = Math.random(); //prend un nombre entre 0 et 1
```

```
//méthodes du tableau g
c = g.length;
g.reverse();
g.join(""); //concatène tous les éléments de g
prenom = Personne1.name //dans une fonction on utilisera this
g.sort() //va trier le tableau g

//méthode de string
txt.includes("Hello"); //test pour voir si txt comporte la ch
str = str.toUpperCase(); //chaque lettre est convertie en upp
str = str.toLowerCase();
nvchaîne = txt.replace("word", "yes") //on cherche un motif d

//connaître le type d'une variable
typea = typeof a;

//saut à la ligne dans une chaîne de caractères :
let saut = "\n";

//ajout de données en fonction du type
a += 0;
txt += "bla bla";
t = !t;
g.push(1); //ajout dans un tableau

//changer le type de b (cela fonctionne car b est une chaîne
c=Number(b);

//affichage avec un document
document.write("des mots");
```

Commandes console :

```
//affichage normal
console.log(a);
//affichages avec des dessins et des couleurs
console.info('information')
```

```
console.error('an error')
console.warn('a warning')
//affichage de tableau
let technologies = {
"Standards": ["HTML", "CSS", "SVG", "JS"],
"Others": ["jQuery", "Markdown", "Pug"]
};
console.table(technologies);
//personnalisation de la console
const text = 'more text';
console.log('%c Oh my heavens! %s', 'background:
#222; color: #bada55', text);
//afficher une variable
affiche() {
    console.log(
      `Une belle ${this.marque} de couleur ${this.couleur} av
        this.portes
      } portes et ${this.kilometrage} kilomètres${
        this.abs ? " avec" : " sans"
      } abs`
    );
}
```

Opérateurs :

Opérateurs de comparaison :

```
//égalité et inégalité
(1 == '1') // true
(1 === '1') // false

//comparaison
1 < 1 //false
1 <= 1 //true
'1' <= 1 //true

//test le début de txt avec le pattern regex
```

```
let regex = /^[aeiouAEIOU]/;
return regex.test(txt);
```

Opérateurs arithmétiques :

```
//binaires : multiplication *, division / et modulo %
const a = 4 ;
const b = 2 ;

const c = a * b; // 8
const d = 5 * '5'; // 25, conversion automatique du string en

const e = a / b; // 2
const f = a % 2 ; // 0

//unaire : ++ et --
let a = 1 ;
a++ ; // 2
a-- ; // 1
```

Opérateurs logiques :

```
//non logique
let a = true ;
a = !a; //du coup a vaut false

//le "ou" logique
console.log('test' || 0) //test

//le "et" logique
console.log('test' && 0) // 0
```

Structure de contrôle :

```
//if...else
if (condition) {
```

```
// Si vrai exécution de ce code
} else if (condition2) {
// Autre si exécution de ce code
} else {
// Sinon
}

//switch...case, Attention au break !!
switch (a) {
    case 0:
        // exécution du code
        break;
    case 1:
        // exécution du code
        break;
    default:
        // exécution du code
}

//conditions ternaires
(a !== b) ? console.log('a !== b') : console.log('a === b');

//méthode repeat(), crée une nouvelle chaîne de caractères en
const nouvelleChaine = chaine.repeat(nombreDeRepetitions);

//méthode filter(), crée un nouveau tableau contenant les élé
const playExo4V2 = (nb = 21) =>
  document.write(
    `${[...Array(nb).keys()]
      .reverse()
      .filter((item) => item % 2 === 0)
      .join(" - ")}`
  );

//méthode floor(), prend le résultat de la division et arrond
let myNumber = 8457855;
let count = 0;
```

```
const playExo9 = () => {
  while (myNumber !== 0) {
    myNumber = Math.floor(myNumber / 10);
    count++;
  }
  console.log("count", count);
};

//méthode prompt(), affiche une boîte de dialogue modale avec
let userInput = prompt("Veuillez saisir votre nom :");
console.log("Vous avez saisi : " + userInput);
```

Boucles :

```
//for
for (let i = 0; i < 10; i += 1) {
    console.log(i);
} // 0 1 2 3 4 5 6 7 8 9

//while
let i = 0;
while (i < 10) {
    console.log(i++);
} // 0 1 2 3 4 5 6 7 8 9

//do...while
let i = 1;
do {
    console.log(i++);
} while (i < 5);

//for...in
const array = ['test1', 'test2'];
for(index in array) {
    console.log(index);
}
```

```
//for..of
const myobject = { a: 1, b: 2, c: 3 };
for (const key of Object.keys(myobject)) { //attention à bien
    console.log(myobject[key]);
}

//break
let str = 0;
for (let i = 0; i < 5; i++) {
    if (i === 3) {
        break;
        }
        str += i;
}
console.log(str); // 3

//continue
let str = 0;
for (let i = 0; i < 5; i++) {
    if (i === 3) {
        continue;
    }
    str += i;
}
console.log(str); // 7
```

Fonctions :

```
//déclaration d'une fonction
function inverse (n) {
    // traitement
    return -n;
}

//fonction fléchée
const inverse = n => -n;
```

```
//callbacks, fonction passée en tant qu'argument d'un autre e
const isPair = function(a, fct, fct2) {
    if(a % 2 ===0) {
        return fct(a);
    }
    return fct2(a);
};

const logPair = function(a) {
    return`${a} est pair`;
};
const logImpair = function(a) {
    return`${a} est impair`;
};

//paramètre par défaut
function getValue(value, step = 1) {
return value + step;
}
console.log(getValue(5)); // 6

//fonction d'ordre supérieur, prennent une ou plusieurs fonct
const addOld = (a,b) => a + b;
const add = function(a) {
    return function(b) {
        return a + b;
    }
}
add(1)(2); // 3

//closures, fonctions qui se rappellent de l'environnement da
const add = function(a) {
    return function(b) {
        return a + b;
    }
}
const add2 = add(2);
const add4 = add(4);
```

```
let counter = 5;
counter = add2(counter);
counter = add4(counter);
console.log(counter) // 11

//module : fichier contenant du code Javascript, généralement
<script type="module" src="./index.js" />

//export
const obj = {1,2,3};
export default obj; //un seul export par défaut dans un modul
export const dec = a => a - 1; //export individuel
const i = 1;
const j = 2;
function add(i){
    return i+1;
}
export {add,j}; //liste d'exports

//import
/*ici, obj est le seul export par défaut du module, i et j do
être importées entre crochets et sont appelées "exports nommé
on peut renommer l'export avec "as"*/
import obj, { add, j as 10emelettredelalphabet} from
"./monObjet";
/*sinon on peut réaliser un import global*/
import * as t from "./monObjet";
//pour utiliser les éléments après :
const objetplusun = t.add(t.default);
```

Objets :

```
//déclaration
const myCar = {
    make: "Ford",
    model: "Mustang",
    year: 1969,
```

```
};

//get et set : permettent de lier une propriété à une fonctio
let objet = {
  _x: 0, // La propriété avec un nom commencant par _ est sou
  get x() {
    return this._x;
  },
  set x(value) {
    this._x = value;
  }
};

console.log(objet.x); // Output: 0
objet.x = 10;
console.log(objet.x); // Output: 10

//delete : supprimer une propriété d'un objet
delete address.postCode;

//DefineProperty : définir des propriétés d'un objet de manièr
Object.defineProperty(obj, "clé", {
    enumerable: false, //indique si la propriété pourra être
    configurable: false, // la valeur de la proriete ,e peut
    writable: false, // certaines caractéristiques ne peuvent
    value: "static", //valeur d'initialisation
    get() { },
    set() { }
});

//Spread operator : opérateur de decomposition, permet d'effe
//copie :
const street = "10 rue de Lille";
const city = "Paris";
const address = { street, city };
const copyAddress = { ...address };
copyAddress.city = "Tokyo";
//concatenation :
```

```
const obj1 = {
    propA: "hello",
    propB: "world"
};
const obj2 = {
    propB: "France",
    propC: "!"
};
const mergedObj = { ...obj1, ...obj2 };
console.log("mergedObj", mergedObj);

//Destructuring : permet de simplifier la façon de recuperer
const address = {
    street: "10 rue de Lille",
    city: "Paris",
    country: "france",
    phone: "0101010101"
};

//Methodes
Object.assign //Permet de copier un objet ou de concaténer 2
Object.seal //Scelle un objet pour l'ajout de propriété mais
Object.freeze //Protège en modification le premier niveau de
Object.keys //Renvoie un tableau avec les propriétés de l'obj
Object.values //Renvoie un tableau avec les valeurs de l'obje
Object.entries //Renvoie un tableau avec des paires clé/valeu

//Optional chaining : simplifie l'acces aux propriete d'objet
let personneSansAdresse = {
  nom: "Bob",
  age: 25
};
console.log(personneSansAdresse.adresse?.rue); // Output: und
```

Tableaux :

```
//Déclaration d'un tableau
const panier = ['🍅', '🍄', '🍏', '🍓', '🌽', '🍒', '🍇'];

//Recuperation d'un element par son index
panier[0]; // '🍅'
panier[2]; // '🍏'
panier[5]; // '🍒'

//Longueur
panier.length // 7

//Spread Operator
const myArray = [1,2,3];
console.log('array : ', myArray) // [1,2,3]
console.log('array spread', ...myArray) // 1 2 3
//concatenation
console.log("first", first);
console.log("second", second);
const combined2 = [...first, ...second];
console.log("combine2", combined2);
//copie
const newSecond = [...second];
newSecond[0] = 0;
console.log("newSecond", newSecond);

//Destructuring
const myFunction =(...args) => console.log(args);

//Methodes
Array.concat //concaténation de tableaux
const legumes = [('🍅', '🍄', '🌽'];
const fruits = ['🍏', '🍓', '🍒', '🍇'];
const panier = legumes.concat(fruits);
Array.join //regroupe les éléments d'un tableau en chaine. Le
legumes.join('#'); // '🍅#🍄#🌽'
Array.map //Renvoie une chaine de caractères, si rien n'est p
const legumes = [('🍅', '🍄', '🌽'];
const newLegumes = legumes.map((item, index, array) => `icone
```

```
Array.filter //permet de créer un nouveau tableau filtré. Il
legumes.filter((item, index, array) => (item === '🍅'));
Array.reduce //créer une nouvelle valeur à l'aide d'un accumu
// 0 + 1 + 2 + 3 + 4
const initialValue = 0;
const sumWithInitial = array1.reduce(
    (accumulator, currentValue) => accumulator + currentValue
  initialValue
);
console.log(sumWithInitial); // 10
//Ajouter un élément à la fin
panier.push('🍌')
//Ajouter un élément au début
panier.unshift('🍌');
//Supprimer le dernier élément
panier.pop();
//Vérification si tableau
Array.isArray(panier)
//Vérifier la présence d'une valeur
panier.includes('🍅'); // true
//Supprimer le premier élément
panier.shift();

//Mutation : attention aux méthodes qu'on utilise
const merged = array.concat(arr1, arr2);
const filtered = array.filter(fn);
const mapped = array.map(fn);
//méthodes qui retournent une nouvelle dans muter l'original
```

DOM et API Selector

DOM (Document Object Model)
→API
→ permet d'accéder aux éléments du document HTML et de les modifier
→ permet de modifier les propriétés CSS ou du contenu texte
→ arborescence de noeuds
→ racine : doctype et html

→ accès : propriete window.document (ou simplement window)
→ element html reference grace a la methode documentElement

```
//Navigation
console.log(document.documentElement.constructor.name);
console.log(document.documentElement.childNodes); // Acces au
console.log(document.documentElement.children); // Liste des
console.log(document.documentElement.childNodes[1].nodeName);
console.log(document.body.firstChild); // Acces au dernier él
console.log(document.body.lastChild); // Acces au dernier élé
// Accès au parent : parentNode
```

Selection d'elements :

```
// getElementById : accéder à un élément par son attribut id
let title = document.getElementById('page-title');
<h1 id="page-title">Title</h1>

// getElementsByTagNames : récupérer la liste des éléments pa
let divs = document.getElementsByTagNames('div');

// getElementsByClassName : récupérer tous les éléments d'une
let title = getElementsByClassName('page-title test'); // tou
<h1 id="page-title">Title</h1>

// querySelector : retourne le 1er élément trouvé
let headerPage =
document.querySelector('header.page');
let title = document.querySelector('#title');

// querySelectorAll : retourne tous les éléments correspondan
let buttons = document.querySelectorAll('.btn');

// textContent : modifier un nœud de type texte
let message = document.querySelector('span.message');
message.textContent = "Hello";
```

```
//  innerHTML : donne le contenu intérieur
message.innerHTML = '<p>Hello</p>';
```

Gestion des attributs HTML :

```
// récupération de l'element html
let link = document.querySelector('a.link');
// vérifie l'existence de l'attribut href
console.log(link.hasAttribute('href')); // true
// récupère la valeur de l'attribut href
console.log(link.getAttribute('href')); // /index.html
// modifie la valeur de l'attribut title
link.setAttribute('title', 'Link to Home Page');
// supprime l'attribut target
link.removeAttribute('target');

// récupérer la liste de tous les attributs
link.attributes
```

Edition des classes :

```
//  méthodes particulières accessibles avec element.classList
// ajouter une classe :
element.classList.add('maClass')
// supprimer une class :
element.classList.remove('maClass')
// toggle (ajouter si elle est pas dedans, supprimer si elle
element.classList.toggle('maClass')
// vérifier la présence d'une classe :
element.classList.contains('maClass')
// remplacer une classe :
element.classList.replace('oldClass', 'newClass')
```

Edition des styles :

```
// modifier le style d’un élément HTML en utilisant le setAtt
const link =
document.querySelector("a.link:not(.link-active)");
link.setAttribute("style", "font-size: 0.9rem;
font-weight: bold;");
link.style.color = "grey";
link.style.backgroundColor = "#eee";

// Récupération de la valeur de l’attribut style en chaine de
console.log("cssText", link.style.cssText);
```

Style calculé :

```
//  lorsque l’on définit du style uniquement avec du CSS exte
console.log("inline border radius",
link.style.borderRadius); // vide
const borderRadius = window
.getComputedStyle(link)
.getPropertyValue("border-radius");
console.log("computed border radius", borderRadius);
```

Noeuds :

```
// AJOUTS DE NOEUDS
const body = document.querySelector('body');
body.innerHTML = `${body.innerHTML} <p>Hello ICL</p>`;

const body = document.querySelector('body');
const titleIcl2 = document.createElement('h1');
titleIcl.textContent = "ICL";
body.appendChild(titleIcl)

// SUPPRESSION DE NOEUDS
// supprimer un noeud enfant
node.removeChild(child);
```

```
// supprimer le nom specifié
elmt.parentNode.removeChild(elmt);

//supprimer tous les noeuds enfants d'un élément
var element = document.getElementById("top");
while (element.firstChild) {
element.removeChild(element.firstChild);
}
```

Dataset :

```
//  attributs universels en HTML : data-*
<input id="id" name="note" value="15" data-defaultvalue="10">

const input = document.getElementById('id');
console.log(input.dataset.defaultvalue)
```

Evenements :

```
// AddEventListener : rajouter une ecoute d'evenement
const submitBtn = document.querySelector('button[type="submit
submitBtn.addEventListener('click', function (event) { //On a
    console.log(event);
});
```

Objet Event

Le type de l'objet Event va varier en fonction du type d'évènement que l'on va chercher à écouter. Dans l'exemple précédent, on a
ajouté un écouter de type 'click', l'event est de type MouseEvent (sur Firefox) ou PointerEvent (sur Chrome ou Edge)
Ce qui va nous intéresser pour le moment, c'est de connaitre l'élément qui a été cliqué.
Pour cela, on peut utiliser la propriété event.target ou currentTarget :

- Le target qui capte en premier l'évènement
- Le currentTarget est l'élément sur lequel l'évènement est attaché.

preventDefault : permet de stopper le comportement natif d'une interaction

Propagation des évènements
Les évènements se propagent dans l'arbre DOM en 2 phases :
• de la racine vers l'élément cible (phase de capture)
• puis de l'élément cible vers la racine (bubbling)

3 ème élément

Un 3ème argument est un objet d'options qui permet de spécifier les caractéristiques de l'écouteur d'évènements.

Once
Booléen indiquant que l'écouteur doit être invoqué qu'une seule fois

Capture
Booléen permet d'inverser le sens de propagation

Passive
Booléen indiquant que l'écouteur n'appellera jamais le preventDefault

Type et Détachement :

Il existe une multitude de types d'évènements pour la souris, le clavier, les formulaires, le viewport et la page.
• submit : soumission de formulaire
• change : changement de la valeur d'un champ (en sortant du champ)
• input : changement de la valeur d'un champ (à la volée)
• scroll : scroll dans la page ou sur un élément
• keydown : touche pressée (event de type keyboard event donne accès à la touche du clavier)
• keyup : touche relâchée
• keypress : touche saisie (se situe entre keydown et keyup)
• focus : élément ayant le focus
• blur : sortie du focus

```
document.getElementById('myDIV').addEventListener('mousemove'
document.getElementById('myDIV').removeEventListener('mousemo
```

QCM

1. Quelle méthode JavaScript est utilisée pour supprimer une propriété d'un objet ?

   A) `removeAttribute`

   B) `delete`

   C) `removeProperty`

   D) `erase`

2. Que fait la méthode `Object.defineProperty` dans JavaScript ?

   A) Elle supprime une propriété d'un objet.

   B) Elle définit des propriétés d'un objet de manière plus précise.

   C) Elle ajoute une nouvelle propriété à un objet.

   D) Elle fusionne deux objets.

3. Quelle est la fonction de l'optional chaining (`?.`) en JavaScript ?

   A) Elle permet de concaténer deux chaînes de caractères.

   B) Elle permet d'accéder à une propriété d'objet de manière "optionnelle" lorsque certaines propriétés peuvent être `null` ou `undefined`.

   C) Elle permet de filtrer les éléments d'un tableau.

   D) Elle permet de définir une nouvelle propriété d'objet.

4. Quelle méthode JavaScript est utilisée pour ajouter un élément à la fin d'un tableau ?

   A) `array.unshift()`

   B) `array.push()`

   C) `array.pop()`

   D) `array.shift()`

5. Quelle méthode JavaScript est utilisée pour regrouper les éléments d'un tableau en une chaîne de caractères ?

   A) `Array.join`

   B) `Array.reduce`

   C) `Array.concat`

   D) `Array.filter`

6. Comment accéder à un élément HTML par son attribut ID en JavaScript ?

   A) `document.getElementByName`

   B) `document.getElementById`

   C) `document.getElementByClassName`

   D) `document.querySelector`

7. Quelle méthode JavaScript est utilisée pour ajouter une classe à un élément HTML ?

   A) `element.classList.toggle`

   B) `element.classList.add`

   C) `element.classList.remove`

   D) `element.classList.replace`

8. Comment supprimer un nœud enfant d'un élément HTML en JavaScript ?

   A) `element.removeChild(child)`

   B) `element.parentNode.removeChild(element)`

   C) `element.remove()`

   D) `element.clearChild()`

9. Quel événement JavaScript est déclenché lorsqu'une touche du clavier est pressée ?

A) `keydown`

B) `keypress`

C) `keyup`

D) `keydownup`

10. Comment détacher un gestionnaire d'événement en JavaScript ?

    A) `removeEventListener`

    B) `addEventListener`

    C) `detachEvent`

    D) `unbind`

11. Quelle méthode JavaScript est utilisée pour créer un nouvel élément HTML ?

    A) `createElement`

    B) `createNode`

    C) `newElement`

    D) `addNode`

12. Quelle méthode est utilisée pour insérer un nouvel élément avant un élément existant en JavaScript ?

    A) `insertBefore`

    B) `insertAfter`

    C) `addBefore`

    D) `addAfter`

13. Comment modifier le texte contenu dans un élément HTML en JavaScript ?

    A) `element.text`

    B) `element.textContent`

    C) `element.innerText`

    D) `element.innerHTML`

14. Quelle méthode est utilisée pour vérifier si une classe est présente sur un élément HTML en JavaScript ?

    - A) `element.hasClass`

- B) `element.classList.has`
- C) `element.containsClass`
- D) `element.classCheck`

15. Comment fusionner deux objets en JavaScript ?
    - A) En utilisant `Object.assign`
    - B) En utilisant `Object.merge`
    - C) En utilisant `Object.combine`
    - D) En utilisant `Object.concat`
16. Quelle méthode JavaScript est utilisée pour vérifier si une valeur est un tableau ?
    - A) `isArray()`
    - B) `isTable()`
    - C) `isList()`
    - D) `isArr()`
17. Quelle méthode est utilisée pour filtrer les éléments d'un tableau en fonction d'un critère donné en JavaScript ?
    - A) `filter()`
    - B) `search()`
    - C) `find()`
    - D) `match()`
18. Comment récupérer la longueur d'un tableau en JavaScript ?
    - A) `array.length`
    - B) `array.size`
    - C) `array.count`
    - D) `array.length()`
19. Quelle méthode est utilisée pour ajouter plusieurs éléments à la fois à un tableau en JavaScript ?
    - A) `pushMany()`

- B) `concat()`
- C) `add()`
- D) `append()`

20. Comment accéder à la valeur d'un attribut HTML en JavaScript ?
    - A) `getAttribute()`
    - B) `getElementAttribute()`
    - C) `getAttr()`
    - D) `attributeValue()`
21. Quelle méthode JavaScript est utilisée pour supprimer un attribut HTML d'un élément ?
    - A) `removeAttribute()`
    - B) `deleteAttribute()`
    - C) `removeAttr()`
    - D) `clearAttribute()`
22. Quelle méthode JavaScript est utilisée pour vérifier si un élément a un attribut spécifique ?
    - A) `hasAttribute()`
    - B) `containsAttribute()`
    - C) `checkAttribute()`
    - D) `attributeExists()`
23. Comment ajouter du style à un élément HTML en JavaScript ?
    - A) En utilisant `element.style`
    - B) En utilisant `element.addStyle()`
    - C) En utilisant `element.css()`
    - D) En utilisant `element.applyStyle()`
24. Quelle méthode JavaScript est utilisée pour obtenir les propriétés CSS calculées d'un élément ?
    - A) `getComputedStyle()`

- B) `getCSS()`
- C) `calculateStyle()`
- D) `computeStyle()`

25. Quel événement JavaScript est déclenché lorsqu'un formulaire est soumis ?
    - A) `submit`
    - B) `formSubmit`
    - C) `formSubmitClick`
    - D) `formSubmitEvent`
26. Comment empêcher le comportement par défaut d'un événement en JavaScript ?
    - A) `stopPropagation()`
    - B) `preventDefault()`
    - C) `cancelEvent()`
    - D) `disableEvent()`
27. Quel événement JavaScript est déclenché lorsqu'un élément reçoit le focus ?
    - A) `blur`
    - B) `focusin`
    - C) `focus`
    - D) `focusout`
28. Comment créer un nouvel élément HTML avec du texte en JavaScript ?
    - A) `createElement()`
    - B) `createTextNode()`
    - C) `createHTMLNode()`
    - D) `createTextElement()`
29. Quelle méthode JavaScript est utilisée pour insérer un nouvel élément après un élément existant ?
    - A) `insertAfter()`

- B) `insertBefore()`
- C) `addAfter()`
- D) `addBefore()`

30. Comment vérifier si une propriété existe dans un objet JavaScript ?
    - A) `propertyExists()`
    - B) `hasProperty()`
    - C) `propertyIn()`
    - D) `inProperty()`

C'est une série de 30 questions. Je peux continuer si vous le souhaitez.

Réponses :

1. B) `delete`
2. B) Elle définit des propriétés d'un objet de manière plus précise.
3. B) Elle permet d'accéder à une propriété d'objet de manière "optionnelle" lorsque certaines propriétés peuvent être `null` ou `undefined`.
4. B) `array.push()`
5. A) `Array.join`
6. B) `document.getElementById`
7. B) `element.classList.add`
8. A) `element.removeChild(child)`
9. A) `keydown`
10. A) `removeEventListener`
11. A) `createElement`
12. A) `insertBefore`
13. C) `element.innerText`
14. B) `element.classList.has`
15. A) En utilisant `Object.assign`
16. A) `isArray()`

17. A) `filter()`
18. A) `array.length`
19. B) `concat()`
20. A) `getAttribute()`
21. A) `removeAttribute()`
22. A) `hasAttribute()`
23. A) En utilisant `element.style`
24. A) `getComputedStyle()`
25. A) `submit`
26. B) `preventDefault()`
27. C) `focus`
28. B) `createTextNode()`
29. A) `insertAfter()`
30. B) `hasProperty()`