DIT DAKAR INSTITUTE OF TECHNOLOGY

Framework Web Java Script

Prof: Razacki KOUNASSO

20H

Plan du cours

I- Introduction

II- Notions de base du langage JavaScript

III- Les objets JavaScript

VI- La programmation événementielle

V- Le DOM

INTRODUCTION

Voyons ce qu'il y a de si spécial avec JavaScript, ce que nous pouvons réaliser avec ça et quelles autres technologies fonctionnent bien avec lui.

Qu'est-ce que Javascript ?

JavaScript a été initialement créé pour « **rendre les pages Web vivantes** ». Les programmes dans ce langage sont appelés scripts . Ils peuvent être écrits directement dans le code HTML d'une page Web et s'exécuter automatiquement lors du chargement de la page. Les scripts sont fournis et exécutés en texte brut. Ils n'ont pas besoin de préparation spéciale ou de compilation pour fonctionner.

JavaScript est très différent d'un autre langage appelé **Java.**

INTRODUCTION

Aujourd'hui, **JavaScript** peut s'exécuter non seulement dans le navigateur, mais également sur le serveur, ou même sur tout appareil doté d'un programme spécial appelé moteur JavaScript .

Le navigateur dispose d'un moteur embarqué parfois appelé « **JavaScript virtual machine** ».

Différents moteurs ont différents « noms de code ». Par exemple:

- **V8** – dans Chrome, Opera et Edge.
- SpiderMonkey – dans Firefox.
- … Il existe d'autres noms de code comme "Chakra" pour IE, "JavaScriptCore", "Nitro" et "SquirrelFish" pour Safari, etc.

INTRODUCTION

ℹ️ Comment fonctionnent les moteurs ?

Les moteurs sont compliqués. Mais les bases sont faciles.

- Le moteur (embarqué s'il s'agit d'un navigateur) **lit** (« parse ») le **script**.
- Ensuite, il convertit ("compiler") le script en code machine.
- Et puis le code machine s'exécute, assez rapidement.

Le moteur applique des optimisations à chaque étape du processus. Il surveille même le script compilé pendant son exécution, analyse les données qui le traversent et optimise davantage le code machine en fonction de ces connaissances.

INTRODUCTION

Que peut faire JavaScript dans le navigateur ?

JavaScript dans le navigateur peut faire tout ce qui concerne la manipulation des pages Web, l'interaction avec l'utilisateur et le serveur Web..

✅ Par exemple, JavaScript intégré au navigateur est capable de :

- Ajoutez un nouveau code HTML à la page, modifiez le contenu existant, modifiez les styles.
- Réagissez aux actions de l'utilisateur, exécutez les clics de souris, les mouvements du pointeur, les pressions sur les touches.
- Envoyez des requêtes sur le réseau à des serveurs distants, téléchargez et chargez des fichiers (technologies dites AJAX et COMET).
- Obtenez et définissez des cookies, posez des questions au visiteur, affichez des messages.
- Mémorisez les données côté client ("stockage local").
- etc

INTRODUCTION

Que pourrions-nous pas faire avec JavaScript dans le navigateur ?

Les capacités de JavaScript dans le navigateur sont limitées pour protéger la sécurité de l'utilisateur. L'objectif est d'empêcher une page Web malveillante d'accéder à des informations privées ou de nuire aux données de l'utilisateur.

❌ Voici un exemples de restrictions :

JavaScript sur une page Web **ne peut pas lire/écrire des fichiers** arbitraires sur le disque dur, les copier ou exécuter des programmes. Il n'a pas d'accès direct aux fonctions du système d'exploitation.

INTRODUCTION

Qu'est-ce qui rend JavaScript unique ?

Il y a au moins trois choses intéressantes à propos de JavaScript :

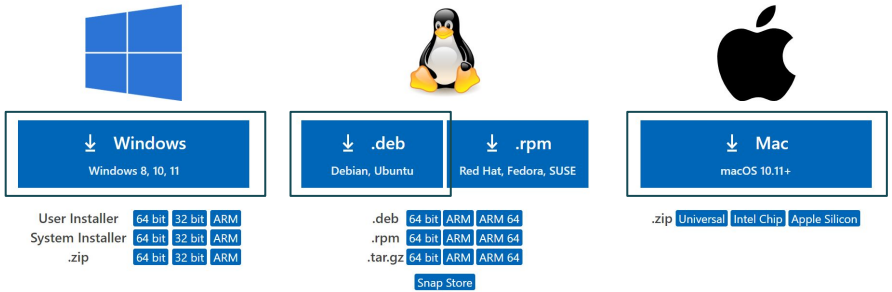
- Intégration complète avec HTML/CSS.
- Les choses simples se font simplement.
- Pris en charge par tous les principaux navigateurs et activé par défaut.

JavaScript est la seule technologie de navigateur qui combine ces trois éléments. C'est ce qui rend JavaScript **unique**. C'est pourquoi c'est l'outil le plus répandu pour créer des interfaces de navigateur. Cela dit, JavaScript peut être utilisé pour **créer des serveurs, des applications mobiles, etc.**

INTRODUCTION

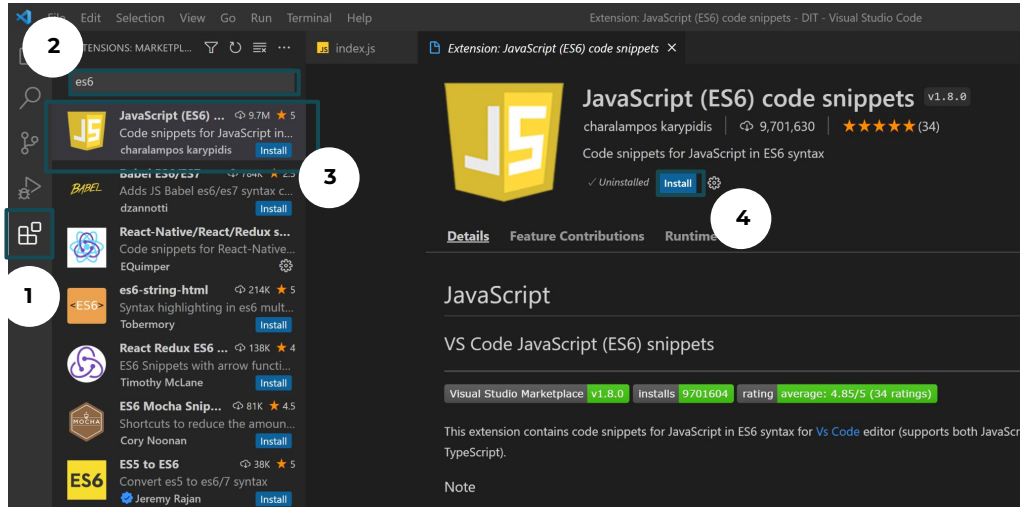
IDE

Pour le cours, nous allons utiliser Visual Studio Code (multiplateforme, gratuit). Vous pouvez télécharger avec le lien suivant (https://code.visualstudio.com/download).



INTRODUCTION

Extension à installer (ES6)



INTRODUCTION

Console développeur

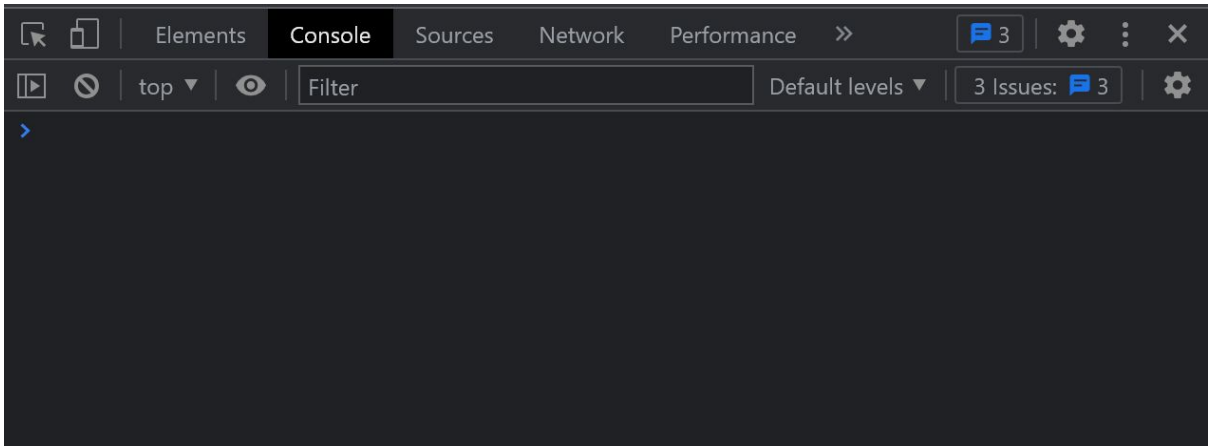
Vous ferez très probablement des erreurs... Oh, de quoi je parle ? Vous allez absolument faire des erreurs, du moins si vous êtes un humain, pas un robot. Mais dans le navigateur, les utilisateurs ne voient pas les erreurs par défaut. Donc, si quelque chose ne va pas dans le script, nous ne verrons pas ce qui est cassé et ne pourrons pas le réparer.

Pour voir les erreurs et obtenir de nombreuses autres informations utiles sur les scripts, des outils de développement ont été intégrés dans les navigateurs. La plupart des développeurs se tournent vers Chrome ou Firefox pour le développement car ces navigateurs disposent des meilleurs outils de développement. Nous resterons sur **Chrome** pour notre cours de JS.

INTRODUCTION

Console développeur (Chrome)

Appuyez sur F12 ou, si vous êtes sur Mac, puis sur .Cmd+Opt+J



INTRODUCTION

Résumé

- JavaScript a été initialement créé en tant que langage réservé aux navigateurs, mais il est maintenant également utilisé dans de nombreux autres environnements.
- Aujourd'hui, JavaScript occupe une position unique en tant que langage de navigateur le plus largement adopté, entièrement intégré à HTML/CSS.
- Il existe de nombreux langages qui sont « **transpilés** » en JavaScript et offrent certaines fonctionnalités. Il est recommandé de les consulter, au moins brièvement, après avoir maîtrisé JavaScript.
- Les outils de développement nous permettent de voir les erreurs, d'exécuter des commandes, d'examiner des variables et bien plus encore.

Notions de base du JS

```
const reverseString = string => [...string].reverse().join('');

// EXAMPLES
reverseString('Medium'); // 'muideM'
reverseString('Better Programming'); // 'gnimmargorP retteB'
```

Notions de base du JS

Hello, DIT!

D'abord, voyons comment nous attachons un script à une page Web. Pour les environnements côté serveur (comme Node.js), vous pouvez exécuter le script avec une commande telle que "***node my.js***".

Les programmes JavaScript peuvent être insérés presque n'importe où dans un document HTML à l'aide de la ***<script>*** balise. Cette balise contient du code JavaScript qui est automatiquement exécuté lorsque le navigateur traite la balise. Elle a quelques attributs qui sont rarement utilisés de nos jours mais qui peuvent encore être trouvés dans l'ancien code: **type** (type="text/javascript" [htm4]), **language**

Notions de base du JS

Hello, DIT!

Par exemple :

Une seule ***<script>*** balise ne peut pas contenir à la fois l' attribut src et le code.

```
8  <script src="/path/to/script.js">
9      // Scripts externes
10 </script>
```

```
1  <!DOCTYPE HTML>
2  <html>
3
4  <body>
5
6      <p>Before the script...</p>
7
8      <script>
9          alert('Hello, DIT!');
10     </script>
11
12     <p>...After the script.</p>
13
14 </body>
15
16 </html>
```

Notions de base du JS

Tp-1 : Afficher une alerte

Créez une page qui affiche un message "Je suis JavaScript!". Faites-le sur votre disque dur, peu importe, assurez-vous simplement que cela fonctionne

Tp-2 : Afficher une alerte avec un script externe

Reprenez la solution de la tâche précédente Afficher une alerte . Modifiez-le en extrayant le contenu du script dans un fichier externe alert.js, résidant dans le même dossier. Ouvrez la page, assurez-vous que l'alerte fonctionne.

Notions de base du JS

Structure du code

La première chose que nous étudierons est les blocs de construction du code.

- Les instructions peuvent être séparées par un point-virgule.

```
<script>
    alert('Hello, DIT!') ; alert("Hello, DIT!")
</script>
```

Notions de base du JS

Structure du code

La première chose que nous étudierons est les blocs de construction du code.

- Habituellement, les instructions sont écrites sur des lignes séparées pour rendre le code plus lisible
- Un point-virgule peut être omis dans la plupart des cas lorsqu'un saut de ligne existe (JavaScript interprète le saut de ligne comme un point-virgule "implicite").

```
12  <script>
13      alert('Hello, DIT!');
14      alert("Hello, DIT!");
15  </script>
```

```
17  <script>
18      alert('Hello, DIT!')
19      alert("Hello, DIT!")
20  </script>
```

Notions de base du JS

Structure du code

La première chose que nous étudierons est les blocs de construction du code.

- Il y a des cas où une nouvelle ligne ne signifie pas un point-virgule

```
22  <script>
23      alert(3 +
24      1
25      + 2);
26  </script>
```

Notions de base du JS

Les Commentaires en JS

Au fil du temps, les programmes deviennent de plus en plus complexes. Il devient nécessaire d'ajouter des commentaires qui décrivent ce que fait le code et pourquoi. Les commentaires peuvent être placés à n'importe quel endroit d'un script. Ils n'affectent pas son exécution car le moteur les ignore simplement.

- Les commentaires d'une ligne commencent par deux barres obliques **//**.
- Les commentaires multilignes commencent par une barre oblique et un astérisque **/*** et se terminent par un astérisque et une barre oblique ***/.**
- Utiliser les touches de raccourci!: **Ctrl+/, Ctrl+Shift+/, cmd+/, cmd+Option+/**

Notions de base du JS

Les Commentaires en JS

Par exemple

```
<script>
    // Ce commentaire occupe une ligne à part entière
    alert('Hello, DIT!'); // Ce commentaire suit la déclaration

    /* Un exemple avec deux messages.
       Il s'agit d'un commentaire multiligne.
    */
    alert("This is a multiline comment.");

</script>
```

Notions de base du JS

Les Variables

Les variables sont utilisées pour stocker ces informations. Une variable est un "stockage nommé" pour les données. Nous pouvons utiliser des variables pour stocker des visiteurs et d'autres données. Pour créer une variable en JavaScript, utilisez le mot clé ***let***.

L'instruction ci-dessous crée (en d'autres termes : déclares) une variable avec le nom **« message ».** Nous pouvons y mettre des données en utilisant l'opérateur d'affectation **=**

```
1  let message;
2  message = 'Hello'; // stocker la chaîne 'Hello' dans la variable nommée message
3  alert(message); // montre le contenu de la variable
```

Notions de base du JS

Les Variables

Pour être concis, nous pouvons combiner la déclaration et l'affectation de la variable en une seule ligne. Nous pouvons également déclarer plusieurs variables sur une seule ligne.

```
/*
définir la variable et
lui attribuer une valeurlet message = 'Hello';
*/
alert(message); // Hello

//déclarer plusieurs variables sur une seule ligne
let user = 'John', age = 25, message2 = 'Hello';
```

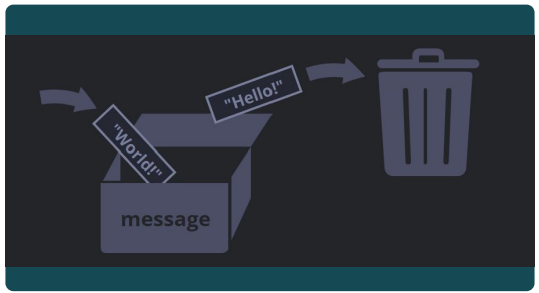
```
let user = 'John';
let age = 25;
let message = 'Hello';
```

```
let user = 'John',
  age = 25,
  message = 'Hello';
```

Notions de base du JS

Les Variables

Nous pouvons facilement saisir le concept de « variable » si nous l'imaginons comme une **« boîte »** pour les données, avec un autocollant au nom unique dessus. Par exemple, la variable message peut être imaginée comme **une boîte étiquetée** "message" avec la valeur "Hello!" qu'elle contient. Nous pouvons mettre n'importe quelle valeur dans la case. Lorsque la valeur est modifiée, les anciennes données sont supprimées de la variable



Notions de base du JS

Les Variables

Il existe deux limitations sur les noms de variables en JavaScript :

1. Le nom ne doit contenir que des lettres, des chiffres ou des symboles <<**$**>> et <<_>>.
2. Le premier caractère ne doit pas être un chiffre.

```
let userName;
let test123;
let $ = 1; // déclare une variable avec le nom "$".
let _ = 2; // et maintenant une variable avec le nom "_".
alert($ + _); // 3
```

Notions de base du JS

Les Variables

ℹ️ **Noms réservés**

Il existe une liste de mots réservés , qui ne peuvent pas être utilisés comme noms de variables car ils sont utilisés par le langage lui-même. Par exemple : ***let, class, return, et function*** sont réservés.

```
1  let let = 5; // ne peut pas nommer une variable "let", erreur !
2  let return = 5; // On ne peut pas non plus l'appeler "return", erreur !
```

Notions de base du JS

Les Variables: Constantes

Pour déclarer une variable constante (immuable), utilisez ***const*** à la place de ***let***. Les variables déclarées à l'aide ***const*** sont appelées "constantes". Ils ne peuvent pas être réaffectés. Une tentative de le faire provoquerait une erreur. Lorsqu'un programmeur est sûr qu'une variable ne changera jamais, il peut la déclarer avec ***const*** pour garantir et communiquer clairement ce fait à tout le monde.

```
1  const myBirthday = '18.04.1982';
2  myBirthday = '01.01.2001'; // erreur, on ne peut pas réassigner la constante !
```

Notions de base du JS

Les Variables: Constantes

✅ **Bonnes pratiques**

Il existe une pratique répandue d'utiliser des constantes comme alias pour les valeurs difficiles à retenir qui sont connues avant l'exécution. Ces constantes sont nommées à l'aide de majuscules et de traits de soulignement.

```
1  const COLOR_RED = "#F00";
2  const COLOR_GREEN = "#0F0";
3  const COLOR_BLUE = "#00F";
4  const COLOR_ORANGE = "#FF7F00";
```

Notions de base du JS

Les Variables: Constantes

✅ **Bonnes pratiques**

Veuillez passer du temps à réfléchir au bon nom pour une variable avant de la déclarer. Cela vous remboursera généreusement. Voici quelques règles à suivre :

- Utilisez des noms lisibles par l'homme comme userNameou shoppingCart.
- Éloignez-vous des abréviations ou des noms courts comme a, b, c, à moins que vous ne sachiez vraiment ce que vous faites.
- Mettez-vous d'accord sur les termes au sein de votre équipe et dans votre esprit. Si un visiteur du site est appelé un « utilisateur », nous devons nommer les variables associées currentUser ou newUser au lieu de currentVisitor ou newManInTown.

Notions de base du JS

Résumé

Nous pouvons déclarer des variables pour stocker des données en utilisant les mots- clés **var**, **let** ou **const**.

- **let**– est une déclaration de variable moderne.
- **var**– est une déclaration de variable à l'ancienne.
- **const**– est comme let, mais la valeur de la variable ne peut pas être modifiée.

Les variables doivent être nommées de manière à nous permettre de comprendre facilement ce qu'elles contiennent.

Notions de base du JS

Tp-3 : Travailler avec des variables

1. Déclarez deux variables : admin et name.
2. Attribuez la valeur "John" à name.
3. Copiez la valeur de name vers admin.
4. Affiche la valeur de admin en utilisant alert(doit sortir "John").

Tp-4 : Travailler avec des variables

1. Créez une variable avec le nom de notre planète. Comment nommeriez-vous une telle variable ?
2. Créez une variable pour stocker le nom d'un visiteur actuel d'un site Web. Comment nommeriez-vous cette variable ?

Notions de base du JS

Tp-5 : Const en majuscule ?

Ici, nous avons une constante birthday pour la date, et aussi l'age. L'age est calculé à partir de birthday l'utilisation de **someCode**(), ce qui signifie un appel de fonction que nous n'avons pas encore expliqué (nous le ferons bientôt !), mais les détails n'ont pas d'importance ici, le fait est que age est calculé d'une manière ou d'une autre sur la base de birthday. Serait-il juste d'utiliser des majuscules pour **birthday**? Pour **age**? Ou même pour les deux ?

```
1  const birthday = '18.04.1982';
2  const age = someCode(birthday);
```

Notions de base du JS

Types de données

Une valeur en JavaScript est toujours d'un certain type. Par exemple, une chaîne ou un nombre. Il existe huit types de données de base en JavaScript. **Sept types** de données primitifs et **un type de données non primitif**.

Type	Description	Exemple
number	Pour les nombres de toute nature : entier ou virgule flottante, les entiers sont limités par .±(2^{53}-1). Outre les nombres normaux, il existe des « valeurs numériques spéciales » qui appartiennent également à ce type de données : Infinity, -Infinity et NaN	let n = 123; n = 12.345;

Notions de base du JS

Type	Description	Exemple
bigint	Pour des nombres entiers de longueur arbitraire. Une BigInt est créée en ajoutant **n** à la fin d'un entier :	const bigInt = 1234567890123456789012345678901234567890**n**;
string	Une chaîne peut avoir zéro ou plusieurs caractères, il n'y a pas de type distinct à caractère unique. (Guillemets doubles : "Hello", Guillemets simples : 'Hello', Backticks : `Hello`)	let str = **"Hello";** let str2 = **'Single quotes are ok too';** let phrase = **`can embed ${str}`;**

Notions de base du JS

Type	Description	Exemple

boolean	Le type booléen n'a que deux valeurs : true et false. Ce type est couramment utilisé pour stocker les valeurs oui/non : true signifie « oui, correct » et false signifie « non, incorrect ».	let nameFieldChecked = true;
		let ageFieldChecked = false;
		let isGreater = 4 > 1;

null	Pour les valeurs inconnues - un type autonome qui a une seule valeur nulle. En JavaScript, null n'est pas une "référence à un objet inexistant" ou un "pointeur nul" comme dans certains autres langages.	let age = null;

Notions de base du JS

Type	Description	Exemple

undefined	Il crée un type qui lui est propre, tout comme null. La signification de undefined est "la valeur n'est pas attribuée". Si une variable est déclarée, mais non affectée, alors sa valeur est undefined	let age;
		alert(age); // affiche "undefined"
		age = undefined;
		alert(age); // affiche "undefined"

symbol	Le type symbol est utilisé pour créer des identifiants uniques pour les objets.	let id = Symbol("id");

Notions de base du JS

Type	Description	Exemple

object	Les objets sont utilisés pour stocker des collections de données et des entités plus complexes.	let user = {
		name: "John",
		store_value "John"
		age: 30
		};

Notions de base du JS

Tp-6 : Const en majuscule ?

Quelle est la sortie du script ?

```
1  let name = "Ilya";
2
3  alert( `hello ${1}` ); // ?
4
5  alert( `hello ${"name"}` ); // ?
6
7  alert( `hello ${name}` ); // ?
```

Notions de base du JS

Interaction: alert, prompt, confirm

Comme nous utilisons le navigateur comme environnement de démonstration, voyons quelques fonctions pour interagir avec l'utilisateur : **alert**, **prompt** et **confirm**.

alert : Celui-ci, nous l'avons déjà vu. Il affiche un message et attend que l'utilisateur appuie sur "OK". La mini-fenêtre avec le message s'appelle une fenêtre modale . Le mot « modal » signifie que le visiteur ne peut pas interagir avec le reste de la page, appuyer sur d'autres boutons, etc. tant qu'il n'a pas traité la fenêtre. Dans ce cas - jusqu'à ce qu'ils appuient sur "OK".

prompt : Elle accepte deux arguments : **title (Le texte à montrer au visiteur) et default (Un deuxième paramètre facultatif, la valeur initiale du champ de saisie).** Il affiche une fenêtre modale avec un message texte, un champ de saisie pour le visiteur et les boutons OK/Annuler.

Notions de base du JS

Interaction: alert, prompt, confirm

confirm: Elle affiche une fenêtre modale avec une question et deux boutons : OK et Annuler. Le résultat est true si OK est pressé et false sinon.

```
1  let age = prompt('Quel âge avez-vous ?', 18);
2  alert(`Vous avez ${age} ans!`); // Vous avez 18 ans !
3
4  let firstname = prompt("What is your firstname?", "default_value");
5  alert(firstname);
```

Notions de base du JS

Conversions de types

Conversion de chaînes: La conversion de chaîne est la plupart du temps évidente. **False devient "false", null devient "null", etc. (value = String(value);)**

Conversion numérique: La conversion numérique se produit automatiquement dans les fonctions et expressions mathématiques : **alert("6" / "2"); let num = Number(str);** Si la chaîne n'est pas un nombre valide, le résultat d'une telle conversion est **NaN.**

- **undefined ====> NaN**
- **null ====> 0**
- **true et false ====> 1 et 0**

Notions de base du JS

Conversions de types

Conversion booléenne: La conversion booléenne est la plus simple. Cela se produit dans les opérations logiques (plus tard, nous rencontrerons des tests de condition et d'autres choses similaires) mais peut également être effectué explicitement avec un appel à **Boolean(value)**.

```
let value = true;
alert(typeof value); // boolean
value = String(value); // la valeur actuelle est une chaîne de caractères "true".
alert(typeof value); // string

let str = "123";
alert(typeof str); // string
let num = Number(str); // devient un numéro 123
alert(typeof num); // number

alert( Boolean(1) ); // true
alert( Boolean(0) ); // false
```

Notions de base du JS

Opérateurs de base, mathématiques

Opérateurs	Exemple
+-*/	2+2, 2-2, 2*2, 2/2
% **(modulo)**	alert(5 % 2); // 1
****** **(Exponentiation)**	alert(2 ** 2); // $2^2 = 4$

Notions de base du JS

Opérateurs de base, mathématiques

Opérateurs	Exemple

+ (concaténation)	let s = "my" + "string";
	alert(s); // mystring
	alert('1' + 2); // "12"

++ (incrémentation)	let x = 3; x++; alert(x) // 3
-- (décrémentation)	let x = 3; x–; alert(x) // 3

Notions de base du JS

Opérateurs de base, mathématiques

ℹ️ **Important**

- L'incrément/décrément ne peut être appliqué qu'aux variables. Essayer de l'utiliser sur une valeur comme 5++ donnera une erreur.
- counter++ : counter = counter + 1,
- counter += 4 : counter = counter + 4
- counter ++ est différent ++counter
- a = b = c = 2 + 2;

Notions de base du JS

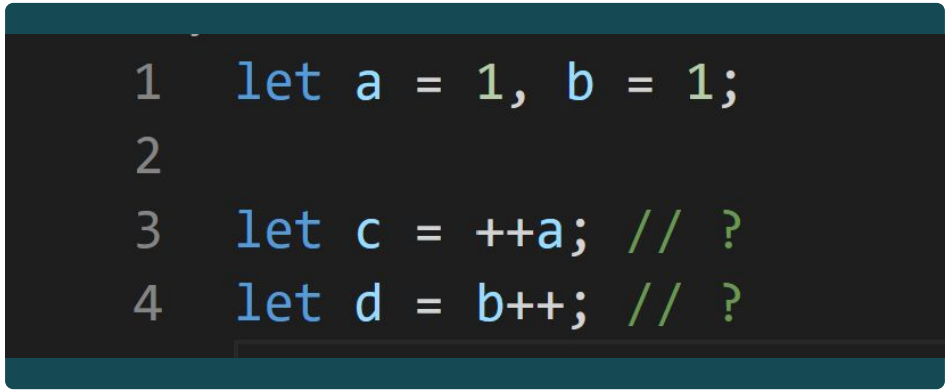
Opérateurs de base, mathématiques

1. **opérateurs logiques**
   a. && et
   b. || ou
   c. ! négation
2. **opérateurs de comparaison**
   a. == égal à, ===
   b. < inférieur à
   c. <= inférieur ou égal à
   d. > supérieur à
   e. >= supérieur ou égal à
   f. != différent de

Notions de base du JS

Tp-7 : Les formes de suffixe et de préfixe

Quelles sont les valeurs finales de toutes les variables a, b , c et d d'après le code ci-dessous ?



```
1  let a = 1, b = 1;
2
3  let c = ++a; // ?
4  let d = b++; // ?
```

Notions de base du JS

Tp-8 : Conversions de types

Quels sont les résultats de ces expressions ?

```
1  "" + 1 + 0
2  "" - 1 + 0
3  true + false
4  6 / "3"
5  "2" * "3"
6  4 + 5 + "px"
7  "$" + 4 + 5
```

```
8  "4" - 2
9  "4px" - 2
10 "  -9  " + 5
11 "  -9  " - 5
12 null + 1
13 undefined + 1
14 " \t \n" - 2
```

Notions de base du JS

Tp-9 : Corriger l'ajout

Voici un code qui demande à l'utilisateur deux nombres et affiche leur somme. Cela ne fonctionne pas incorrectement. La sortie dans l'exemple ci-dessous est 12(pour les valeurs d'invite par défaut). Pourquoi? Répare-le. Le résultat devrait être 3

```
1  let a = prompt("First number?", 1);
2  let b = prompt("Second number?", 2);
3
4  alert(a + b); // 12
```

Notions de base du JS

Structures conditionnelles : if

Structure conditionnelle simple

```
if (condition) {
    // traitement si Vrai
}
```

```
if (condition1) {
    //  traitement si Vrai
} else if (condition2) {
    //  traitement si Vrai
} else {
    //  traitement si Faux
}
```

Structure conditionnelle équilibrée

```

if (condition) {
    // traitement si Vrai
} else {
    // traitement si Faux
}
```

Notions de base du JS

Structures conditionnelles: if

Exemple

```
let a = 3;
let b = 4;

if (a > b) {
    alert(`${3} > ${b}`)
} else {
    alert(`${3} <= ${b}`)
}

// Multiple '?'
let c = 4
    , d = 5;

let e = (c > d) ? c - d : d - c;
alert(e);
```

Notions de base du JS

Structures conditionnelles: switch

```
switch(x) {
    case 'value1':  // if (x === 'value1')
      // ...
      break;

    case 'value2':  // if (x === 'value2')
       // ...
      break;

    default:
      // ...
      break;
}
```

Exemple

```
let a = 2 + 2;

switch (a) {
  case 3:
    alert( 'Too small' );
    break;
  case 4:
    alert( 'Exactly!' );
    break;
  case 5:
    alert( 'Too big' );
    break;
  default:
    alert( "I don't know such values" );
}
```

Notions de base du JS

Tp-10 : Réécrivez le "switch" en un "if"

```
switch (browser) {
    case 'Edge':
      alert( "You've got the Edge!" );
      break;
    case 'Chrome':
    case 'Firefox':
    case 'Safari':
    case 'Opera':
      alert( 'Okay we support these browsers too' );
      break;
    default:
      alert( 'We hope that this page looks ok!' );
  }
```

Notions de base du JS

Tp-11 : Réécrire "if" en "switch"

```
let a = +prompt('a?', '');
if (a == 0) {
  alert( 0 );
}
if (a == 1) {
  alert( 1 );
}
if (a == 2 || a == 3) {
  alert( '2,3' );
}
```

Notions de base du JS

Structures itératives

Nous avons souvent besoin de répéter des actions. Par exemple, sortir les marchandises d'une liste les unes après les autres ou simplement exécuter le même code pour chaque numéro de 1 à 10. Les boucles sont un moyen de répéter le même code plusieurs fois.

Boucle : “for”

```
for (begin; condition; step) {
  // ... loop body ...
}
```

```
for (let i = 0; i < 3; i++) {
  alert(i); // shows 0, then 1, then 2
}
```

Notions de base du JS

Structures itératives

Nous avons souvent besoin de répéter des actions. Par exemple, sortir les marchandises d'une liste les unes après les autres ou simplement exécuter le même code pour chaque numéro de 1 à 10. Les boucles sont un moyen de répéter le même code plusieurs fois.

Boucle : “while”

```
while (condition) {
  // code
  // so-called "loop body"
}
```

```
let i = 0;
while (i < 3) {
  alert( i );
  i++; // shows 0, then 1, then 2
}
```

Notions de base du JS

Structures itératives

Nous avons souvent besoin de répéter des actions. Par exemple, sortir les marchandises d'une liste les unes après les autres ou simplement exécuter le même code pour chaque numéro de 1 à 10. Les boucles sont un moyen de répéter le même code plusieurs fois.

Boucle : “do..while”

```
do {
   // loop body
} while (condition);
```

```
let i = 0;
do {
  alert( i );
  i++;
} while (i < 3);
```

Notions de base du JS

Structures itératives

ℹ️ **Informations**

- L’instruction break permet de sortir immédiatement d'une boucle
- L’instruction continue permet de passer immédiatement à l'itération suivante
- Voir **for...in** pour boucler sur les propriétés de l'objet.
- Voir **for...of** et **iterables** pour boucler sur des tableaux et des objets itérables.

Notions de base du JS

Tp-12 : Sortie des nombres pairs dans la boucle

Utilisez la boucle **for** pour générer des nombres pairs de **2 à 10**.

Tp-13 : Remplacer "pour" par "tant que"

Réécrivez le code en modifiant la boucle for par while sans modifier son comportement (la sortie doit rester la même).

```
for (let i = 0; i < 3; i++) {
    alert( `number ${i}!` );
}
```

Notions de base du JS

Les fonctions

Très souvent, nous devons effectuer une action similaire à de nombreux endroits du script. Par exemple, nous devons afficher un message attrayant lorsqu'un visiteur se connecte, se déconnecte et peut-être ailleurs. Les fonctions sont les principaux blocs de construction du programme. Ils permettent au code d'être appelé plusieurs fois sans répétition.

Nous avons déjà vu des exemples de fonctions intégrées, **alert(message)** comme **prompt(message, default)** et **confirm(question)**. Mais nous pouvons également créer nos propres fonctions.

```
function name(parameter1, parameter2=”defaultValue, ... parameterN) {
    // body
}
```

Notions de base du JS

Les fonctions

```
1  function showMessage() {
2      alert('Hello everyone!');
3  }
```

```
5  function checkAge(age) {
6      if (age >= 18) {
7          return true;
8      } else {
9          return confirm('Do you have permission from your parents?');
10     }
11 }
12
13 let age = prompt('How old are you?', 18);
14
15 if ( checkAge(age) ) {
16   alert( 'Access granted' );
17 } else {
18   alert( 'Access denied' );
19 }
```

Notions de base du JS

Les fonctions

ℹ️ **Informations**

- Pour renvoyer un résultat, il suffit d'écrire le mot clé return
- Dans une fonction les variables locales sont déclarées avec
- le mot clé **var** ou **let**
- Dans ce cas, elles ont une portée limitée à cette seule fonction, contrairement aux variables globales qui sont déclarées à l'extérieur des fonctions.
- Prenez le soin de bien nommer les fonctions : buildDit(), Build_dit()

Notions de base du JS

Tp-14 : Le "else" est-il obligatoire ?

La fonction suivante renvoie true si le paramètre âge est supérieur à 18. Sinon il demande une confirmation et renvoie son résultat :

```
function checkAge(age) {
    if (age > 18) {
        return true;
    } else {
        // ...
        return confirm('Did parents allow you?');
    }
}
```

Notions de base du JS

Tp-15 : Réécrivez la fonction en utilisant '?' ou '||'

La fonction suivante renvoie true si le paramètre age est supérieur à 18. Sinon, il demande une confirmation et renvoie son résultat.

```
function checkAge(age) {
    if (age > 18) {
        return true;
    } else {
        //...
        return confirm('Did parents allow you?');
    }
}
```

Notions de base du JS

Les Objets

Comme nous le savons , il existe huit types de données en JavaScript. Sept d'entre eux sont appelés "primitifs", car leurs valeurs ne contiennent qu'une seule chose (que ce soit une chaîne ou un nombre ou autre).

En revanche, les **objets** sont utilisés pour stocker des collections à clé de diverses données et d'entités plus complexes. En JavaScript, les objets pénètrent presque tous les aspects du langage. Nous devons donc les comprendre avant d'aller en profondeur ailleurs.

Un objet peut être créé avec des crochets **{...}** de chiffres avec une liste facultative de propriétés . Une propriété est une paire "**clé : valeur**", où **key est une chaîne (également appelée "nom de propriété"), et value peut être n'importe quoi.**

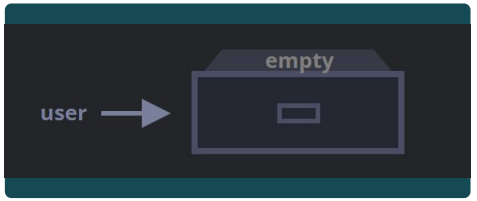
Notions de base du JS

Les Objets

On peut imaginer un objet comme une armoire avec des dossiers signés. Chaque donnée est stockée dans son fichier par la clé. Il est facile de trouver un fichier par son nom ou d'ajouter/supprimer un fichier.

Un objet vide ("armoire vide") peut être créé en utilisant l'une des deux syntaxes suivantes :

```
let user = new Object();
let user_2 = {};
```

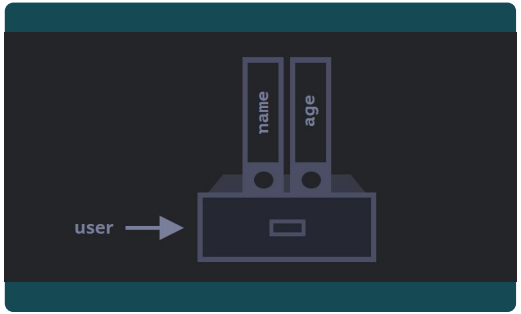


Notions de base du JS

Les Objets

Exemple: Dans l' objet user, il y a deux propriétés : La première propriété a le nom "name" et la valeur "John" et le second a le nom "âge" et la valeur 30. Pour accéder à la valeur des propriétés il faut utiliser un **point. user.name** nous retournera dans notre cas “John”

```
let user = {     // an object
   name: "John",  // by key "name" store value "John"
   age: 30        // by key "age" store value 30
};
alert(user.age) // affiche 30
```



Notions de base du JS

Les Objets

ℹ️ **Informations**

- un objet désigne un type d'entité qui peut posséder des caractéristiques appelées propriétés et des fonctions appelées méthodes.
- Les méthodes effectuent des opérations sur les propriétés et sont susceptibles de fournir certains résultats.
- On accède ensuite aux propriétés et méthodes d'un objet grâce à l'opérateur point

Notions de base du JS

Les Objets

Exemple

```
var rectA = {
    nom: "A",
    longueur: 20,
    largeur: 15,
    posX: 30,
    posY: 25
}
```

```
rectA.perimetre = function () {
    return 2 * (this.longueur + this.largeur);
}
rectA.surface = function () {
    return this.longueur * this.largeur;
}
alert(rectA.surface());
```

Notions de base du JS

Les Tableaux

Les objets vous permettent de stocker des collections de valeurs à clé. C'est très bien. Mais assez souvent, nous constatons que nous avons besoin d'une collection ordonnée , où nous avons un 1er, un 2ème, un 3ème élément et ainsi de suite. Par exemple, nous en avons besoin pour stocker une liste de quelque chose : utilisateurs, marchandises, éléments HTML, etc. Il existe une structure de données spéciale nommée **Array**, pour stocker les collections ordonnées. Il existe deux syntaxes pour créer un tableau vide :

```
let arr = new Array();
let arr_2 = [];
```

```
let fruits = ["Apple", "Orange", "Plum"];

alert( fruits[0] ); // Apple
alert( fruits[1] ); // Orange
alert( fruits[2] ); // Plum

fruits[2] = 'Pear';

fruits[3] = 'Lemon';
```

Notions de base du JS

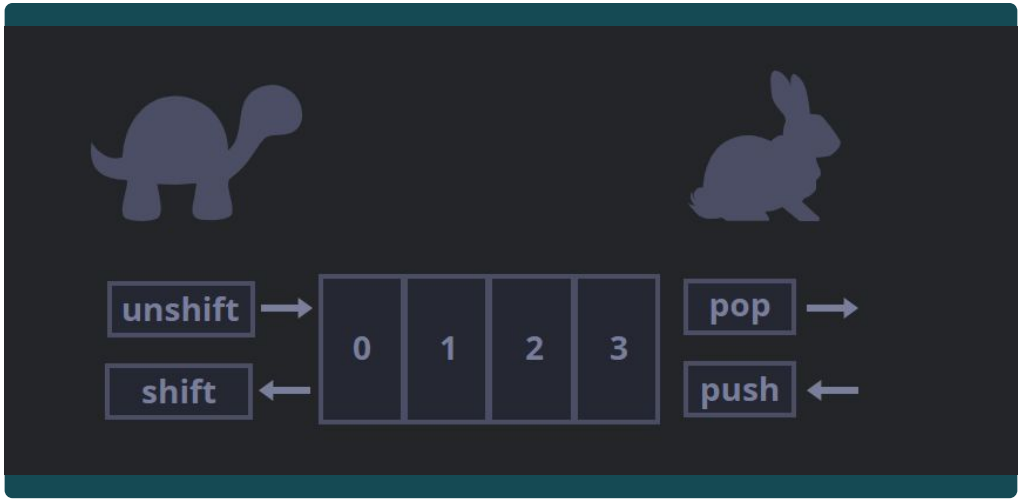
Les Tableaux

ℹ️ Informations

- arr.at(i) est exactement le même que arr[i], si i >= 0
- arr.at(-1) permet d'avoir accès à la dernière valeur du tableau
- arr.push(element) ajoute un élément à la fin du tableau
- arr.pop() supprimer le dernier élément du tableau
- arr.shift() Extrait le premier élément du tableau
- arr.unshift(element) Ajoute l'élément au début du tableau
- arr.length : retourne la taille du tableau

Notions de base du JS

Les Tableaux



Notions de base du JS

Tp-16 : Opérations sur les tableaux.

Essayons 5 opérations de tableau.

1. Créez un tableau **styles** avec les éléments « **Jazz** » et « **Blues** ».
2. Ajoutez "**Rock-n-Roll**" à la fin.
3. Remplacez la valeur du milieu par "**Classiques**". Votre code pour trouver la valeur médiane devrait fonctionner pour tous les tableaux de longueur impaire.
4. Supprimez la première valeur du tableau et affichez-la.
5. Préfixez **Rap** et **Reggae** au tableau.

Notions de base du JS

Tp-16 : Opérations sur les tableaux.

Le tableau dans le processus (résultat attendu) :

Jazz, Blues

Jazz, Blues, Rock-n-Roll

Jazz, Classics, Rock-n-Roll

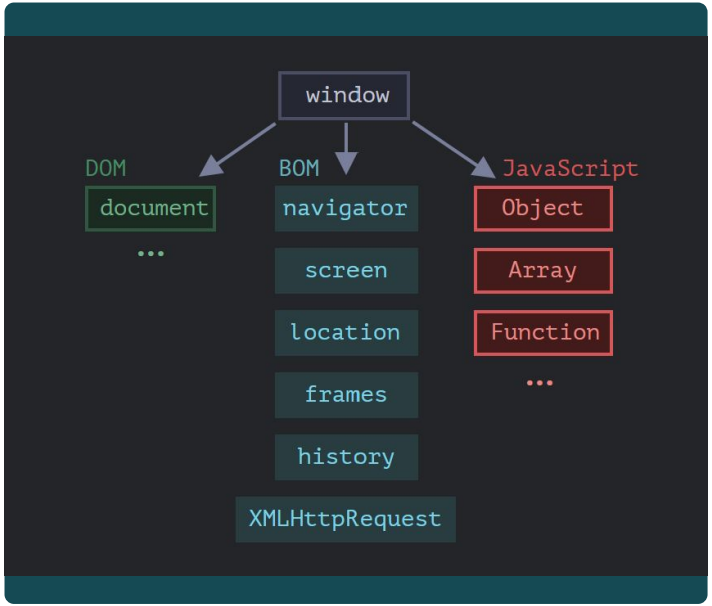
Classics, Rock-n-Roll

Rap, Reggae, Classics, Rock-n-Roll

Les objets JavaScript

Objet window

L'objet window est l'objet par excellence dans Javascript, car il est le parent de chaque objet qui compose la page web. Il a deux rôles. Premièrement, c'est un objet global pour le code JavaScript, et deuxièmement, il représente la fenêtre du navigateur et fournit des méthodes pour la contrôler.



Les objets JavaScript

Objet window

Propriété	Description	Exemple
defaultStatus	Contenu standard de la barre d'état	window.defaultStatus="Bienvenue";
closed	Renvoie l'état d'une fenêtre (false=ouvert, true=fermé)	if(window.closed) { ... }
name	Nom de la fenêtre	alert(window.name);

Les objets JavaScript

Objet window

Méthode	Description	Exemple
setInterval();	Répéter une action pendant un certains nombre de temps	setInterval(() => { alert(“Répéter”) }, 1000);
setTimeout();	Mettre un delay avant l'exécution d’un code	setTimeout(() => { alert(“Répéter”) }, 1000);
close();	Ferme la fenêtre	window.close();
open();	Ouvre la fenêtre	f1=window.open(" accueil.html","accueil")

Les objets JavaScript

Objet document

Le modèle d'objet de document, ou DOM en abrégé, représente tout le contenu de la page sous forme d'objets pouvant être modifiés. L' objet document est le principal point d'entrée de la page. Nous pouvons modifier ou créer n'importe quoi sur la page en l'utilisant. Ici, nous avons utilisé **document.body.style**, mais il y en a beaucoup, beaucoup plus.

```
// change the background color to red
document.body.style.background = "red";
// change it back after 1 second
setTimeout(() => document.body.style.background = "", 1000);
```

Les objets JavaScript

Objet document

Propriété	Description	Exemple
bgColor	Couleur de fond du document	document.bgColor=blue;
fgColor	Couleur de texte du document	document.fgColor=black;
URL	URL complète du document en cours	alert(document.URL);

Les objets JavaScript

Objet document

Méthode	Description	Exemple
write()	Écrit une chaîne de caractères dans le document en cours	document.write(" Hello!");
writeln()	Écrit une chaîne de caractères suivie d'un retour à la ligne dans le document en cours	document.writeln(" Hello! "); document.writeln(" Bonjour!");

Les objets JavaScript

Objet String

- **Propriété**
  1. length : retourne la longueur de la chaîne de caractères
- **Méthodes**
  - charAt( ) : renvoie le caractère se trouvant à une certaine position;
  - charCodeAt( ) : renvoie le code du caractère se trouvant à une certaine position;
  - concat( ) : permet de concaténer 2 chaînes de caractères;
  - fromCharCode( ) : renvoie le caractère associé au code;
  - indexOf( ) : permet de trouver l'indice d'occurrence d'un caractère dans une chaîne;
  - slice( ) : retourne une portion de la chaîne;
  - substr( ) : retourne une portion de la chaîne;
  - substring( ) : retourne une portion de la chaîne;
  - toLowerCase( ) : permet de passer toute la chaîne en minuscule;
  - toUpperCase( ) : permet de passer toute la chaîne en majuscules;

Les objets JavaScript

Objet Array

- **Propriété**
  1. length : retourne le nombre d'éléments du tableau
- **Méthodes**
  - concat( ) : permet de concaténer 2 tableaux;
  - join( ) : convertit un tableau en chaîne de caractères;
  - reverse( ) : inverse le classement des éléments du tableau;
  - slice( ) : retourne une section du tableau;
  - sort( ) : permet le classement des éléments du tableau

Les objets JavaScript

Objet Date

- **Méthodes**
  - getDate( ) : renvoie le numéro du jour dans le mois, donc un nombre entier entre 1 et 31;
  - getMonth( ) : renvoie le numéro du mois dans l'année, mais attention ce numéro est un entier entre 0 (pour janvier) et 11 (pour décembre);
  - getFullYear( ) : renvoie le numéro de l'année en 4 chiffres;
  - getHours( ) : elle renvoie le numéro de l'heure;
  - getMinutes( ) : renvoie le numéro des minutes
  - getSeconds(): renvoie le numéro des secondes.

Les objets JavaScript

Objet Date

Exemple

```
<html>
<head>
<title>Heure objet date</title>
</head>
<body>
<script>
    let temps = new Date();
    document.write("<p>il est ",
     temps.getHours(),
     " heures",
     temps.getMinutes(),
     " minutes sur le poste client</p>");
</script>
</body>
</html>

```

Les objets JavaScript

Objet Math

- **Propriété**
  - E : renvoie la valeur de la constante d'Euler (~2.718);
  - LN2 : renvoie le logarithme népérien de 2 (~0.693);
  - LN10 : renvoie le logarithme népérien de 10 (~2.302);
  - LOG2E : renvoie le logarithme en base 2 de e (~1.442);
  - LOG10E : renvoie le logarithme en base 10 de e (~0.434);
  - PI : renvoie la valeur du nombre pi (~3.14159);
  - SQRT1_2 : renvoie 1 sur racine carrée de 2 (~0.707);
  - SQRT2 : renvoie la racine carrée de 2 (~1.414);
- **Méthodes**
  - – abs( ), exp( ), log(), sin( ), cos( ), tan( ), asin( ), acos( ), atan( ), max( ), min( ), sqrt( ) sont les opérations mathématiques habituelles;

Les objets JavaScript

Objet Math

- **Méthodes**
  - atan2( ) : retourne la valeur radian de l'angle entre l'axe des abscisses et un point;
  - ceil( ) : retourne le plus petit entier supérieur à un nombre;
  - floor( ) : retourne le plus grand entier inférieur à un nombre;
  - pow( ) : retourne le résultat d'un nombre mis à une certaine puissance;
  - random( ) : retourne un nombre aléatoire entre 0 et 1;
  - round( ) : arrondit un nombre à l'entier le plus proche.

La programmation événementielle

Notion d'événement

Un événement est un signal que quelque chose s'est produit. Un clic souris, une saisie de données, un chargement de page, le passage de la souris sur une zone,... constituent des événements. Ces événements peuvent provoquer des actions à l'aide de scripts JavaScript.

Voici une liste des événements DOM les plus utiles, juste pour jeter un coup d'œil

- **Événements de souris**
  - click : lorsque la souris clique sur un élément (les appareils à écran tactile le génèrent sur un tap).
  - contextmenu : lorsque la souris fait un clic droit sur un élément.
  - mouseover/ mouseout : lorsque le curseur de la souris survole / quitte un élément.
  - mousedown/ mouseup : lorsque le bouton de la souris est enfoncé/relâché sur un élément.
  - mousemove : lorsque la souris est déplacée.

La programmation événementielle

Notion d'événement

- **Événements clavier**
  - keydown et keyup : lorsqu'une touche du clavier est enfoncée et relâchée.
- **Événements d'élément de formulaire**
  - submit : lorsque le visiteur soumet un <form>.
  - focus : lorsque le visiteur se concentre sur un élément, par exemple sur un <input>
- **Documenter les événements**
  - DOMContentLoaded : lorsque le HTML est chargé et traité, DOM est entièrement construit.
- **Événements CSS**
  - transitionend : lorsqu'une animation CSS se termine.

La programmation événementielle

Gestionnaires d’événements

Pour créer un gestionnaire d'événement, il suffit d'utiliser une balise et d'ajouter le mot clé de l'événement avec un code JavaScript indiquant l'action à entreprendre si l'événement se produit

<BALISE onQuelquechose="code javascript">

```
<input value="Click me" onclick="alert('Click!')" type="button">
```

La programmation événementielle

Gestionnaires d'événements

- **Événement onClick**
  - Se produit lorsque l'utilisateur clique sur un élément spécifique dans une page, comme un lien hypertexte, une image, un bouton, du texte, etc.
  - Ces éléments sont capables de répondre séparément à cet événement
  - Il peut également être déclenché lorsque l'utilisateur clique n'importe où sur la page s'il a été associé non pas à un élément spécifique, mais à l'élément body tout entier

```
<input value="Click me" onclick="alert('Click!')" type="button">
```

La programmation événementielle

Gestionnaires d'événements

- **Événement onMouseover**
  - Analogue à onClick sauf qu'il suffit que l'utilisateur place le
  - pointeur de sa souris sur l'un des éléments précités (lien
  - hypertexte, image, bouton, texte, etc.) pour qu'il ait lieu
- **Événement onMouseout**
  - A l'inverse de onMouseover, cet événement se produit lorsque le pointeur de la souris quitte la zone de sélection d'un élément

```
<img src="image1.webp" onMouseOver="this.src='image2.jpg';" onMouseOut="src='image1.webp';">
```

La programmation événementielle

Tp-17 : Changement de la couleur de l’arrière plan

Écrire un formulaire du type menu déroulant permettant de choisir la couleur de l'arrière-plan à volonté.

```
<form>
    <SELECT
    onChange="ecrirez votre  code JS ici">
    <option value="40E0D0"> Turquoise
    <option value="2E8B57"> Vert comme la mer
    <option value="87CEEB"> Bleu comme le ciel
    <option value="F4A460"> Brun comme le sable
    <option value="FFF0F5"> Bleu lavande
    <option value="FF1493"> Rose
    <option value="FFFFFF" selected> Blanc
    </SELECT>
 </form>
```

La programmation événementielle

Tp-18 : Horloge digitale

Réaliser une horloge digitale donnant le jour et l'heure

Tp-19 : Dessin animé

On veut simuler à partir d'images un ouvrier en train de courir. Implémentez les fonctions suivantes :

- **imageSuivante()** : qui à chaque appel affiche l'image suivante
- **courir()** : qui appelle l'image suivante toutes les 10 secondes
- **stop()** : qui permet de stopper le processu

La programmation événementielle

Tp-19 : Dessin animé

Le DOM

Tp-19 : Dessin animé