Semaine 13

Fonctions utilitaires et résolution de problèmes

Intro. à la programmation

Menu du jour

- Fonctions pour chaînes de caractères
- Fonctions mathématiques
- Commenter ses fonctions
- Apprendre d'autres langages
- Résolution de problèmes

- Fonctions préexistantes
  - Depuis le début de la session, nous avons vu quelques **fonctions préexistantes** en JavaScript : Math.random(), alert(...), console.log(...), etc.
  - Aujourd'hui, nous allons en voir quelques-unes de plus qui pourraient vous servir dans le futur.

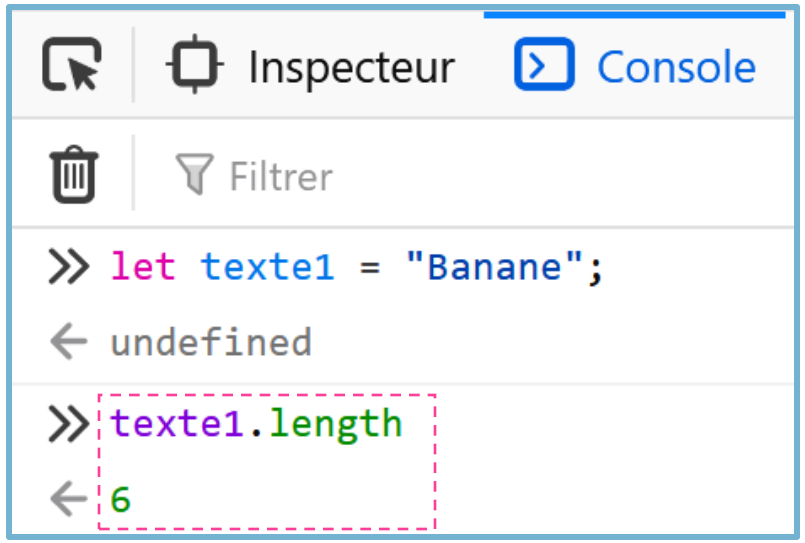
Fonctions pour chaînes de caractères

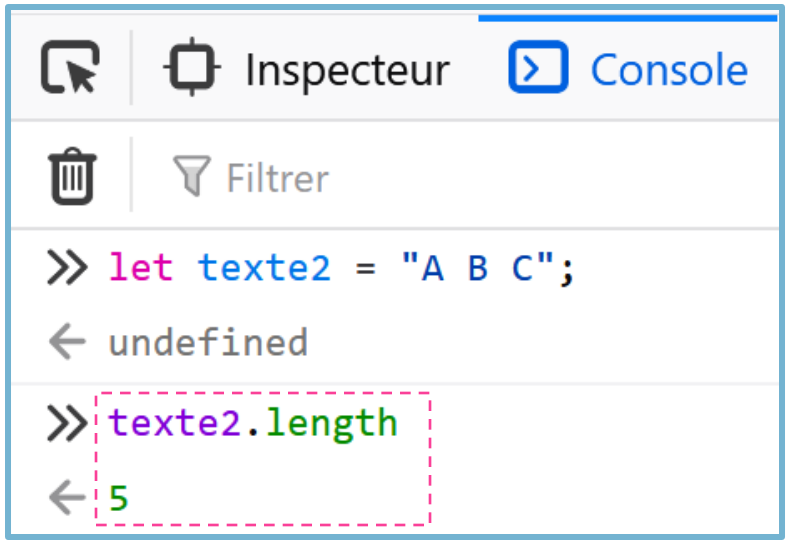
- Fonctions pour les chaînes de caractères
  - .length
  - .substring()
  - .replace()
  - .replaceAll()
  - .toUpperCase()
  - .toLowerCase()
  - .charAt()

Fonctions pour chaînes de caractères

❖ Obtenir la taille d'une chaîne de caractères

- ♦ .length fonctionne aussi avec les chaînes de caractères !
  - .length n'est pas tout à fait une fonction. (Donc pas de **parenthèses**) C'est une propriété qui contient le nombre de caractères dans la chaîne.



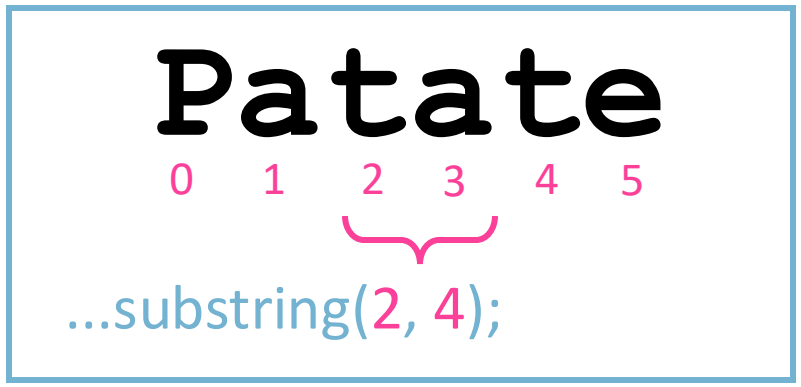


Fonctions pour chaînes de caractères

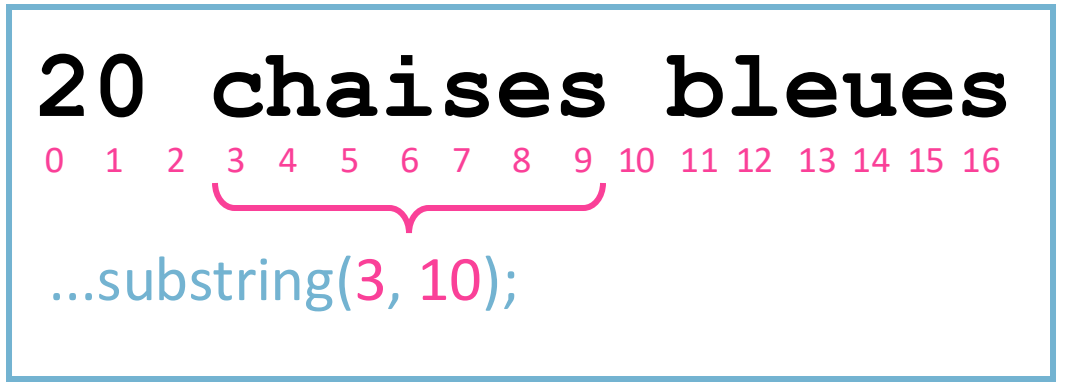
❖ Obtenir une partie de la chaîne de caractères

- ♦ .substring(...) retourne une « **sous-chaîne** » :
  - Le **premier paramètre** est l'index du premier caractère à conserver.
  - Le **deuxième paramètre** est l'index du caractère où la sous-chaîne s'arrête. (Exclus)

```
>> let texteComplet = "Patate";
   let textePartiel = texteComplet.substring(2, 4);
← undefined
>> textePartiel
← "ta"
```




```
>> let texteComplet = "20 chaises bleues";
   let textePartiel = texteComplet.substring(3, 10);
← undefined
>> textePartiel
← "chaises"
```



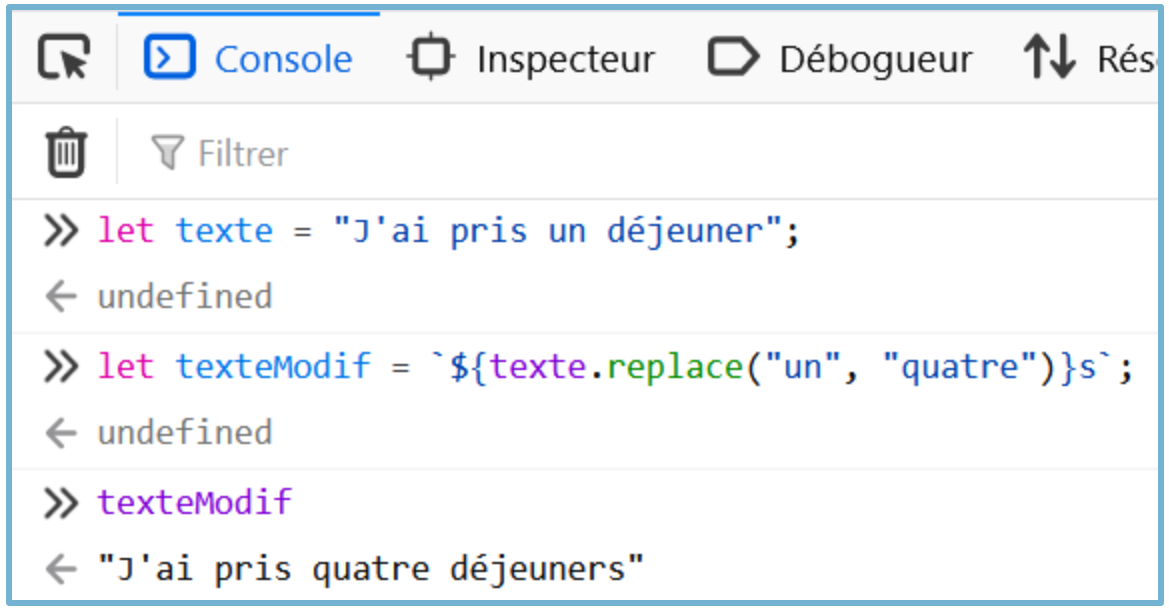
Fonctions pour chaînes de caractères

❖ Remplacer un caractère ou un groupe de caractères

- ♦ .replace(...) permet de **remplacer** un groupe de caractères. Elle retourne la chaîne de caractères modifiée.
  - Le **premier paramètre** est le groupe de 1+ caractère(s) à remplacer.
  - Le **deuxième paramètre** est le groupe de remplacement.

```
>> let texte = "J'aime les licornes";
   let texteModif = texte.replace("cornes", "vres");
<- undefined
>> texteModif
<- "J'aime les livres"
```

```
>> let texte = "Je suis très content";
   let texteModif = texte.replace("très ", "");
<- undefined
>> texteModif
<- "Je suis content"
```



Ci-dessus, on peut voir que même si « un » apparait également dans le mot « déjeuner », **seule la première occurrence est remplacée**.

Fonctions pour chaînes de caractères

❖ Remplacer plusieurs groupes de caractères

- ♦ .replaceAll(...) est similaire à .replace(), mais elle remplace **toutes les occurrences** du texte à remplacer.
  - Le **premier paramètre** est le groupe de 1+ caractère(s) à remplacer.
  - Le **deuxième paramètre** est le groupe de remplacement.

```
>> let texte = "J'ai pris un déjeuner";
   let texteModif = texte.replaceAll("un", "quatre");
← undefined
>> texteModif
← "J'ai pris quatre déjequatreer"
```

```
>> let texte = "À cause de mon rhume, mes amis se moquent de moi";
   let texteModif = texte.replaceAll("m", "b");
← undefined
>> texteModif
← "À cause de bon rhube, bes abis se boquent de boi"
```

Fonctions pour chaînes de caractères

- Tout mettre en majuscules / minuscules
  - .toLowerCase() et .toUpperCase() retournent la chaîne de caractères, mais complètement en minuscules / majuscules.
    - Ça n'a aucun impact sur les caractères qui ne sont pas des lettres.

```
>> let texte = "Je trouve que Sophie est normale.";
   let texteMaj = texte.toUpperCase();
   let texteMin = texte.toLowerCase();
<- undefined
>> texteMaj
<- "JE TROUVE QUE SOPHIE EST NORMALE."
>> texteMin
<- "je trouve que sophie est normale."
```

Fonctions pour chaînes de caractères

- Obtenir un caractère à une position précise dans une chaîne
  - .charAt(...) retourne le caractère situé à l'index de notre choix dans une chaîne de caractères.
    - Un **paramètre** est nécessaire : c'est l'index du caractère demandé.

```
>> let texte = "zèbre";
   let lettre = texte.charAt(0);
← undefined
>> lettre
← "z"
```

```
>> let texte = "Omar fait de la bicyclette.";
   let lettre = texte.charAt(4);
← undefined
>> lettre
← " "
```

```
>> let texte = "Simone va ouvrir une buvette.";
   let lettre = texte.charAt(texte.length - 1);
← undefined
>> lettre
← "."
```

Fonctions mathématiques

- Fonctions mathématiques
  - Math.round()
  - Math.ceil() et Math.floor()
  - Math.min() et Math.max()
  - Math.random()

Fonctions mathématiques

❖ Arrondir un nombre

- ♦ Math.round(...) retourne un nombre arrondi au nombre entier le plus près.

```
>> let distance = 24.1563235;
<- undefined
>> let texte = `J'habite à ${Math.round(distance)} kilomètres d'ici !`;
<- undefined
>> texte
<- "J'habite à 24 kilomètres d'ici !"
```

Nombre arrondi **vers le bas**

```
>> let age = 24.67;
   let ageArrondi = Math.round(age);
<- undefined
>> ageArrondi
<- 25
```

Nombre arrondi **vers le haut**

Fonctions mathématiques

- Arrondir un nombre forcément vers le bas / vers le haut
  - Math.floor(...) retourne un nombre arrondi vers le bas.
  - Math.ceil(...) retourne un nombre arrondi vers le haut.

```
>> let age = 17.98;
<- undefined
>> `J'ai ${Math.floor(age)} ans.`
<- "J'ai 17 ans."
```

Généralement, quand on mentionne son âge, on arrondit toujours vers le bas 👶

```
>> let longueur = 13.05;
<- undefined
>> `J'ai besoin de ${Math.ceil(longueur)} mètres de corde.`
<- "J'ai besoin de 14 mètres de corde."
```

Si on ne veut pas manquer de quelque chose, on l'arrondit vers le haut 🙄

Fonctions mathématiques

❖ Obtenir le maximum ou le minimum

- ♦ Math.max(...) retourne la valeur maximale entre plusieurs nombres.
- ♦ Math.min(...) retourne la valeur minimale entre plusieurs nombres.

```
>> let a = 5;
   let b = 3;
← undefined
>> Math.max(a, b);
← 5
```

```
>> Math.min(2, 4, 1, 3)
← 1
```

Autant pour max que pour min, on peut mettre 2 valeurs ou plus, séparées par des virgules.

Cette fonction permet de soigner le joueur (lui redonner de la vie) sans dépasser le maximum de 100. Ici, gVieJoueur + valeurSoin donnait 143, alors 100 a été choisi par Math.min().

```
function soignerJoueur(valeurSoin){

    gVieJoueur = Math.min(100, gVieJoueur + valeurSoin);

}
```

```
>> gVieJoueur
← 93
>> soignerJoueur(50);
← undefined
>> gVieJoueur
← 100
```

Fonctions mathématiques

❖ Obtenir un nombre aléatoire

- ♦ Nous avons déjà vu Math.random(), mais voici des exemples un peu plus sophistiqués.

Obtenir un nombre aléatoire entre 0 et 0.99999...

```
>> let nombreAleatoire = Math.random();
<- undefined
>> nombreAleatoire
<- 0.8758528200387565
```

Obtenir un nombre aléatoire entre 0 et 4.99999...

```
>> let nombreAleatoire = Math.random() * 5;
<- undefined
>> nombreAleatoire
<- 3.814723212502409
```

Obtenir un nombre aléatoire entre 5 et 9.99999...

```
>> let nombreAleatoire = 5 + Math.random() * 5;
<- undefined
>> nombreAleatoire
<- 7.241046042976543
```

Fonctions mathématiques

❖ Obtenir un nombre aléatoire

- ♦ Nous avons déjà vu Math.random(), mais voici des exemples un peu plus sophistiqués.

Obtenir un nombre aléatoire parmi 1, 2, 3, 4 et 5.
(Arrondir vers le bas élimine tous les nombres à virgule !)

```
>> let nombreAleatoire = 1 + Math.floor(Math.random() * 5);
<- undefined
>> nombreAleatoire
<- 3
```

Recette pour obtenir un nombre entier entre X et Y :

```
X + Math.floor(Math.random() * (Y - X + 1))
```

Exemple pour un nombre entre 1 et 10 :

```
>> let nombre = 1 + Math.floor(Math.random() * 10);
```

Exemple pour un nombre entre 17 et 23 :

```
>> let nombre = 17 + Math.floor(Math.random() * 7);
```

(23 - 17 + 1 donne 7)

Commenter ses fonctions

❖ Commentaires dans le code

- ♦ Nous les connaissons depuis la semaine 2, mais nous ne les avons jamais rédigés nous-mêmes !

- ♦ TP3 : Vous devrez **créez vos propres fonctions** et vous devrez … les **commenter** !
  - ○ Lorsqu'on crée du code, c'est important de le commenter (le décrire) pour que nos collègues puissent **comprendre** et naviguer **facilement** notre travail.
  - ○ Les prochaines diapos expliquent comment rédiger de bons commentaires.

Commenter ses fonctions

❖ Commenter une fonction

♦ Points clés :

1. Décrire brièvement l'utilité de la fonction. (Sans entrer dans des détails trop techniques)

2. Si la fonction reçoit des paramètres, que représentent-ils ?

3. Si la fonction retourne une valeur, que représente-t-elle ?

♦ Exemple 1

```
// Reçoit une valeur numérique au choix, un minimum et un maximum en paramètres.
// Retourne true si la valeur est située entre le minimum et le maximum fournis
// et false sinon.
function respecteIntervalle(valeur, min, max){

    if(valeur >= min && valeur <= max) {
        return true;
    }
    return false;

}
```

Commenter ses fonctions

❖ Commenter une fonction

♦ Exemple 2 et 3

```
// À l'aide d'un chemin vers une image source et d'une largeur en nombre
// de pixels, crée un nouvel élément img et le retourne.
function creerImage(source, largeur){

    let image = document.createElement("img");
    image.style.width = largeur + "px";
    image.classList.add("thumbnail");
    image.src = source;
    image.alt = "Miniature de photo";
    return image;

}
```

```
// Si le jeu n'est pas déjà en cours, lance le jeu et réinitialise
// le score.
function lancerJeu(){

    if(gJeuActif == false){

        gJeuActif = true;
        gScore = 0;
        document.querySelector(".score").textContent = `Score : ${gScore}`;
        gPlanif = setInterval(miseAJour, 50);

    }

}
```

Commenter ses fonctions

❖ Commenter une fonction

♦ **Erreurs** typiques

- Trop décrire la fonction et aller dans les détails techniques

```
// Si gJeuActif veut false, met gJeuActif à true, puis met gScore à 0,
// puis met à jour le contenu textuel de .score dans la page et lance
// un planificateur à intervalle qui appelle la fonction miseAJour()
// toutes les 50 millisecondes.
function lancerJeu(){

    if(gJeuActif == false){

        gJeuActif = true;
        gScore = 0;
        document.querySelector(".score").textContent = `Score : ${gScore}`;
        gPlanif = setInterval(miseAJour, 50);

    }

}
```

```
// Reçoit une valeur numérique au choix, un minimum et un maximum en paramètres.
// Si valeur est plus grande ou égale à min et que valeur est plus petite ou égale à
// max, on retourne true, sinon on retourne false.
function respecteIntervalle(valeur, min, max){

    if(valeur >= min && valeur <= max) {
        return true;
    }
    return false;

}
```

❌ Au lieu de **résumer conceptuellement en français** les fonctions, on décrit les lignes de code de la fonction. ❌

Commenter ses fonctions

❖ Commenter une fonction

♦ **Erreurs** typiques

- Oublier de parler des paramètres

```
// Retourne une valeur multipliée par 3
function triplerValeur(n){

    return n * 3;
}
```

Quelle valeur ?

Amélioration

```
// Retourne la valeur numérique reçue en paramètre, multipliée par 3
function triplerValeur(n){

    return n * 3;
}
```

- Oublier de parler du retour

```
// Calcule le prix avec taxe pour un prix reçu en paramètre.
function prixAvecTaxes(prix){

    return prix * 1.15;
}
```

Et … ? On affiche le prix dans la page ?
On le retourne ? Que fait-on avec ?

Amélioration

```
// Calcule le prix avec taxe pour un prix reçu en paramètre et le retourne.
function prixAvecTaxes(prix){

    return prix * 1.15;
}
```

Apprendre d'autres langages

- Bonne nouvelle !
  - Les langages de programmation ont énormément de similarités !

```javascript
let a = 5;

if(a > 100){
    a += 3;
}
```

Code en JavaScript

```java
int a = 5;

if(a > 100){
    a += 3;
}
```

Code en Java

```csharp
int a = 5;

if(a > 100){
    a += 3;
}
```

Code en C#

```cpp
int a{ 5 };

if(a > 100){
    a += 3;
}
```

Code en C++

```lua
local a = 5

if a > 100 then
    a += 3
end
```

Code en Lua

```ruby
a = 5

if a > 100
    a += 3
end
```

Code en Ruby

```python
a = 5

if a > 100:
    a += 3
```

Code en Python

- Bien entendu, c'est un **petit échantillon** et il y a plus de **différences** que cela.

- Cela dit, lorsqu'on connait déjà un langage de programmation, en apprendre d'autres est de plus en plus facile !

Apprendre d'autres langages

❖ Apprendre d'autres langages

- ♦ Bien que nous ayons appris beaucoup de notions en JavaScript, nous n'avons vu **que les bases** ! Si vous explorez d'autres langages, vous rencontrerez plusieurs nouvelles notions qui ne vous seront pas familières.

- ♦ Malgré tout, n'ayez pas peur d'apprendre d'autres langages dès maintenant !
  - ○ Il faut apprendre le langage C# pour exploiter Unity en profondeur.
  - ○ Il faut apprendre Kotlin, Java ou autre pour faire des applications Android.
  - ○ Il faut apprendre C#, Java, Python ou autre pour faire un serveur Web.
  - ○ Il faut apprendre SQL pour exploiter les bases de données relationnelles.
  - ○ etc.

- ♦ Selon les types de projets qui vous intéressent, il y a beaucoup à apprendre
  - ○ Bien entendu, dans l'univers TIM, vous n'aurez pas forcément besoin d'apprendre beaucoup de langages de programmation 😇

❖ Résolution de problèmes

- ♦ Nous avons terminé de faire le tour des bases en JavaScript 🥳
- ♦ Pour ce dernier labo, nous nous concentrerons surtout sur la « résolution de problèmes ».
  - ○ (Sauf les 2 premiers exercices, qui sont pour les fonctions de texte et mathématiques)
  - ○ Vous ferez face à des exercices plus « libres » et difficiles que d'habitude :
    - Vous pouvez utiliser **n'importe quelle notion** apprise cette session, **tant que le résultat fonctionne** !
    - Le but est de réfléchir et de réussir à exploiter les bases que nous avons acquises durant toute la session.
    - 📝 Attention ! À l'**examen final**, un exercice parmi les exercices 3 à 10 sera présent. Il est donc important de comprendre tous les exercices, quitte à prendre le temps, avec l'enseignant(e), de refaire ceux qui vous donnent plus de fil à retordre.

❖ Résolution de problèmes

♦ L'aide-mémoire est maintenant complet. En effectuant les exercices de résolution de problème, essayez de garder à l'esprit toutes les notions qu'on a vues !

Besoin de manipuler des variables ou des valeurs ?

let

=, +=, -=

+, -, *, /

Fonctions mathématiques

Concaténation

Fonctions pour chaînes de caractères

Seulement exécuter du code si une condition est respectée ?

if

else

else if

.. ? .. : ..

<, <=, >, >=, ==, !=

&&, ||

switch

Répéter ou réutiliser du code similaire ?

function()

function(a, b)

while

do while

for

Besoin de modifier la page Web ?

querySelector

querySelectorAll

.textContent

.style.propriété

.classList...

.nomAttribut

.createElement

.addEventListener

Besoin de manipuler beaucoup de valeurs liées ?

[1, 2, 3]

.length

.pop

.push

.splice

Autres ...

currentTarget

setTimeout

setInterval

clearInterval

alert

console.log