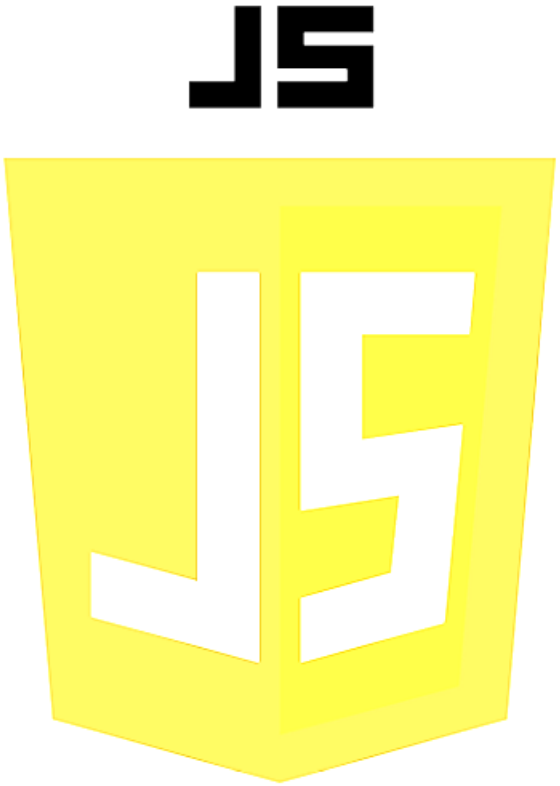
Introduction au `JavaScript`



NSI Première

Académie de Versailles

M. Boehm, C. Poulmaire, P. Remy

Sommaire

Le principe général du langage `JavaScript` est de créer des sites web interactifs, c'est-à-dire des sites web qui puissent afficher du contenu en fonction d'événements réalisés côté client. C'est un langage de programmation libre créé en 1995 par Brendan Eich de la famille des langages interprétés orientés objets. C'est également un langage compatible avec le DOM des pages web.

1 Insertion d'un code `JavaScript`

Il existe deux possibilités pour insérer un code `JavaScript` dans un document `HTML` :

- En l'écrivant entre les balises `<script>` et `</script>` du conteneur `<head>`. On utilise alors la syntaxe suivante :

  `<script>`*`code JavaScript`*`</script>`

- En utilisant un fichier externe d'extension `.js`. On utilise alors la syntaxe suivante :

  `<script src=`*`nom_fichier`*`></script>`

  Le paramètre `nom_fichier` est obligatoire et est une chaîne de caractères correspondant au chemin relatif depuis le fichier `HTML` source.

2 Instructions, variables, commentaires et sorties

- **Une instruction** se termine toujours par un point-virgule `;`
- **Le nom d'une variable** commence toujours par le mot-clé `let`
- **Affecter une valeur à une variable `x` :** `let x= ...`
- **Afficher le contenu d'une variable `x` dans une boîte de dialogue :** `alert(x)`
- **Demander à l'utilisateur une valeur :** `prompt("`*`TexteAffiche`*`")` (la valeur renvoyée est une chaîne de caractères)

3 Opérations arithmétiques

addition	soustraction	multiplication	division	puissance $n^{ème}$ de x
`+`	`-`	`*`	`/`	`x**n`

Remarque 3.1 Attention, les opérations doivent toujours être écrites. Par exemple, on écrit `2*x` et non pas `2x`.

4 Structures de données

4.1 Les types

- `typeof x` : renvoie le type de la variable `x`. On a les correspondances suivantes :

nombre	chaîne de caractères	booléen	objet	non définie
`number`	`string`	`boolean`	`object`	`undefined`

Exemple 4.1 Les instructions

```
let x = 25;
alert(typeof x);
let y;
alert(typeof y);
alert(typeof 1.25);
alert(typeof "texte");
alert(typeof true);
alert(typeof [1,2,3]);
```

affichent respectivement `number`, `undefined` (variable définie mais non initialisée), `number`, `string`, `boolean` et `object` (on a en fait un tableau).

Remarque 4.2 Comme en `Python`, il existe des fonctions de transtypage. Par exemple, `Number('3')` renvoie l'entier `3`.

4.2 Les structures booléennes

Les données de type `boolean` sont des données qui ne peuvent prendre que deux valeurs : `true` ou `false`.

- ⋆ Ce type de données apparaît lorsque l'on teste si une condition est réalisée. Par exemple, l'instruction `2 < 8` renvoie la valeur `true`, alors que l'instruction `5 < 1` renvoie la valeur `false`.
- ⋆ Il peut être également intéressant d'affecter directement des données booléennes à des variables, notamment pour définir des instructions bloquantes à l'aide d'une boucle `while`.

4.3 Les chaînes de caractères

- ⋆ Une chaîne de caractères est définie entre des guillemets doubles `"..."` ou des guillements simples `'...'`.
- ⋆ Une chaîne de caractères n'est pas modifiable.
- ⋆ Le caractère `\"` permet d'intégrer un guillement double dans une chaîne de caractères.
- ⋆ Le caractère \ permet également d'écrire des chaînes de caractères sur plusieurs lignes.
- ⋆ La concaténation de deux chaînes de caractères se fait avec l'opérateur +.

Exemple 4.3 Ecriture d'une chaîne sur plusieurs lignes avec intégration de guillemets doubles :

```
let texte = "Bonjour. Le \
             texte ci-dessous est \
             écrit en \"JavaScript\".";
```

- ⋆ Quelques fonctions utiles sur les chaînes de caractères :
  - *chn*`[i]` : renvoie le caractère de la chaîne `chn` situé au rang `i+1` (le premier caractère est donc indexé par 0).
  - *chn*`.length` : renvoie le nombre de caractères de la chaîne `chn`.
  - *chn*`.toUpperCase()` : renvoie une copie en majuscule de la chaîne `chn`, cette dernière n'étant pas modifiée.
  - *chn*`.toLowerCase()` : renvoie une copie en minuscule de la chaîne `chn`, cette dernière n'étant pas modifiée.
  - *chn*`.indexOf(`*elem*`)` : renvoie la position de la première occurence de la sous-chaîne `elem` dans la chaîne `chn` si elle existe et -1 sinon.
  - *chn*`.splice(`*deb*`,`*fin*`)` : renvoie la sous-chaîne de la chaîne `chn` comprise entre les index `deb` et `fin-1`.

    Exemple 4.4 On considère la variable `let texte="JavaScript"`. Alors :
    - `alert(texte[0])` affiche `J`;
    - `alert(texte.length)` affiche `10`;
    - `alert(texte.toUpperCase())` affiche `JAVASCRIPT`;
    - `alert(texte.toLowerCase())` affiche `javascript`;
    - `alert(texte)` affiche `JavaScript`;
    - `alert(texte.indexOf("a"))` affiche `1`;
    - `alert(texte.indexOf("w"))` affiche `-1`;
    - `let texte1 = texte.slice(1,3); alert(texte1);` affiche `av`.

  - *chn*`.replace(`*old*`,`*new*`)` : renvoie une chaîne de caractères dans laquelle la première occurence de la sous-chaîne `old` a été remplacée par la sous-chaîne `new`; la chaîne initiale n'est pas modifiée.
  - *chn*`.replace(/`*old*`/g,`*new*`)` : renvoie une chaîne de caractères dans laquelle toutes les occurences de la sous-chaîne `old` ont été remplacées par la sous-chaîne `new`; la chaîne initiale n'est pas modifiée.

    Exemple 4.5 Les instructions

    ```
    let texte = "JeveScript";
    let texte1 = texte.replace("e","a");
    alert(texte); alert(texte1);
    let texte2 = texte.replace(/e/g,"a");
    alert(texte); alert(texte2);
    ```

    affichent respectivement `JeveScript`, `JaveScript`, `JeveScript` et `JavaScript`. En particulier, on remarque que dans l'écriture `/old/g`, il n'y a aucun guillemet pour encadrer la chaîne `old`.

4.4 Les tableaux

Les tableaux `JavaScript` sont des données structurées analogues aux listes `Python`. Ils sont définis par des crochets et ses éléments sont séparés par des virgules.

- ⋆ *tableau*`[i]` : renvoie l'élément situé à l'index `i` du tableau (les index commencent à 0).
- ⋆ *tableau*`.length` : renvoie la longueur du tableau.
- ⋆ *tableau*`.push(e1,e2,...,en)` : rajoute les éléments `e1`, ..., `en` à la fin du tableau.
- ⋆ *tableau*`.unshift(e1,e2,...,en)` : rajoute les éléments `e1`, ..., `en` au début du tableau.
- ⋆ *tableau*`.pop()` : retire et renvoie le dernier élément du tableau.
- ⋆ *tableau*`.shift()` : retire et renvoie le premier élément du tableau.
- ⋆ *tableau*`.indexOf(`*elem*`)` : renvoie l'index de la première occurrence de `elem` s'il est présent dans le tableau et `-1` sinon.
- ⋆ *tableau*`.splice(`*pos*`,`*n*`)` : supprime `n` éléments du tableau à partir de l'index `pos`.

Exemple 4.6 On considère le tableau `let tab = ["Sandrine","Mathilde"]`. Alors :

- `alert(tab[0])` affiche `Sandrine`.
- `tab[0] = "Pascal"; alert(tab)` affiche `Pascal, Mathilde`.
- `alert(tab.length)` affiche `2`.
- `tab.push("Laurent","Julie")` modifie `tab` en `[Pascal,Mathilde,Laurent,Julie]`.
- `tab.unshift("Charles")` modifie `tab` en `[Charles,Pascal,Mathilde,Laurent,Julie]`.
- `alert(tab)` affiche `Charles, Pascal, Mathilde, Laurent, Julie`.
- `let premier = tab.shift(); alert(premier)` affiche `Charles`.
- `alert(tab)` affiche `Pascal, Mathilde, Laurent, Julie`.
- `let dernier = tab.pop(); alert(dernier)` affiche `Julie`.
- `alert(tab)` affiche `Pascal, Mathilde, Laurent`.
- `alert(tab.indexOf("Mathilde"))` affiche `1`.
- `alert(tab.indexOf("Sandrine"))` affiche `-1`.
- `tab.splice(1,2); alert(tab)` affiche `Pascal`.

- ⋆ *chn*`.split(sep)` : découpe la chaîne de caractères `chn` en tableau suivant le séparateur `sep`.
- ⋆ *tab*`.join(sep)` : crée une chaîne de caractères à partir du tableau `tab`, les éléments étant séparés par le séparateur `sep`.

Exemple 4.7 On considère la chaîne de caractères `let chn = "Pauline;Guillaume;Alain"`.

- `let tab = chn.split(";")` permet de définir le tableau `[Pauline, Guillaume, Alain]`.
- `let chn2 = tab.join("-:-")` permet de définir la chaîne de caractères `Pauline-:-Guillaume-:-Alain`.

5 Structures de contrôles

5.1 Les tests

5.1.1 Les opérateurs de comparaison

égal	différent	inférieur	supérieur	inférieur strict	supérieur strict
==	!=	<=	>=	<	>

Exemple 5.1 Les instructions

```
3 == 5; "ab" != "cd"; 5 <= 8; 4 >= -3; 3 < 1;
```

renvoient respectivement `False`, `True`, `True`, `True` et `False`.

5.1.2 Les opérateurs logiques

ou	et	non
||	&&	!

Exemple 5.2 Les instructions

```
3==3 or 9>24; 3==3 and 9>24; !(3==3)
```

renvoient respectivement `True`, `False` et `False`.

5.1.3 La fonction `confirm`

- ⋆ `confirm("`*`TexteAffiche`*`")` : ouvre une boîte de dialogue avec le texte `TexteAffiche` et attend une réponse de l'utilisateur ; renvoie `true` ou `false` suivant le choix fait.

Exemple 5.3 L'instruction `confirm("Voulez-vous valider ?");` renvoie `true` si l'on clique sur `OK` et `false` si l'on clique sur `Annuler`.

5.2 Les structures d'embranchements

Lorsque l'on veut exécuter des instructions seulement si une condition est vérifiée ou non, on utilisera une structure d'embranchement.

5.2.1 La structure `si ... alors ...`

La structure algorithmique

```
si une condition est vraie alors
   bloc d'instructions
fin si
```

s'écrit

```
if (condition)
{
   bloc d'instructions
}
```

Exemple 5.4 Le code `JavaScript` suivant permet d'afficher `TRUE` si le nombre `x` est positif :

```
if (x >= 0)
{
    alert('TRUE');
}
```

5.2.2 La structure `si ... alors ... sinon ...`

La structure algorithmique

```
si une condition est vraie alors
   bloc d'instructions 1
sinon
   bloc d'instructions 2
fin si
```

s'écrit

```
if (condition)
{
   bloc d'instructions 1
}
else
{
   bloc d'instructions 2
}
```

Exemple 5.5 Le code `JavaScript` suivant permet d'afficher `négatif` ou `positif` suivant le signe de `x` :

```
if (x<=0)
{
    alert('négatif');
}
else
{
    alert('positif');
}
```

5.2.3 Les structures d'embranchements multiples

⋆ La structure algorithmique

```
si condition 1 est vraie alors
   bloc d'instructions 1
sinon si condition 2 est vraie alors
   bloc d'instructions 2
... (on peut mettre autant de sinon si que nécessaire)
sinon
   dernier bloc d'instructions
fin si
```

s'écrit

```
if (condition 1)
{
   bloc d'instructions 1
}
else if (condition 2)
{
   bloc d'instructions 2
}
...
else
{
   dernier bloc d'instructions
}
```

Exemple 5.6 Le code `JavaScript` suivant permet d'afficher le signe de `x` :

```
if (x<0)
{
    alert('strictement négatif');
}
else if (x==0)
{
    alert('nul');
}
else
{
    alert('strictement positif');
}
```

⋆ Le langage `JavaScript` admet également une autre structure d'embranchement appelée `switch` :

```
switch (variable ou expression)
{
   case valeur 1:
     bloc d'instructions 1
     break;
   case valeur 2:
     bloc d'instructions 2
     break;
   ...
   case valeur n:
     bloc d'instructions n
     break;
   default:                    // optionnel
     bloc d'instructions
}
```

Lorsque la structure `switch` est utilisée dans une fonction qui a besoin de renvoyer une valeur, on utilise la commande `return` dans les blocs d'instructions.

Exemple 5.7 Etant donné une variable `fuit`, afficher 25 si `fruit='Orange'`, afficher 40 si `fruit='Pomme'` et afficher 0 dans tous les autres cas.

```
switch (fruit)
{
    case 'Orange':
        alert('25');
        break;
    case 'Pomme':
        alert('40');
        break;
    default:
        alert('0');
}
```

5.3 La structure de boucle `pour`

La structure

```
pour i de m à n faire
   bloc d'instructions
fin pour
```

s'écrit

```
for(let i=m;i<=n;i=i+1)
{
   bloc d'instructions
}
```

Remarque 5.8 L'instruction `i=i+1` correspond à l'incrémentation du compteur de boucle `i`. Avec cette instruction, l'incrémentation est de 1. On peut bien sûr la modifier et effectuer des incrémentations de 2, 3, voire négative.

Exemple 5.9 Parcours d'une chaîne de caractères sur les index :

```
let texte = 'JavaScript';
for(let i=0; i<texte.length; i++)
{
    alert(texte[i]);
}
```

Remarque 5.10 On peut parcourir de la même façon un tableau.

5.4 La structure de boucle `tant que`

La structure

```
tant que une condition reste vraie faire
   bloc d'instructions
fin tant que
```

s'écrit

```
while(condition)
{
   bloc d'instructions
}
```

Exemple 5.11 Parcours d'un tableau indexé numériquement :

```
let tab = [3,2,1,'Partez !'];
let i = 0;
while(i<tab.length)
{
    alert(tab[i]);
    i+=1;
}
```

6 Les fonctions

Les fonctions JavaScript se manipulent comme les fonctions Python. Leur seule différence réside dans la syntaxe. En JavaScript, la syntaxe d'une fonction est la suivante :

```
function nom_de_la_fonction(liste_paramètres )
{
   instruction 1;
   ...
   instruction n;
   return liste_valeurs
}
```

Remarque 6.1

- La liste des paramètres peut être vide.
- La fonction peut ne pas renvoyer de valeurs (on parle alors de *procédure*).
- Comme dans les autres langages de programmation, une fonction peut faire appel à plusieurs *sous-fonctions*.

7 Gestion du DOM

7.1 Attributs événementiels

La plupart des éléments HTML admettent des attributs événementiels de façon à ce qu'un événement JavaScript soit activé :

- onclick : exécute un code JavaScript lorsque l'on clique gauche avec la souris ;
- ondblclick : exécute un code JavaScript lorsque l'on double-clique gauche avec la souris ;
- onmousedown : exécute un code JavaScript lorsque l'on clique sur un bouton de la souris ;
- onmouseenter : exécute un code JavaScript lorsque le pointeur de la souris entre dans l'élément ;
- onmouseleave : exécute un code JavaScript lorsque le pointeur de la souris quitte l'élément ;
- onmousemove : exécute un code JavaScript lorsque le pointeur de la souris est présent dans l'élément et déplacé à l'intéreur de celui-ci ;
- onmouseout : exécute un code JavaScript lorsque le pointeur de la souris est à l'extérieur de l'élément ;
- onmouseover : exécute un code JavaScript lorsque le pointeur de la souris entre dans l'élément ou dans l'un de ses fils ;
- onmouseup : exécute un code JavaScript lorsque l'on relâche un des boutons de la souris.

Remarque 7.1 Tous ces attributs ne sont pas valables pour tous les éléments HTML. On consultera internet pour plus de détails.

Pour utiliser ces attributs, on utilise la syntaxe suivante :

```
<nom_balise nom_attribut_even="code_JavaScript">
```

Exemple 7.2

- Le code HTML

```
<button onclick="alert('Bonjour !');">Hello World</button>
```

  permet de créer un bouton contenant le texte Hello World et affichant dans une boîte de dialogue le texte Bonjour ! lorsque l'on clique dessus.

- Le code HTML

```
<span onclick="javascript:location.href='https://www.google.fr'">Google</span>
```

  permet de spécifier le texte Google pour que l'on puisse accéder au moteur de recherche www.google.fr lorsque l'on clique dessus.

7.2 Les sélecteurs `JavaScript`

Pour pouvoir manipuler le DOM, il est nécessaire de récupérer les éléments `HTML` sur lesquels on veut agir. On dispose de plusieurs fonctions suivant le type de sélection que l'on veut faire :

- `document.getElementById(`*`nom_id`*`)` : renvoie l'élément d'identifiant `nom_id` s'il existe et `null` sinon.
- `document.getElementsByClassName(`*`nom_class`*`)` : renvoie un tableau de tous les éléments dont l'attribut `class` est `nom_class` et un tableau vide s'il n'en existe pas.
- `document.getElementsByTagName(`*`nom_balise`*`)` : renvoie un tableau de tous les éléments dont les noms de balises sont égaux à `nom_balise` et un tableau vide s'il n'en existe pas.

Exemple 7.3

- L'instruction `document.getElementById('Rouge')` renvoie l'élément d'identifiant `Rouge` et `null` s'il n'existe pas.
- L'instruction `document.getElementsByTagName('h1')` renvoie un tableau contenant tous les éléments de balises `<h1>` et un tableau vide s'il n'en existe pas.

Remarque 7.4 Un tableau vide est un tableau de longueur 0.

Remarque 7.5 On peut aussi utiliser le mot-clé `this` pour pouvoir manipuler directement l'élément admettant l'attribut événementiel (on le passe en général en paramètre de la fonction à exécuter).

Une fois le ou les éléments sélectionnés, on peut agir dessus : soit en modifiant le contenu, soit en modifiant un attribut existant (style, border, width, etc.).

7.3 Modification du contenu d'un élément

- *`elem`*`.innerHTML` : renvoie une chaîne de caractères contenant le contenu textuel de l'élément `elem` défini préalablement par un sélecteur.
- *`elem`*`.innerHTML = `*`nouveau_texte`* : remplace le contenu textuel de l'élément `elem` défini préalablement par un sélecteur par la chaîne de caractères `nouveau_texte`.

  Exemple 7.6 L'instruction `E.innerHTML = 'Texte modifié'` remplace sur la page web le texte initial de l'élément `E` par le texte `Texte modifié`.

- *`elem`*`.insertAdjacentHTML(`*`pos`*`,`*`texte_ajout`*`)` : ajoute au contenu textuel de l'élément `elem` défini préalablement par un sélecteur le texte `texte_ajout` à la position `pos`. Cette position peut-être `'beforebegin'`, `'afterbegin'`, `'beforeend'` ou `'afterend'` comme sur le schéma suivant :

```
<!-- beforebegin -->
<nom_balise>
   <!-- afterbegin -->
   texte existant
   <!-- beforeend -->
</nom_balise>
<!-- afterend -->
```

7.4 Modification du style d'un élément

- `window.getComputedStyle(`*`elem`*`).getPropertyValue(`*`attribut_css`*`)` : renvoie la valeur `CSS` de l'attribut `attribut_css` de l'élément `elem` préalablement défini à l'aide d'un sélecteur.
- *`elem`*`.style.`*`attribut_css = val`* : affecte la valeur `val` à l'attribut `attribut_css` de l'élément `elem` préalablement défini à l'aide d'un sélecteur.

Exemple 7.7

- L'instruction `window.getComputedStyle(element).getPropertyValue('color')` renvoie la valeur de l'attribut `color` de l'élément `element`.
- L'instruction `E.style.display='block'` permet d'attribuer le style `block` à l'élément `E`.
- L'instruction `E.style.color='red'` permet d'attribuer la couleur rouge à l'élément `E`.
- L'instruction `E.style.backgroundColor='blue'` permet d'attribuer la couleur bleu à l'arrière-plan de l'élément `E`.

Remarque 7.8

- Les attributs `CSS` s'écrivant avec des traits d'union sont en général écrits tout attaché avec des majuscules pour différencier les différents mots. Par exemple, `background-color` est l'attribut `CSS`, mais on le recherche sous la forme `backgroundColor`.
- Les couleurs sont toujours renvoyées sous la forme `'rgb(r, v, b)'` (ne pas oublier les espaces).

7.5 Modification d'un attribut quelconque d'un élément

- ⋆ *`elem.nom_attribut`* : renvoie une chaîne de caractères contenant la valeur de l'attribut `nom_attribut` de l'élément `elem` défini préalablement par un sélecteur.
- ⋆ *`elem.nom_attribut = val`* : affecte la valeur `val` (donnée sous forme d'une chaîne de caractères) à l'attribut `nom_attribut` de l'élément `elem` défini préalablement par un sélecteur.

Exemple 7.9 On considère l'élément `HTML` suivant (barre de progression) :

```
<progress id='barre' value='0' max='100'>0</progress>
```

On récupère cet élément par un sélecteur sur l'identifiant : `let barre = document.getElementById('barre');`
On peut agir sur l'attribut `value` de la façon suivante :

- ⋆ `barre.value` : renvoie la valeur de l'attribut `value`, c'est-à-dire `'0'` ;
- ⋆ `barre.value = '25'` : affecte à l'attribut `value` la valeur `'25'`.