Algorithmique

Définition 1

Ensemble de règles opératoires propres à un calcul.

Définition 2

Méthode de calcul qui indique la démarche à suivre
pour résoudre une série de problèmes équivalents
en appliquant dans un ordre précis une suite finie de règles.

Dictionnaire de l'Académie française

Définition 3

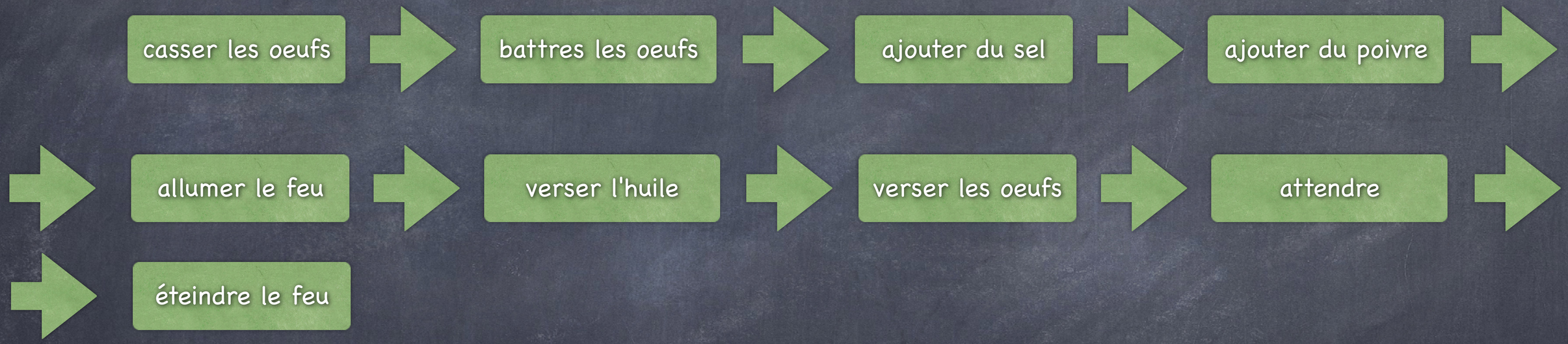
*Un algorithme est un procédé de **calcul**;*
*il décrit une succession **d'opérations** à exécuter*
*dans un certain **ordre** et sous certaines **conditions***
*pour passer des données de base aux **résultats**.*

*Il correspond à un processus, découpé en **étapes***
*dont **l'enchaînement** permet d'arriver à un **but fixé**.*

Conclusion

- Données en entrée
- Contenu des opérations
- Ordre des opérations
- Données en sortie (résultat)

Faire une omelette



Ouvrir une porte

Etat initial : porte fermée

- ✓ Prendre la poignée
- ✓ Tourner la poignée
- ✓ Tirer la porte

Etat final : porte ouverte

Problèmes

- ✓ Si la porte et déjà ouverte
- ✓ Si la porte est verrouillée
- ✓ S'il faut pousser et non tirer la porte

Ouvrir une porte

Etat initial : porte fermée

- ✓ Si la porte est fermée
  - ✓ Prendre la poignée
  - ✓ Tourner la poignée
  - ✓ Tirer la porte

Etat final : porte ouverte

Ouvrir une porte

Etat initial : porte fermée

- ✓ Si la porte est fermée
  - ✓ Si la porte est verrouillée
    - ✓ Mettre la clef dans la serrure
    - ✓ Tourner la clef (combien de fois ?)
  - ✓ Prendre la poignée
  - ✓ Tourner la poignée
  - ✓ Tirer la porte

Etat final : porte ouverte

Ouvrir une porte

Etat initial : porte fermée

- ✓ Si la porte est fermée
  - ✓ Si la porte est verrouillée
    - ✓ Mettre la clef dans la serrure
    - ✓ Tant que l'on peut tourner la clef
      - ✓ Tourner la clef
  - ✓ Prendre la poignée
  - ✓ Tourner la poignée
  - ✓ Tirer la porte

Etat final : porte ouverte

Ouvrir une porte

Etat initial : porte fermée

- ✓ Si la porte est fermée
  - ✓ Si la porte est verrouillée
    - ✓ Mettre la clef dans la serrure
    - ✓ Tant que l'on peut tourner la clef
      - ✓ Tourner la clef
  - ✓ Prendre la poignée
  - ✓ Tourner la poignée
  - ✓ Si on peut tirer la porte
    - ✓ Tirer la porte
  - ✓ Sinon
    - ✓ pousser la porte

Etat final : porte ouverte

Ouvrir une porte

Etat initial : porte fermée

- ✓ Si la porte est fermée
  - ✓ Si la porte est verrouillée
    - ✓ Mettre la clef dans la serrure
    - ✓ Tant que l'on peut tourner la clef
      - ✓ Tourner la clef
  - ✓ Prendre la poignée
  - ✓ Tourner la poignée
  - ✓ Si on peut tirer la porte
    - ✓ Tirer la porte
  - ✓ Sinon
    - ✓ pousser la porte

Etat final : porte ouverte

algo déverrouiller

algo ouvrir porte

Ouvrir une porte

Etat initial : porte fermée

- ✓ Si la porte est fermée
  - ✓ **Déverrouiller la porte** — algo déverrouiller
  - ✓ Prendre la poignée
  - ✓ Tourner la poignée
  - ✓ **Ouvrir la porte** — algo ouvrir porte

Etat final : porte ouverte

Notions

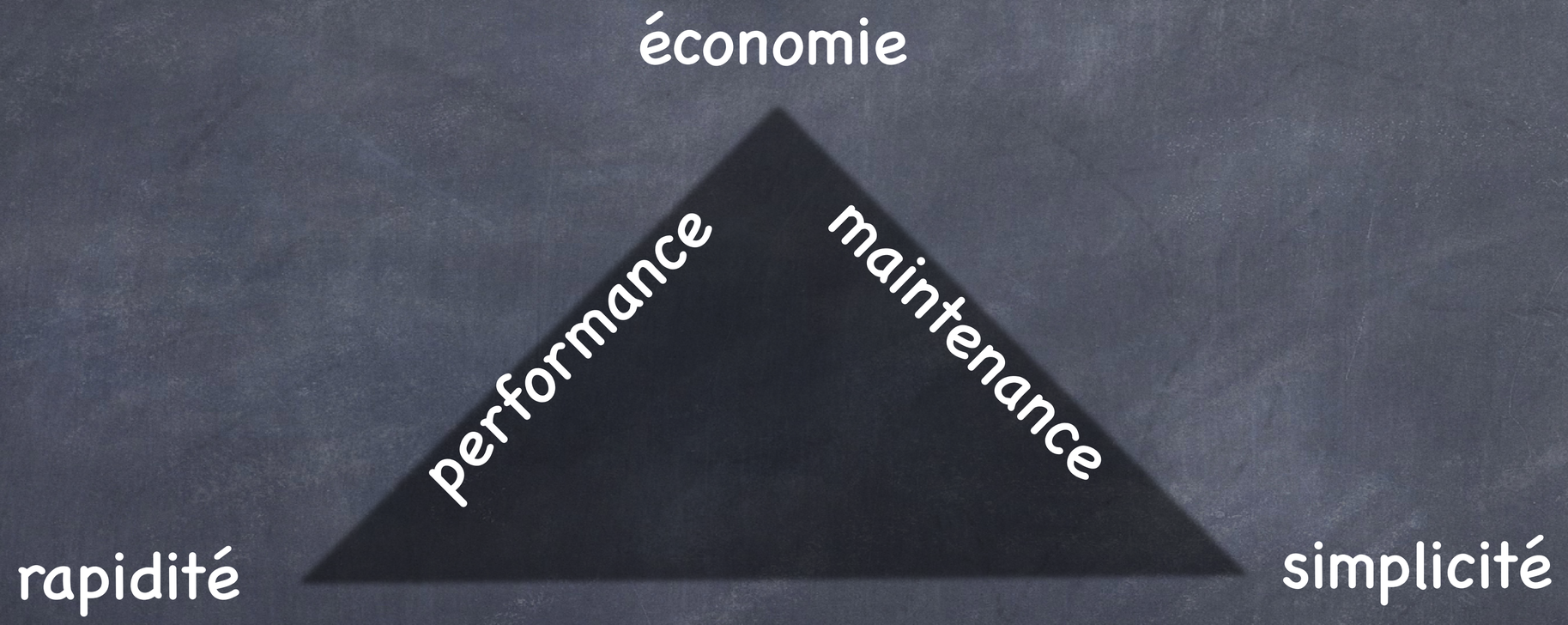
- ✓ Alternatives
- ✓ Répétitions
- ✓ Découpage en sous algorithmes

Observations

L'erreur est humaine
mais il faut un ordinateur
pour la transformer en catastrophe

- ✓ Tester le contexte (état initial)
- ✓ Tester le résultat (état final)
- ✓ Détailler les opérations
  (l'ordinateur est dépourvu de bon sens)

Qualités des algorithmes



A vous

Ecrire l'algorithme qui décrit les opération pour :

Prendre une photo avec votre téléphone portable
Démarrer votre voiture et sortir d'un stationnement en épis
Se laver les dents

Enregistrer et payer ses achats à une caisse automatique
du point de vue de l'utilisateur
du point de vue de la caisse

Algo = recette de cuisine

Cuisine

- ✓ Ingrédients
- ✓ Ustensiles
- ✓ Opérations (utilisation des ustensiles sur les ingrédients)

Algo

- ✓ Données
- ✓ Instructions
- ✓ Expressions (utilisation des instructions sur les données)

Calculer un prix TTC

- ✓ Demander prix HT
- ✓ Demander taux de TVA en %
- ✓ Calculer TTC = HT + (HT * (TVA/100))
- ✓ Afficher TTC

Quelle sont les données ici ?

Les types de données élémentaires

- ✓ du texte
- ✓ des nombres
- ✓ des réponses oui/non (booléens)

Les opérations élémentaires

- ✓ addition
- ✓ soustraction
- ✓ multiplication
- ✓ division (entière ou fractionnelle)
- ✓ modulo (reste de la division entière)
- ✓ concaténation
- ✓ comparaison

Les types de données complexes

- ✓ des dates et des heures
- ✓ des listes de données
- ✓ des fonctions (sous algorithme)
- ✓ des objets (un bouton, une fenêtre...)

Javascript

Code dans la page


```
<script>
  instructions javascript;
</script>
```

uniquement pour du code spécifique à une page, ne permet pas la factorisation

Code dans un fichier .js


```
<script src="js/mons-script.js">
</script>
```

Permet la factorisation et le partage du code entre les pages, solution la plus souvent adoptée

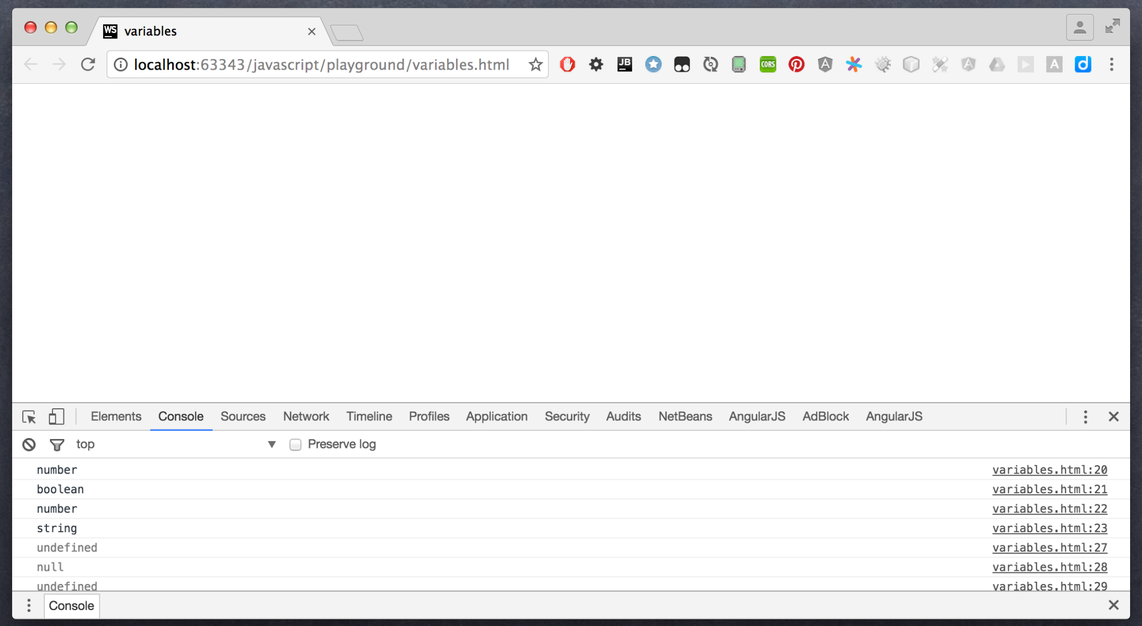
Attention, il ne faut pas mettre de code javaScript entre les ouverture et fermeture de balise

Premier script


```
<script>

 //affiche une boite
 //d'alerte
 alert('coucou');

</script>
```

- ✓ Le script s'exécute au chargement de la page
- ✓ Les instructions se terminent par un ; (facultatif)
- ✓ // pour un commentaire mono-ligne

Afficher la console



Tous le navigateurs
Clic droit - Inspecter

Firefox
CTRL + MAJ + k
CMD + OPT + k

Chrome
CTRL + MAJ + J
CMD + OPT + J
Clic droit - Inspecter

Internet Explorer
F12

Safari
Clic droit - Inspecter

Les variables

Définition

- ✓ Une information stockée en mémoire vive pendant l'exécution d'un programme
- ✓ Au cours de l'exécution du programme, cette information peut être lue ou modifiée

Nom des variables

- ✓ Pas d'espaces
- ✓ Pas d'accents
- ✓ Pas de caractères spéciaux ou de ponctuation
  hormis $ et _
- ✓ Ne peut commencer par un chiffre
  (mais peut en contenir après le premier caractère)
- ✓ Eviter les mots réservées de JavaScript
- ✓ Attention à la casse, age et Age sont deux variables différentes pour JavaScript

Nomenclature

- ✓ Underscore, chaque mot est séparé par un tiret bas
- ✓ CamelCase, la première lettre est en minuscule, les mots suivants débutent par une majuscule
- ✓ PascalCase, tous les mots débutent par une majuscule

Quelques conseils

- ✓ Noms explicites, indiquent le rôle, le contenu et éventuellement l'unité
  - ✓ isAuthenticated
  - ✓ unitPrice, totalBeforeTax
  - ✓ buildingLengthInMeters, HalfLifeInYears
- ✓ Adopter une nomenclature et s'y tenir
- ✓ Préférer les noms en anglais

Les types de variables

- ✓ Typage dynamique
  - ✓ le type de la variable est déterminé par son contenu
  - ✓ Le type n'est pas fixe, il peut changer pendant l'exécution du programme

Les types scalaires

- ✓ number : un nombre entier ou décimal
- ✓ string : une suite de caractère alphanumériques
- ✓ boolean : true ou false
- ✓ undefined : le type d'une variable non encore initialisée

Les types composites

- ✓ array : un tableau indicé de valeurs
- ✓ object : un objet
- ✓ function : une fonction

Le type number


```
var entier = 5;
var decimal = 3.8;

//Notation scientifique

//54 * 10^3 = 54000
var entier = 54e3;

//54 * 10^-3 = 0.054
var entier = 54e-3;
```

```
//octal base 8
//8 en base 10
var a = 010;

//héxadécimal
//255 en base 10
var a = 0xFF;
```

Le type string


```
var nom = "Maloron";
var prenom = 'Sébastien';

//Chaîne vide
var phrase= "";

var phrase = "il m'a dit";

var code = '<p class="actif">';

var phrase = "Il m'a dit : \"vas\" ";

// Concaténation
var phrase = "Bonjour " + prenom
             + " " + nom;
```

\n	saut de ligne
\r	retour charriot
\t	tabulation
\"	guillemet
\'	apostrophe
\\	anti-slash

obtenir le type d'une variable

opérateur typeof

```
var nom = "Seb";

//retourne string
console.log(typeof nom);
```

Calcul TTC en JS


```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>tva</title>
</head>
<body>

    <script>
        // Les données en entrée
        var ht = 100;
        var taux = 20;

        // Le calcul
        var ttc = ht + (ht * taux/100);

        // La sortie
        console.log(ttc);
    </script>

</body>
</html>
```

A vous

Sur le modèle du calcul TTC

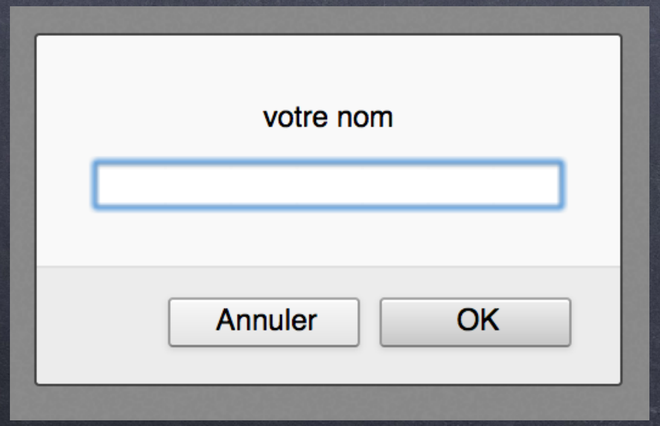
Ecrire un programme qui calcule l'aire d'un cercle

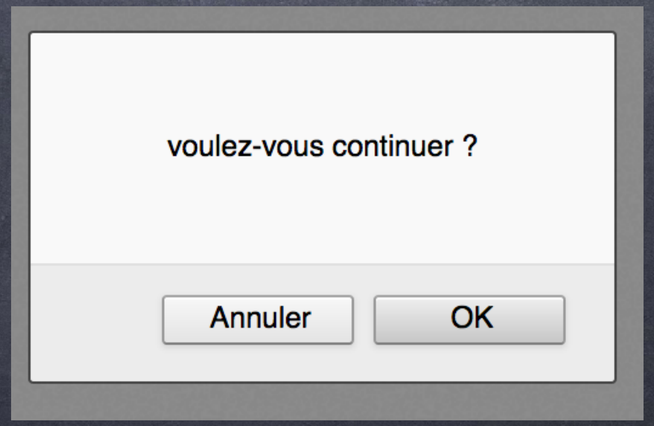
Ecrire un programme qui convertit des euros en yens

Ecrire un programme qui convertit une température depuis des degrés Celsius vers des degrés Fahrenheit

Les entrées


```
<script>
 //fenêtre de saisie
 var nom = prompt("votre nom");

 //fenêtre de confirmation
 var proceed = confirm("voulez-vous continuer ?");
</script>
```





Calcul TTC en JS


```
<script>
        // Les données en entrée
        var ht = prompt("Le montant");
        var taux = prompt("Le taux de tva");

        // Le calcul
        var ttc = ht + (ht * taux/100);

        // La sortie
        console.log(ttc);
    </script>
```

Il y a un soucis

Correction


```
<script>
        // Les données en entrée
        var ht = prompt("Le montant");
        var taux = prompt("Le taux de tva");

        // Le calcul
        var ttc = ht*1 + (ht * taux/100);

        // La sortie
        console.log(ttc);
    </script>
```

Prompt retourne un texte, hors l'addition sur des textes pose problème

En multipliant par un on force Javascript à convertir le texte en nombre

A vous

Reprendre vos précédents programmes
et ajouter des prompts

Opérateurs arithmétiques

opérateur	description	exemple pour a=5	résultat
+	addition	a + 2	7
-	soustraction	a - 2	3
/	division	a / 2	2.5
*	multiplication	a * 2	10
%	modulo reste de la division entière	a % 2	1
**	exponentiation	a**2	25

Opérateurs d'affectation

opérateur	description	exemple pour a = 5	résultat
=	affectation	a = 2	2
+=	affectation addition	a += 2	7
/=	division	a /= 2	2.5
*=	multiplication	a *= 2	10
%=	modulo reste de la division entière	a %= 2	1

Opérateurs d'affectation

oérateur	description	exemple pour a = 5	résultat
++	incrémentation	a++	6
--	décrémentation	a--	4

post-incrémentation

```
var a = 1;
/* retourne 2 car a est
incrémenté après l'évaluation de
l'expression */
console.log(a++ * 2);
```

pré-incrémentation

```
var a = 1;
/* retourne 4 car a est
incrémenté avant l'évaluation de
l'expression */
console.log(++a * 2);
```

Opérateurs de comparaison

opérateur	description	exemple pour a = 5	résultat
==	égal	a == 5	true
===	identique	a === "5"	false
>	strictement supérieur	a > 5	false
>=	supérieur ou égal	a >= 5	true
<	strictement inférieur	a < 5	false
<=	inférieur ou égal	a <= 5	true
!=	différent	a != 5	false
!==	non identique	a !== "5"	true

Opérateurs logiques

opérateur	description	exemple pour a = true et b = false	résultat
&&	ET logique	a && b	false
||	OU logique	a || b	true
!	Négation logique	! (a && b)	true

A vous

Demander une taille en mètres
et la convertir en pieds et en pouces
1 pied = 30,48 cm, 1 pouce = 2,54 cm

Demander une capacité en Giga octet
et la convertir en bits
1 octet = 8 bits, 1 ko = 1024 octets,
1 Mo = 1024 Ko, 1 Go = 1024 Mo,

Convertir un angle de degrés en radians

Constantes


```
const PI = 3.14116;
var radius = prompt("Le rayon");

console.log(PI * radius**2);
```

- ✓ Une constante est une données que l'on ne peut pas changer
- ✓ Comme la taille de la constante est fixe, Javascript peut stocker cette information dans un espace mémoire plus efficace que pour les variables
- ✓ Les constantes sont souvent notées en majuscule
- ✓ Quand il n'y a aucune raison pour que l'information change au cours du programme on préférera utiliser des constantes

Les alternatives

Condition


```
if(condition){
 instructions;
}
```

La condition est une expression booléenne

```
if(condition){
 //condition == true
 instructions;
} else {
 //condition == false
 instructions
}
```

La clause else est invoquée lorsque la condition est fausse

Exemple


```
<script>
        var age = prompt("Entrer votre âge");

        if(age < 18){
            console.log("mineur");
        } else {
            console.log("majeur");
        }
</script>
```

Validation des saisies


```
<script>
        var age = prompt("Entrer votre âge");

        // affiche NaN (Not a Number)
        // quand on entre du texte
        console.log(age *1)

        if(isNaN(age * 1)){
            console.log("Saisie incorrecte")
        }else if(age < 18){
            console.log("mineur");
        } else {
            console.log("majeur");
        }
    </script>
```

Validation des saisies (autre méthode)


```
<script>
        var age = prompt("Entrer votre âge");

        // Conversion de l'âge en entier
        // si impossible retourne null
        age = parseInt(age);

        if(age == null){
            console.log("Saisie incorrecte")
        }else if(age < 18){
            console.log("mineur");
        } else {
            console.log("majeur");
        }
    </script>
```

Switch


```
<script>
        var note = parseInt(prompt("Votre note"));

        switch (note) {
            case 1:
                console.log("très mauvais");
                break;
            case 2:
                console.log("mauvais");
                break;
            case 3:
                console.log("moyen");
                break;
            case 4:
                console.log("bon");
                break;
            case 5:
                console.log("très bon");
                break;
            default:
                console.log("saisir une note entre 1 et 5")
        }
    </script>
```

A vous

Reprendre le programme TTC
et ajouter la validation des saisies

Conditions multiples


```
<script>
        var login = prompt("Entrer votre identifiant");
        var pass = prompt("Entrez votre mot de passe");

        if(login == "admin" && pass == "123"){
            console.log("Accès autorisé");
        } else {
            console.log("Accès interdit")
        }
    </script>
```

Gérer le cas où la saisie est vide

Correction


```
<script>
        var login = prompt("Entrer votre identifiant");
        var pass = prompt("Entrez votre mot de passe");

        if(login == "" || pass == ""){
            console.log("Saisie incomplète");
        }else if(login == "admin" && pass == "123"){
            console.log("Accès autorisé");
        } else {
            console.log("Accès interdit")
        }
    </script>
```

Evaluation des combinaisons

Dans une expression booléenne multiple
avec l'opérateur &&
si la première assertion est fausse
les autres ne sont pas évaluées

Faux ET Vrai = Faux

Faux ET Faux = Faux

Évaluation à false


```
<script>
    var age = parseInt(prompt("Votre age"));

    // test si age n'est pas évalué comme false
    // évaluation implicite
    if (age) {
        console.log("valide");
    }

    // age n’est pas un booléen
    // l'évaluation explicite échoue
    if (age == true) {
        console.log("valide");
    }
</script>
```

valeurs évaluées comme false
0
chaîne vide
undefined
null

A vous

Valider la saisie d'un âge entre 7 et 77 ans

Demander une quantité de produit commandé
(le prix unitaire étant de 12 € HT).
Afficher le total TTC en appliquant une remise
de 5% si le total HT est supérieur à 400
et 10% s'il est supérieur à 1200.
Note : la remise s'applique sur le montant HT

Demander une année et afficher si elle est bissextile

A vous

Décomposer un nombre de secondes en unités supérieures.

Par exemple :

70 secondes donneront

1:10

une minute et dix secondes,

4840 donneront

1:20:40

une heure, vingt minutes et quarante secondes.

A vous

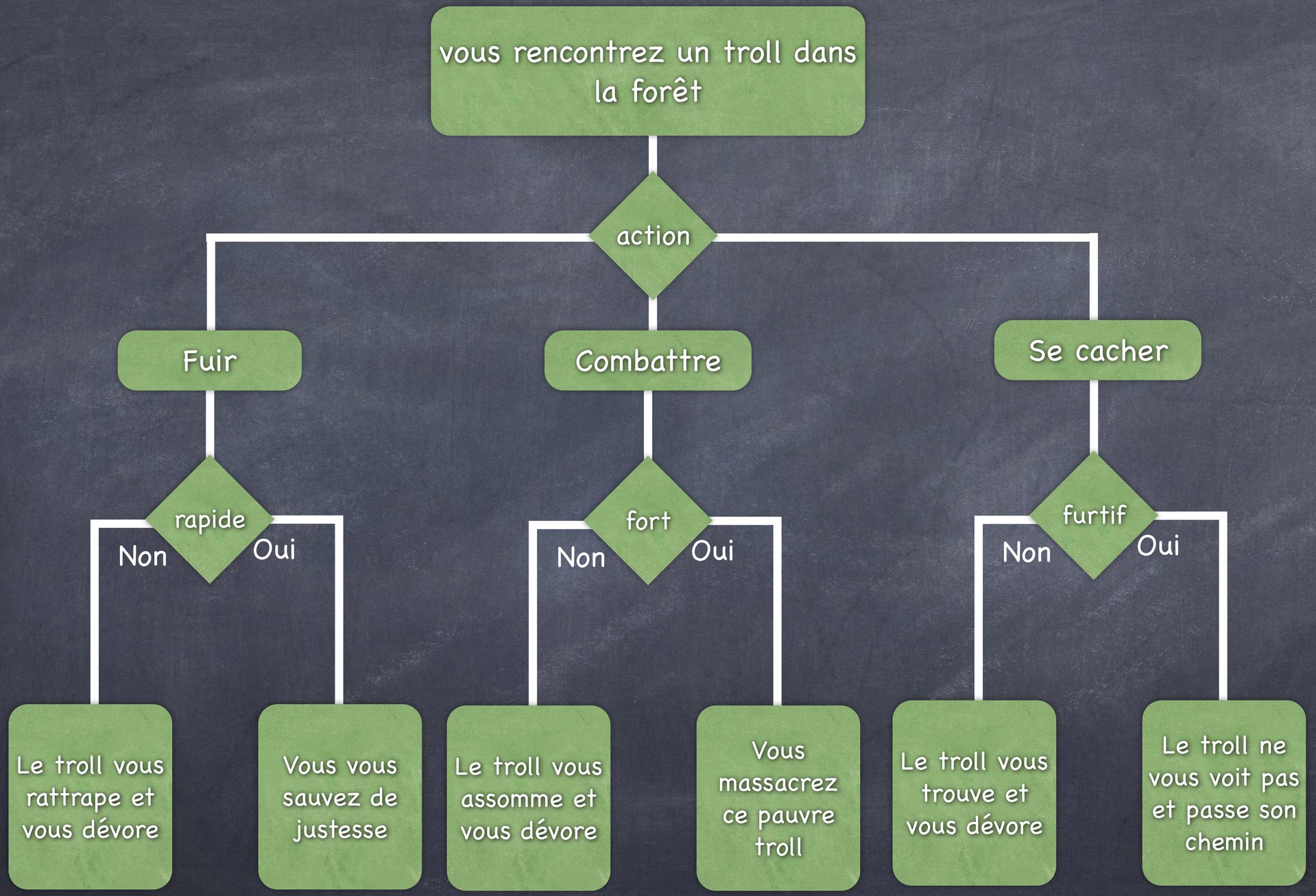
Implémenter les test de fizzbuzz sur un nombre saisi

- ✓ S'il est multiple de 3 afficher fizz
- ✓ S'il est multiple de 5 afficher buzz
- ✓ S'il est multiple de 3 et 5 afficher fizzbuzz
- ✓ Sinon afficher le nombre

A vous

Traduire un age en une tranche d'âge

- ✓Bébé (0-5)
- ✓Enfant (6-13)
- ✓Ado (14-17)
- ✓Adulte (18-65)
- ✓Sénior (66-120)
- ✓Tortue (121-250)
- ✓Séquoia (au dela)


vous rencontrez un troll dans la forêt
action
Fuir
Combattre
Se cacher
rapide
Non
Oui
fort
Non
Oui
furtif
Non
Oui
Le troll vous rattrape et vous dévore
Vous vous sauvez de justesse
Le troll vous assomme et vous dévore
Vous massacrez ce pauvre troll
Le troll vous trouve et vous dévore
Le troll ne vous voit pas et passe son chemin

Les boucles

Ecriture dans la page

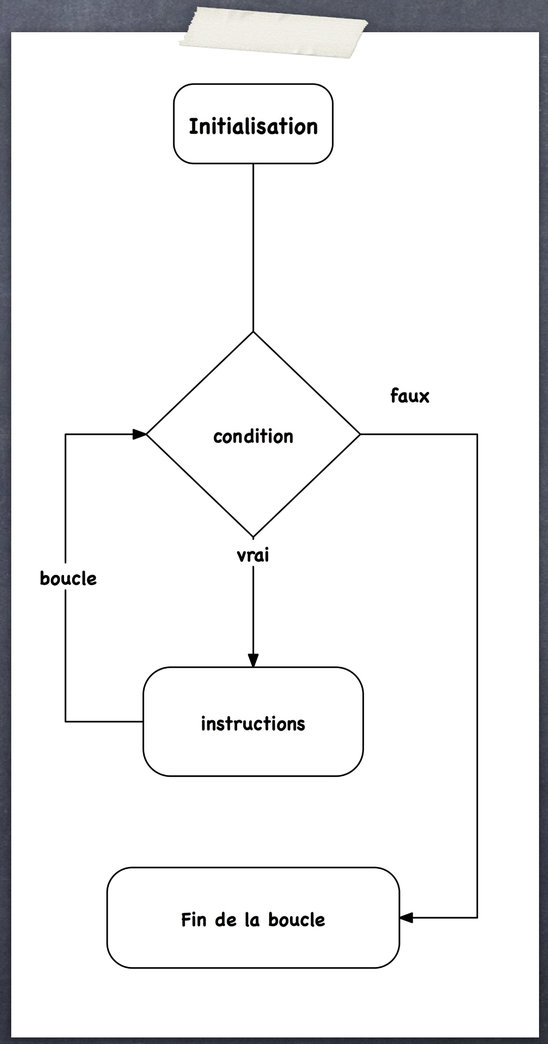

```html
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
</head>

<body>

    <div id="target"></div>

    <script>
        // Ciblage d'un élément de la page html
        var output = document.getElementById("target");

        // Ecriture dans un élément cible
        output.innerHTML = "<h1>Hello</h1>";
    </script>
</body>

</html>
```

Principe

- ✓ Exécuter plusieurs fois une suite d'instructions
  - ✓ Tant que (quelque chose est vrai)
  - ✓ Répéter (n fois)

Tant que

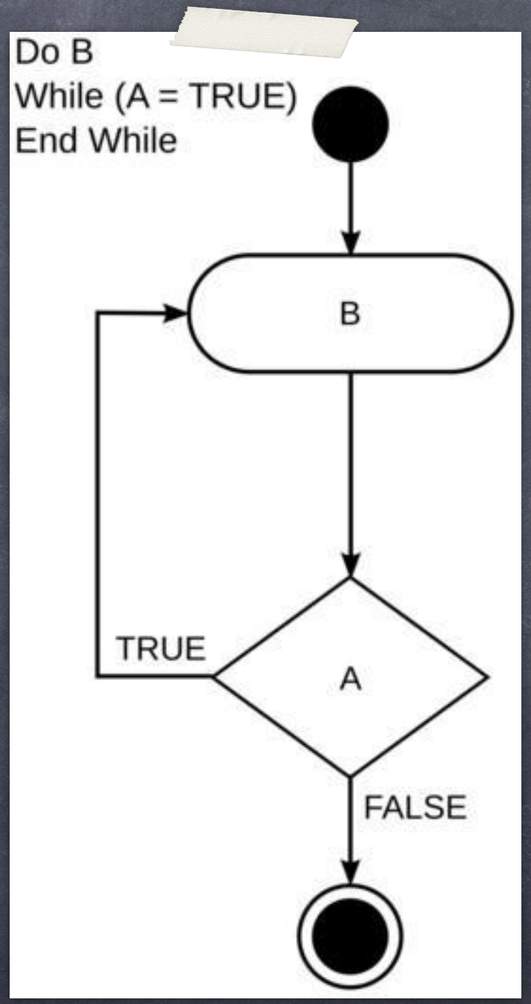

```
while(condition){
 instructions;
}
```



Jusqu'à ce que


```
do {
 instructions;
} while(condition);
```

Les instructions sont exécutées au moins une fois



Sortie d'une boucle


```
var iterations = 0;

while(true) {

 iterations ++;

 if(iterations > 10){
  Break;
 }

}
```

Que fait ce code ?


```
<script>
        var sum = 0;
        var max = 10;
        while(sum < max){
            var n = prompt("Un nombre");
            sum = sum + parseInt(n);
        }

        console.log(sum);
    </script>
```

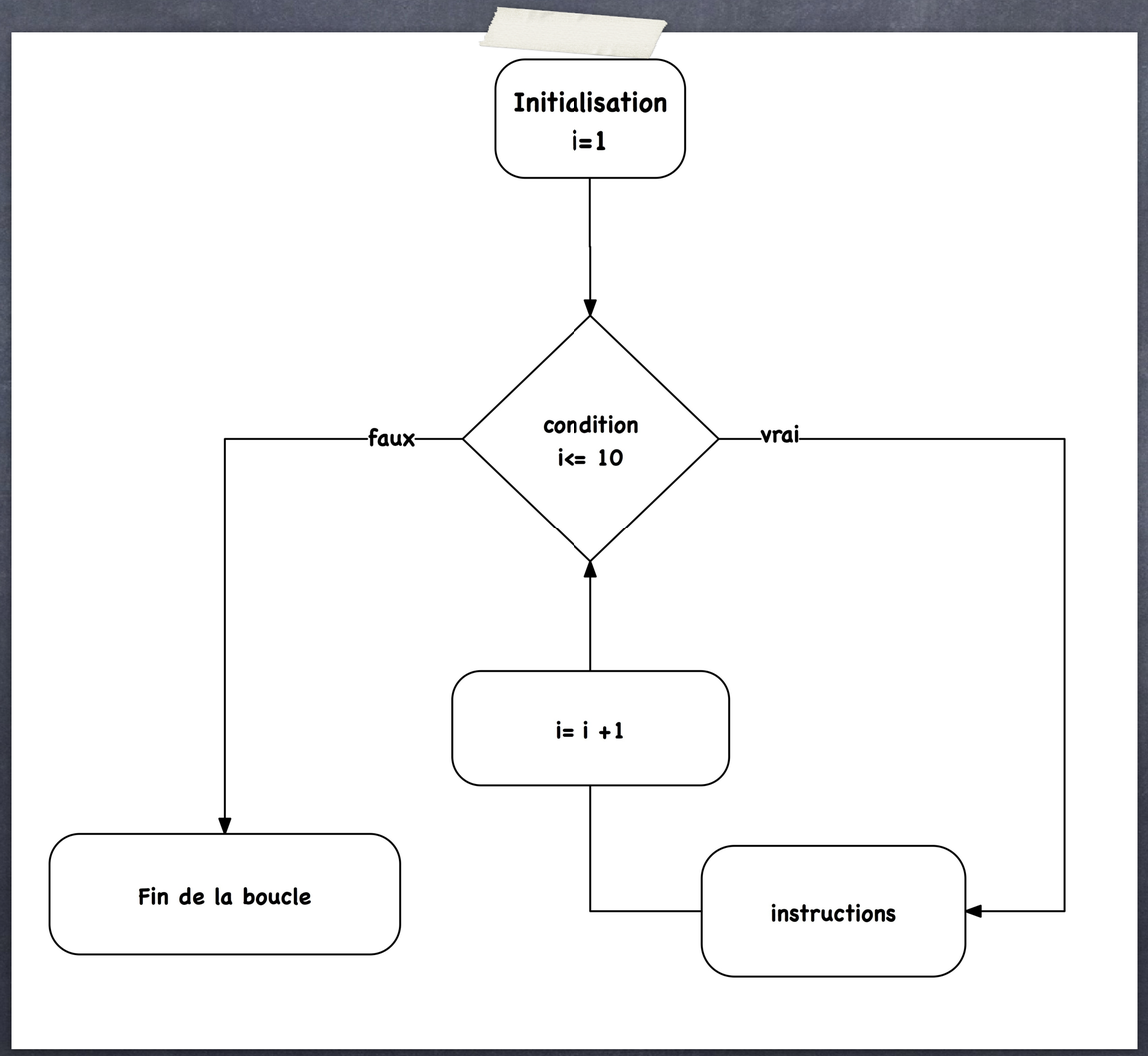
Ecrire un code qui fait la même chose avec un break

Que fait ce code ?


```
<script>
    var n = "";
    while(isNaN(n) || n <0){
        n = prompt("Un nombre positif");
        n = parseInt(n);
    }

    console.log(n);
</script>
```

Ecrire un code qui fait la même chose
avec une condition en fin de boucle
do while au lieu de while

Avec compteur



Syntaxe


```
for(initialisation; condition; expression){
  instructions;
}
```

```
for(var i = 1; i <=10; i++){
  instructions;
}
```

```
for(var i = 10; i >0; i-=2){
  instructions;
}
```

Exemple


```
<script>
   for(var i=1; i <= 5; i++){
     console.log("bonjour");
   }
</script>
```

Affiche bonjour 5 fois

Affichage dans le navigateur


```html
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
</head>

<body>

    <div id="target"></div>

    <script>
        var output = document.getElementById("target");
        var html = "";

        for(var i=1; i <= 5; i++){
            html += "<h1>Bonjour</h1>";
        }

        output.innerHTML = html;
    </script>
</body>

</html>
```

A vous

Reprendre le code précédent
et afficher bonjour dans des titres
d'importance décroissante
h1, h2, h3, h4, h5

Faire la même chose
mais avec des titres
d'importance croissante

Correction décroissant


```
<script>
        var output = document.getElementById("target");
        var html = "";

        for(var i=1; i <= 5; i++){
            html += "<h" + i + ">Bonjour</h" + i + ">";
        }

        output.innerHTML = html;
    </script>
```

Correction croissant


```
<script>
        var output = document.getElementById("target");
        var html = "";

        for(var i=5; i <= 1; i--){
            html += "<h" + i + ">Bonjour</h" + i + ">";
        }

        output.innerHTML = html;
    </script>
```

Table de multiplication


```
<script>
    var output = document.getElementById("target");

    var html = "<ul>";

    var n = prompt("Entrez un nombre");
    n = parseInt(n);

    for(var i=1; i <= 9; i++){
        html += "<li>" + n + " fois "
             + i + " : " + (i*n) + "</li>" ;
    }

    html += "</ul>";
    output.innerHTML = html;
</script>
```

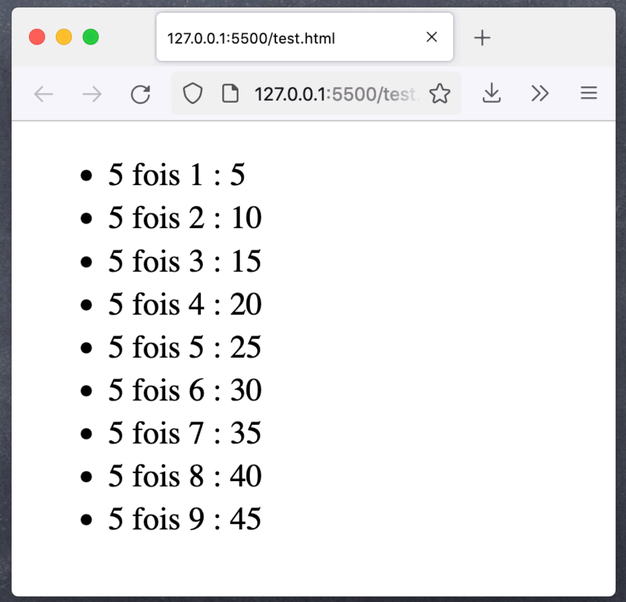
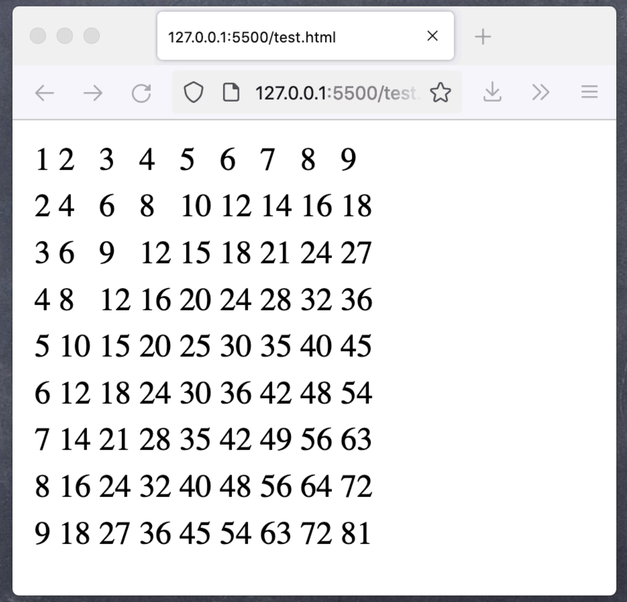


Table de multiplication 2


```
<script>
    var output = document.getElementById("target");
    var html = "<table>";

    for(var i=1; i <= 9; i++){
        html += "<tr>";
        for(var k=1; k <=9; k++){
            html += "<td>" + (i*k) + "</td>";
        }
        html += "</tr>";
    }
    html += "</table>";

    output.innerHTML = html;
</script>
```



Application

Écrire un programme qui affiche les formes suivantes dans la console

```
**********
```

```
*****
*****
*****
*****
```

```
*
**
***
****
*****
```

Application

Écrire un programme qui demande la saisie d'un entier dans une boucle.

Si la saisie n'est pas un entier afficher une boite de dialogue demandant à l'utilisateur s'il veut quitter l'application.

Si tel est le cas afficher la somme des entiers saisis sinon demander à nouveau la saisie d'un entier.

Application

Écrire un programme qui génère un nombre aléatoire entre 1 et 100 puis qui demande à l'utilisateur de saisir un nombre et ce tant que le nombre aléatoire n'a pas été trouvé. A chaque tentative, le programme indiquera si le nombre saisi et plus petit ou plus grand que le nombre à deviner.

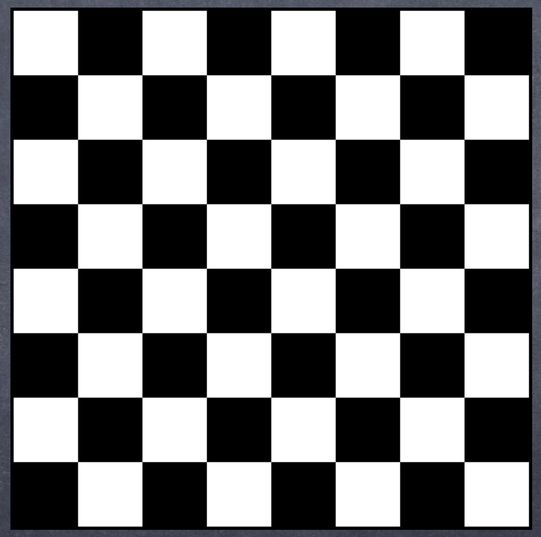
A la fin du jeu, le programme affichera le nombre de tentatives qui ont été nécessaires pour gagner.

(Math.random() retourne un aléatoire entre 0 et 1)

```
Math.floor(Math.random() * (max - min + 1)) + min
```

Application

Écrire un programme qui génère un échiquier sous la forme d'un tableau html. Un échiquier contient 8 ligne et 8 colonnes,
les cases alternent entre noire et blanche



Chaîne de caractères

Principes

Un texte ou chaîne de caractère est une liste de caractères

caractères	h	e	l	l	o
indices	0	1	2	3	4

Quelques méthodes

Fonction	Description
`str.length`	nombre de caractères dans la chaîne str
`str.charAt(pos)`	retourne le caractère à l'index pos. Sur les navigateurs récents on peut également utiliser str[pos]
`str.charCodeAt(pos)`	retourne le code du caractère à l'index pos
`String.fromCharCode(code)`	retourne le caractère correspondant au code passé en argument

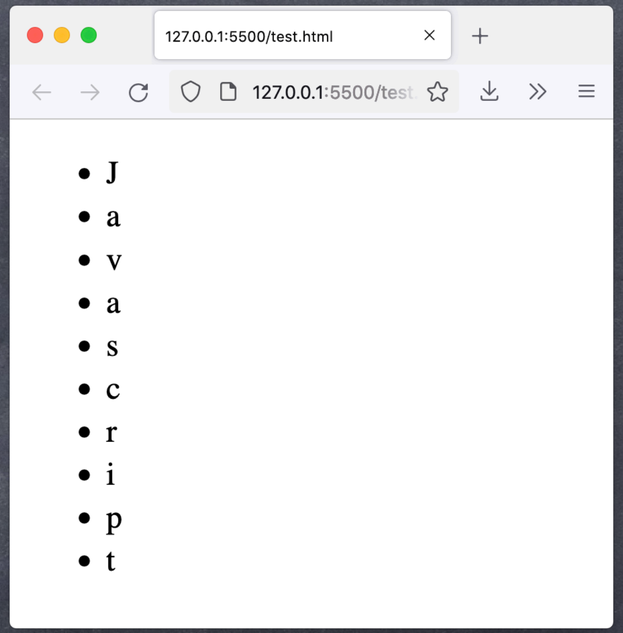
La table des caractères

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Boucler sur une chaîne


```
<script>
        var output = document.getElementById("target");

        var html = "<ul>";

        var txt = prompt("Entrez un texte");

        for (var i = 0; i < txt.length; i++) {
            html += "<li>" + txt.charAt(i) + "</li>";
        }

        html += "</ul>";
        output.innerHTML = html;
</script>
```



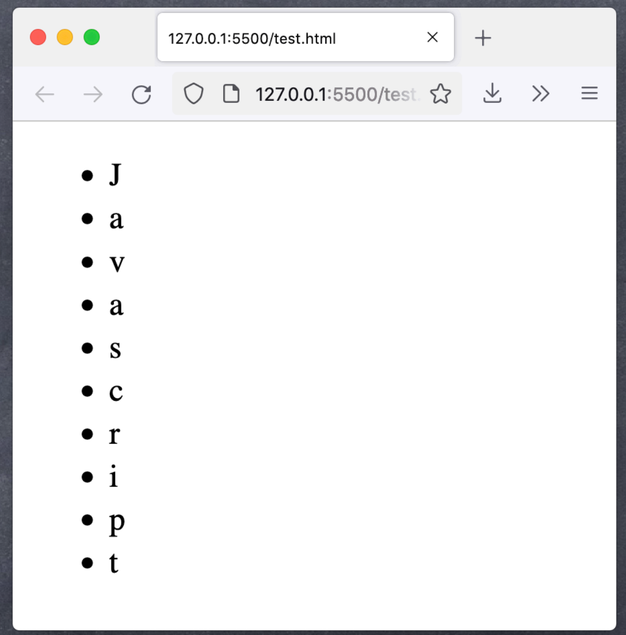
Essayez d'inverser la boucle

Inverser une chaîne


```
<script>
        var txt = prompt("Entrez un texte");

        var inverse = "";

        for (var i = txt.length-1; i >=0 ; i--) {
            inverse += txt.charAt(i);
        }

        console.log(inverse);
</script>
```

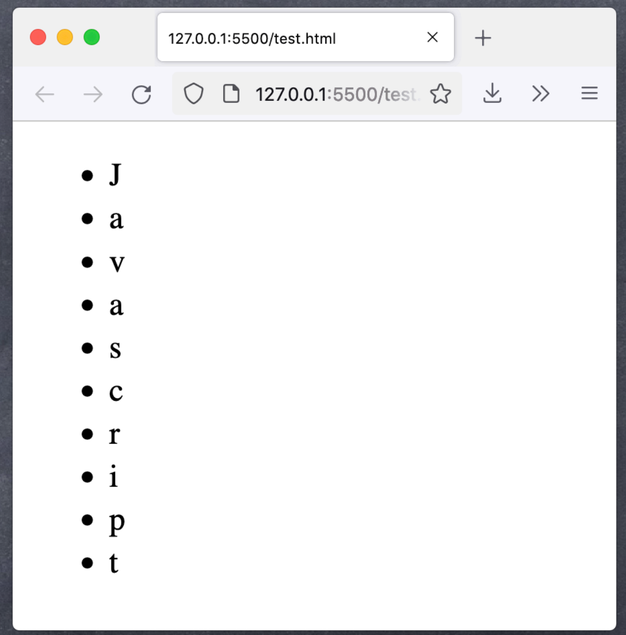
Boucler sur une chaîne avec des indices


```
<script>
        var output = document.getElementById("target");

        var html = "<ul>";

        var txt = prompt("Entrez un texte");

        for (var i in txt) {
            html += "<li>" + txt.charAt(i) + "</li>";
        }

        html += "</ul>";
        output.innerHTML = html;
</script>
```



Boucler sur une chaîne lettre par lettre


```
<script>
        var output = document.getElementById("target");

        var html = "<ul>";

        var txt = prompt("Entrez un texte");

        for (var letter of txt) {
            html += "<li>" + letter + "</li>";
        }

        html += "</ul>";
        output.innerHTML = html;
</script>
```



Définir si un texte comporte une majuscule


```
<script>
        var txt = prompt("Entrez un texte");

        var found = false;

        for (var i in txt) {
            code = txt.charCodeAt(i)
            if( code >= 65 && code <= 90){
                found = true;
                // On a trouvé une capitale
                // inutile de continuer la boucle
                break;
            }
        }

        console.log(found);
</script>
```

A vous

En vous inspirant du programme précédent
écrire un programme qui :

Trouve l'index de la première lettre majuscule d'un texte
Détermine si un texte est un nombre (ne contient que des chiffres)
Détermine si un texte ressemble à une adresse email

Passer la première lettre en majuscule


```
<script>
        var txt = prompt("Entrez un texte");
        var str = "";

        var code = txt.charCodeAt(i)

        if(code >= 97 && code <= 122){
            code -= 32;
            str += String.fromCharCode(code);
        }

        for (var i in txt) {
            str += txt.charAt(i);
        }

        console.log(str);
</script>
```

A vous

Ecrire un programme qui passe tout un texte en majuscule

Ecrire un programme qui passe en majuscule
la première lettre de chaque mot

A vous

Ecrire un programme qui passe convertit du snake case au camel case

snake_case
camelCase

Ecrire un programme qui fait l'inverse

A vous

Ecrire un programme qui élimine les espaces aux extrémités d'une chaîne

Ecrire un programme qui compte le nombre de mots dans une phrase

A vous

Ecrire un programme qui valide un mot de passe

Au moins un chiffre

Au moins une majuscule

Au moins une minuscule

Au moins 8 caractères

A vous

Ecrire un programme qui chiffre un texte
en décalant chaque lettre d'une position
a devient b, c devient d, les espaces sont conservés

fonctions

Définition

- ✓ Regroupe une série d'instructions sous une dénomination
- ✓ Ces instructions peuvent être exécutées en invoquant la fonction
- ✓ Il est possible d'invoquer plusieurs fois une fonction
- ✓ Un fonction peut elle même invoquer d'autres fonctions

Utilité

- ✓ Organiser et simplifier le code
  - ✓ décomposer un problème global en unités simples
- ✓ Factoriser, éviter la duplication du code
- ✓ Auto-documenter le code, améliorer la lisibilité
- ✓ Améliorer la fiabilité du code
- ✓ Améliorer notre productivité en tant que développeur

Création


```
// Création de la fonction
function hello(){
    console.log("Hello");
}
// appel de la fonction
hello();
```

Une fonction est un type de données et peut donc être stockée dans une variable ou une constante

```
// Création de la fonction
const hello = function(){
    console.log("Hello");
}
// appel de la fonction
hello();
```

Argument

Les arguments rendent la fonctions plus générique

```
// Création de la fonction
// avec un argument
function hello(name){
    console.log("Hello " + name);
}
// appel de la fonction
hello("world");
```

Valeur de retour


```
function hello(name){
    return "Hello " + name;
}
// appel de la fonction
// la valeur de retour
// est stockée dans une variable
var msg = hello("world");
console.log(msg);
```

Valeur par défaut


```
function hello(name = "Seb"){
    return "Hello " + name;
}
// appel de la fonction
var msg = hello();
console.log(msg);
```

Une fonction de calcul


```
function ttc(ht, tva = 20){
    return ht * (1 + (tva/100));
}
// appel de la fonction
console.log(ttc(180));

console.log(ttc(180, 8));
```

A vous

Reprendre le programme
de conversion de Celsius à Fahrenheit
et en faire une fonction

Reprendre le programme
de validation de mot de passe
et en faire une fonction

Découpage d'un problème en fonctions


```
function getCell(i,k){
    return "<td>" + (i*k) + "</td>";
}
function getRow(i){
    var html = "<tr>";
    for (var k = 1; k <= 9; k++) {
        html += getCell(i,k);
    }
    html += "</tr>";
    return html;
}
function getTable(){
    var html = "<table>";
    for (var i = 1; i <= 9; i++) {
        html += getRow(i);
    }
    html += "</table>";
    return html
}
output = document.getElementById("target");
output.innerHTML = getTable();
```

Chaque fonction ne possède qu'une responsabilité et ne résout qu'une partie du problème

A vous

Ecrire une fonction qui retourne le code html d'une liste déroulante contenant des années.
Les arguments de cette fonction seront l'année de départ et le nombre d'années à inclure dans la liste

Portée des variables


```
var global = 5;

function calc(n){
    return global + n;
}

console.log(calc(6));
```

Ici la variable "global" est accessible pour la fonction même si elle est initialisée en dehors

Une variable déclarée dans le corps d'une fonction n'existe donc que pour cette fonction.

Une fois l'appel de la fonction terminée la variable est détruite

```
var global = 5;

function calc(n){
    // variable locale à la fonction
    var local = 8;
    return global + n;
}

console.log(calc(6));
// undefined, local n'existe pas
// en dehors de la fonction
console.log(local);
```

Variables globales


```
var divisor = 2;

function calc(n){
    return n/divisor;
}

function changeDivisor(){
    divisor --;
}
```

En quoi ce code pose-t-il un problème ?

Fonction de callback


```
function showInConsole(content){
    console.log(content);
}

function add(n1, n2, callback){
    var sum = n1 + n2;
    callback(sum);
}

// Notez qu'il n'y a pas de double parenthèse
add(4,3, showInConsole);

// Callback avec une fonction anonyme
add(5,3, function(sum){
    console.log(sum)
});
```

A quoi peut servir un callback ?

Récursivité


```
// compte à rebours
function countDown(n) {
    // affichage du nombre
    console.log(n);
    // décrémentation
    const newNumber = n - 1;
    // appel récursif
    if (newNumber > 0) {
        countDown(newNumber);
    } else {
        console.log("ignition");
    }
}
```

Une fonction qui s'appelle elle même

Factorielle


```
function factorial(x) {

    if (x === 0) {
        return 1;
    } else {
        return x * factorial(x - 1);
    }
}

console.log(factorial(5));
```

itération	retour
f(3)	3 * f(2)
f(2)	3 * 2 * f(1)
f(1)	3 * 2 * 1 * f(0)
f(0)	3 * 2 * 1 * 1

Tableaux

Présentation

indice	valeur
0	pommes
1	poires
2	oranges
3	coings
4	groseilles

- Liste de données indexées par une clef numérique
- Premier indice 0

Déclaration


```
const fruits = new Array(
"pommes", "poires"
);
```

ou

```
const fruits = ["pommes", "poires"];
```

un tableau peut contenir
des données de types différents
y compris des types composites
(tableaux, objets, fonctions)

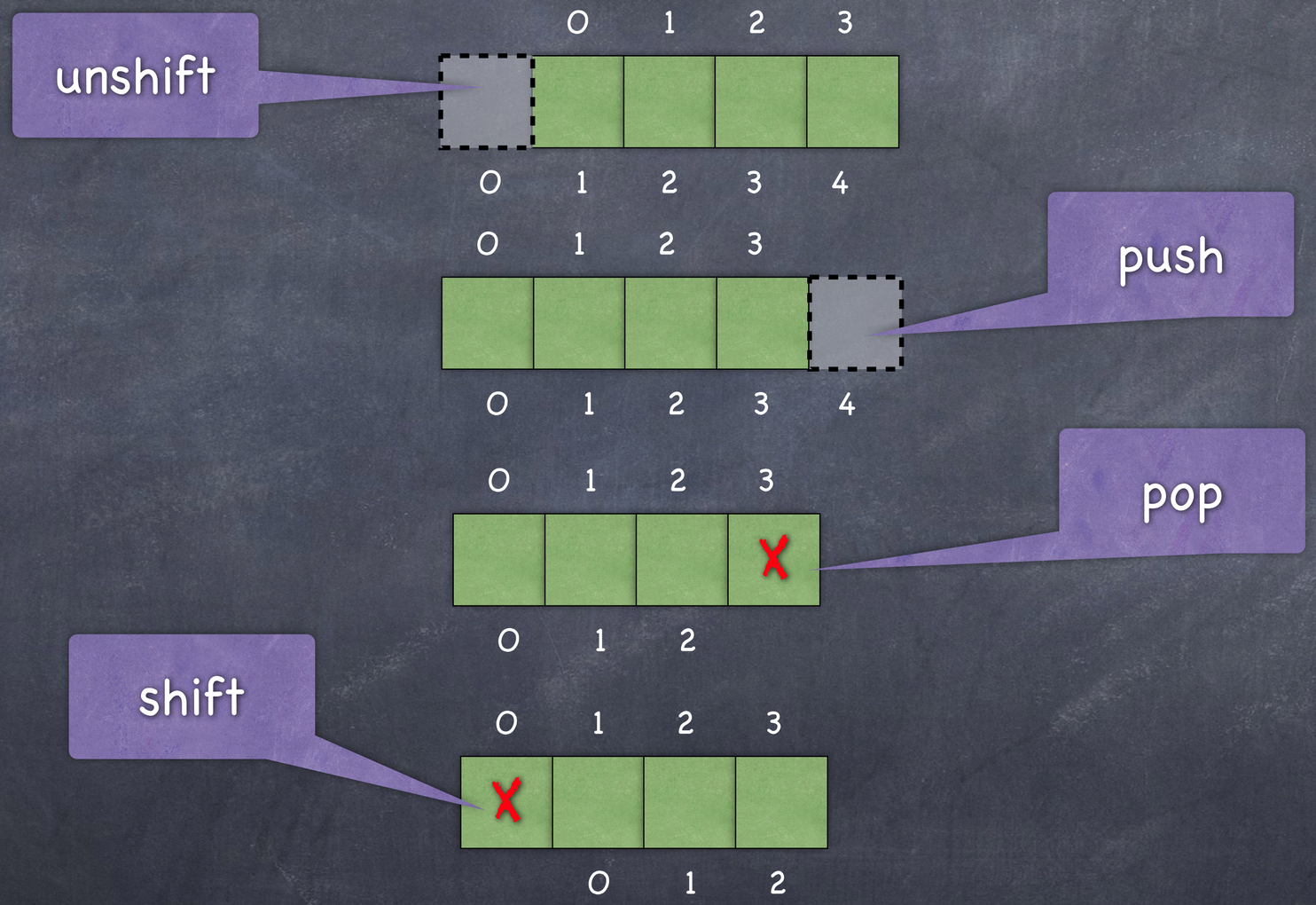
Accès aux éléments


```
const fruits = ["pommes", "poires","oranges"];

//Affiche le deuxième élément
console.log(fruits[1]);

//Modifie la valeur du troisième élément
fruits[2] = "figues";

//Ajoute un élément
fruits[3] = "mangues";
```

Opérations sur les tableaux

opération	code
ajouter un élément à la fin d'un tableau	`array.push(valeur)`
ajouter un élément au début du tableau	`array.unshift(valeur)`
obtenir le nombre d'éléments d'un tableau	`array.length`
retourner et supprimer le dernier élément	`el = array.pop()`
retourner et supprimer le premier élément	`el = array.shift()`

Récap


unshift
0 1 2 3
0 1 2 3 4
push
0 1 2 3
0 1 2 3 4
pop
0 1 2 3
0 1 2
shift
0 1 2 3
0 1 2

Parcourir un tableau


```
const fruits = ["pommes", "poires", "oranges"];

var fruitsSize = fruits.length;

for (var i = 0; i < fruitsSize; i++) {

    console.log(fruits[i]);

}
```

Parcourir un tableau (autre syntaxe)


```
var fruits = ["pommes", "poires", "oranges"];

for (var index in fruits) {

    console.log(fruits[index]);

}
```

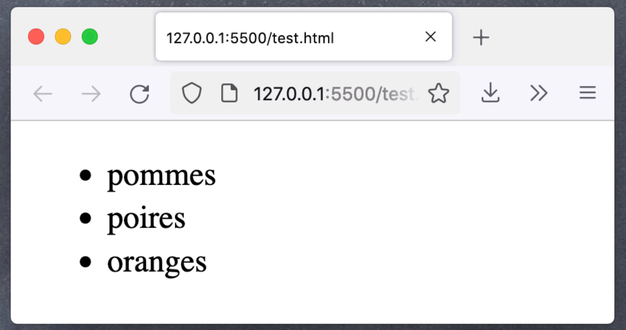
Parcourir un tableau (autre syntaxe)

Ici on récupère les valeurs et non les indices

```
var fruits = ["pommes", "poires", "oranges"];

for (var item of fruits) {

    console.log(item);

}
```

Afficher les données d'un tableau


```
const out = document.getElementById("target");

var fruits = ["pommes", "poires", "oranges"];

var html = "<ul>";

for (var item of fruits) {
    html += "<li>" + item + "</li>";
}

html += "</ul>";

out.innerHTML = html;
```



Afficher sous forme de liste déroulante

Copier un tableau


```
const names = ['Pauline', 'Sylvie', 'Marcel'];
// Simple référence
const users = names;
// on change users
users[0] = 'Antoine';
console.log(names);
// users est en fait une référence à names

// Copie profonde
const users2 = Array.from(names);
// ou bien
const users3 = [...names];
```

L'opérateur ... prends toutes les valeur d'un tableau ordinal et les passe en argument

A vous

Définir un tableau de nombres

calculer la moyenne

trouver l'indice et la valeur du nombre le plus grand

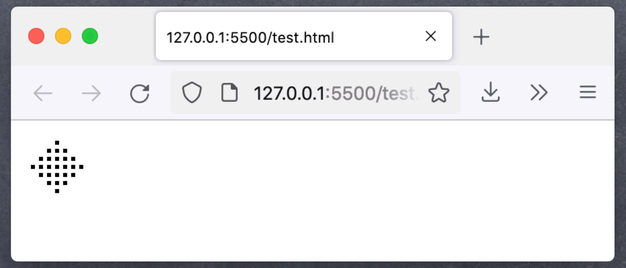
A vous

Ecrire une fonction qui retourne vrai si une valeur passée en argument existe dans un tableau

Si on présuppose que le tableau est trié peut-on adopter une méthode plus efficace ?

Double boucle


```
<script>
    const out = document.getElementById("target");

    const matrix = [
        [0, 0, 0, 1, 0, 0, 0],
        [0, 0, 1, 1, 1, 0, 0],
        [0, 1, 1, 1, 1, 1, 0],
        [1, 1, 1, 1, 1, 1, 1],
        [0, 1, 1, 1, 1, 1, 0],
        [0, 0, 1, 1, 1, 0, 0],
        [0, 0, 0, 1, 0, 0, 0],
    ];

    var html = "<table>";

    for (var row of matrix) {
        html += "<tr>";
            for(var col of row){
                if(col == 0){
                    var color = "white";
                } else {
                    color = "black";
                }

                html += '<td style="background-color:'+ color +'"></td>'
            }
        html += "</tr>";
    }

    html += "</table>";

    out.innerHTML = html;
</script>
```



Créer une matrice aléatoire et afficher la

A vous

Ecrire un programme
qui recherche la position de la
première occurrence d'une valeur
au sein d'un tableau

Ecrire un programme
qui permute les valeurs de deux indices

A vous

Ecrire un programme qui insère un élément à une position donnée dans un tableau

Ecrire un programme qui supprime un élément à une position donnée dans un tableau

A vous


```
const fruits = [
        "pommes", "oranges",
        "poires", "oranges",
        "pommes", "figues",
        "poires", "pommes"
];
```

Ecrire un programme qui compte le nombre d'occurrences de "pommes"

Ecrire un programme qui retourne vrai s'il y a des doublons

Ecrire un programme qui retourne un tableau sans doublons

Dictionnaires

Présentation

clef	valeur
nom	Maloron
prenom	Sébastien
age	45

Liste de données indexées par une clef alphanumérique unique

Déclaration


```
const person = {
    nom: 'Sébastien',
    age: 54
};
```

```
const Person = new Object();
Person.nom = "Sébastien";
Person.age = 54;
```

```
console.log(person.nom);
console.log(person['age']);
```

Boucle sur un dictionnaire


```
const person = {
    nom: 'Maloron',
    prenom: 'Sébastien',
    age: 45
};
for (let key in person){
    console.log(key + " : " + person[key]);
}
```

Les clefs sont dynamiques


```
var person = {};
//Le fait d'affecter une valeur
//génère automatiquement une clef
person.nom = 'Maloron';
person.prenom = 'Sébastien';
console.log(person);
```

Tester l'existence d'une clef


```
var person = {};
//avec l'opérateur in
if ('age' in person) {
    console.log('la clef age existe');
}
//méthode alternative
if (person.hasOwnProperty('nom')) {
    console.log('la clef nom existe');
}
```

A quoi sert un dictionnaire

- ✓ Rassembler les propriétés d'une même chose, par exemple une personne possède un nom, un prénom et un âge
- ✓ Créer une table de correspondance ce qui peut remplacer avantageusement un switch

Occurrences des lettres dans une phrase


```
const sentence = prompt("une phrase");
const letters = {};

for(let char of sentence){
    console.log(char);
    if(! char in letters){
        letters[char] = 1;
    } else {
        letters[char] += 1;
    }
}

console.log(letters);
```

Calculer le pourcentage d'apparition de chaque lettre dans un autre dictionnaire

Table de correspondance


```
const city = prompt("votre ville");
const countriesByCity = {
    Paris: "France",
    Londres: "Royaume Uni",
    Washington: "USA",
    Dublin: "Irlande"
};

if( city in countriesByCity){
    console.log(countriesByCity[city]);
} else {
    console.log("Votre ville n'est pas référencée");
}
```

Notation des clefs


```
const city = prompt("votre ville");
const countriesByCity = {
    Paris: "France",
    Londres: "Royaume Uni",
    Washington: "USA",
    Dublin: "Irlande",
    "Gif sur Yvette": "France"
};
```

Si la clef comporte des espaces ou des accents elle doit être notée entre guillemets

Dictionnaire complexe


```
const person = {

    nom: 'Maloron',
    prenom: 'Sébastien',
    age: 45,

    skills: ['Java', 'Python', 'PHP'],

    address: {
        street: '5 rue Orfila',
        zipCode: '75020',
        city: 'Paris'
    }
};
```

Tableau ordinal de dictionnaires


```
const persons =
[
    { nom: 'Maloron', prenom: 'Sébastien' },
    { nom: 'Maloron', prenom: 'Vincent' },
    { nom: 'Maloron', prenom: 'Clémence' },
];
```

Afficher ces données dans une table html

Créer une fonction générique qui affiche n'importe quel dictionnaire dans une table html

Une structure de données


```
const books = [
    { title: 'Apprendre SQL avec MySQL', authors: ['Christian Soutou'], tags: ['Informatique', 'Base de données'], price: 25  },
    { title: 'Best practices PHP 5', authors: ['Guillaume Ponçon'], tags: ['Informatique', 'PHP'], price: 30  },
    { title: 'A Guide to SQL', authors: ['Philip Pratt', 'Mary Last'], tags: ['Informatique', 'Base de données'], price: 18 },
    { title: 'PHP/MySQL avec Dreamweaver 8', authors: ['Jean-Marie Defrance'], tags: ['Informatique', 'PHP', 'Base de données'], price: 25 },
    { title: 'Ajax, jQuery et PHP', authors: ['Jean-Marie Defrance'], tags: ['Informatique', 'PHP', 'jQuery', 'Ajax', 'Javascript'], price: 20 },
    { title: 'Le Colonel Chabert', authors: ['Honoré de Balzac'], tags: ['Roman', 'classique'], price: 10 },
    { title: 'JavaScript pour le Web 2.0', authors: ['Arnaud Gougeon', 'Thierry Templier'], tags: ['Informatique', 'Javascript'], price: 26 },
    { title: 'Botanique systématique des plantes à fleurs', authors: ['Daniel Jeanmonod', 'Murielle Figeat'], tags: ['Science', 'botanique'], price: 18 },
    { title: 'La Société des affects', authors: ['Frédéric Lordon'], tags: ['Science sociales'], price: 20 },
    { title: 'Les Quadratures de la politique économique', authors: ['Frédéric Lordon'], tags: ['Science politique'], price: 20 },
    { title: 'Eloge de l\'amour', authors: ['Nicolas Truong', 'Alain Badiou'], tags: ['Philosophie'], price: 10 },
    { title: 'Diagnostic de la performance de l\'entreprise', authors: ['Christophe Everaere', 'Alain Asquin', 'Alain Marion'], tags: ['Economie', 'Gestion'], price: 20 },
    { title: '100 fiches pour comprendre les sciences économiques', authors: ['Dominique Chamblay', 'Marc Montoussé'], tags: ['Economie'], price: 18 },
 ];
```

A vous

Écrire un programme qui retourne :

- tous les livres dont le prix est inférieur à 20
- tous les livres possédant le tag : Base de données
- tous les livres qui ont plus d'un auteur
- la somme du prix des livres parlant d'informatique

A vous

Écrire un programme qui :

- supprime tous les livres de Jean-Marie Defrance
- ajoute un livre à la fin du tableau
- ajoute un livre au début du tableau
- augmente le prix des livres d'informatique de 10%

Quelques astuces

Une fonction dans un objet


```
const sophie = {
    firstName: "Sophie",
    lastName: "Martin",
    age: 34,
    getFullName: function(){
        return `${this.firstName} ${this.lastName.toUpperCase()}`;
    }
}

console.log(sophie.getFullName());
```

Ajouter une fonction canDrinkAlcohol qui retourne vrai si la personne à 21 ans ou plus

What is this ?


```
const sophie = {
    firstName: "Sophie",
    lastName: "Martin",
    age: 34,
    getFullName: function(){
        // affiche l'objet sophie
        console.log(this);
        return `${this.firstName} ${this.lastName.toUpperCase()}`;
    }
}
// Affiche Window
console.log(this);
console.log(sophie.getFullName());
```

This donne le contexte, l'objet sur lequel est exécuté la fonction. Par défaut c'est Window

Les arrow functions


```
//Syntaxe basique
const add = (n1, n2) => {
    console.log(n1 + n2);
};

//Si une seule instruction on peut omettre les accolades
const add2 = (n1, n2) => console.log(n1+ n2);

//Si un seul argument on peut omettre les parenthèses
const sayHello = name => console.log("Hello " + name);

//return implicite pour les fonction qui n'ont qu'une seule ligne
const square = x => x * x;
```

Attention, avec les arrow functions
this n'est pas contextuel,
il fait toujours référence au contexte global

This n'est plus contextuel


```
const point = {
    x: 0,
    y: 0,
    setCoordinates: function(x,y){
        // affiche point
        console.log(this);
        this.x = x;
        this.y = y;
    },
    calcDistance: (x,y) => {
        // affiche window
        console.log(this);
        return Math.sqrt(
            (this.x- x)**2 +
            (this.y - y)**2
        );
    }
};
```

Avec les arrow function, this fait toujours référence au contexte global.

Ici c'est gênant, mais il est des cas où c'est intéressant notamment lorsqu'une fonction callback doit conserver son contexte de déclaration (très utile dans React)