Développeur Web et web mobile

WWW.GRETANET.COM

GRETA AIX-MARSEILLE

Copyright- 2023 @ Audrey POEYDOMENGE

Prérequis : HTML et CSS

JavaScript
Initiation

MDN documentation écrite par Mozilla

Programmer et partager du JS en ligne
Sauvegarde dans le cloud, rendu en temps réel :
https://codepen.io/pen
https://jsfiddle.net
http://jsbin.com

Greta-Cfa
Formation | Accompagnement | Apprentissage

Copyright- 2023 @ Audrey POEYDOMENGE

Initiation

Introduction
JavaScript
Performance et référencement
ECMAScript
Utilisation
Code JS
Module
Affichage
Script et instruction
Commentaire et indentation
Outils de developpement

Variables
Définition
Valeurs et types
Mots-clé let et const ES 6
Référence
typeof
Manipulation de variable : opérateurs arithmétiques
Manipulation de variable : opérateurs de comparaison
Comparateur strict

Les fonctions
Définition
Syntaxe déclarative
Des fonctions natives
Les paramètres
Des paramètres par défaut
Fonction anonyme
Fonction fléchée ES 6
IIEF
Récursive
closure
Structures de contrôle conditionnelle
Structures itératives
Portée des variables (scope)
Syntaxe et mots-clé JS

Les tableaux
Définition et méthodes
Structure itérative forEach
for of - for in ES 6
Affectation par décomposition ES 6
Collection MAP ES 6
Collection SET ES 6
Rest Parameter ES 6
Spread Operator ES 6

Greta - Cfa
Formation | Accompagnement | Apprentissage

Copyright- 2023 @ Audrey POEYDOMENGE

Initiation

Les Objets
Programmation orientée objet
Paradigme
Les objets
Les caractéristiques d'un objet
Objet littéral
L'objet Object
Fonction constructeur
Programmation orientée prototype

Les Objets natifs
String
Number
Array
Boolean
Date
Timer
Math
RegExp

Browser Object Model
Propriétés et méthodes de window
L'objet Document
L'objet Location
L'objet Screen

Document Object Model
Noeud
Propriétés de noeud
Hiérarchie des noeuds
Des propriétés et méthodes pour naviguer (parentNode, childNode)
DOM
Propriétés de l'objet document
Méthode de l'objet document
Manipuler les contenus (innerHTML, innerText)
Manipuler les styles CSS (style)
Manipuler les classes (classList, add)
Manipuler les attributs (getAttribute & setAttribute)
Créer des elements (createElement, appendChild, createTextNode)
Les littéraux de templates ES 6

Les évènements
Fonctionnement
Attribut d'évènement
Écouteur d'événement
Des événements
L'objet Event

Développeur Web et web mobile

WWW.GRETANET.COM

GRETA AIX-MARSEILLE

Copyright- 2023 @ Audrey POEYDOMENGE

JavaScript

Le JavaScript est un langage qui a été créé en 1995 par Brendan Eich et qui est devenu une norme ECMA (organisation chargée de sa standardisation) en 1997.

Il s'agit d'un langage de programmation sous forme de scripts qui est directement exécuté/interprété par tous les navigateurs (multiplateforme = chaque navigateur à son propre moteur de rendu, v8 pour chrome, spider monkey pour firefox, chakra pour edge).

Pour fonctionner, on place le code JS dans la page HTML et lorsque le navigateur charge la page, l'interpréteur intégré lit le code et l'exécute. Il est donc traité à l'exécution.

JavaScript contient une bibliothèque standard d'objets tels que STRING, NUMBER, ARRAY etc...

JS est actuellement le langage de programmation le plus utilisé car sa polyvalence favorise le developpement front-end (coté navigateur) et back-end(coté serveur). Utilisé côté serveur, il est capable de remplir l'ensemble des fonctionnalités d'un langage serveur et de retourner du html.

C'est un langage qui dispose d'une grande communauté et d'un large écosystème (librairies et frameworks).

Mais surtout c'est un langage à l'écoute de tous les événements (clic, chargement, hover...), à chaque événement intercepté, une action peut être déclenchée.

Enfin, c'est un langage orienté objet qui utilise la notation pointée pour accéder aux propriétés (attributs) et fonctionnalités (méthodes) de chaque élément.

Copyright- 2023 @ Audrey POEYDOMENGE

JavaScript

Le JavaScript est un langage qui a été créé en 1995 par Brendan Eich et qui est devenu une norme ECMA (organisation chargée de sa standardisation) en 1997.

Il s'agit d'un langage de programmation sous forme de scripts qui est directement exécuté/interprété par tous les navigateurs (multiplateforme = chaque navigateur à son propre moteur de rendu, v8 pour chrome, spider monkey pour firefox, chakra pour edge).

Pour fonctionner, on place le code JS dans la page HTML et lorsque le navigateur charge la page, l'interpréteur intégré lit le code et l'exécute. Il est donc traité à l'exécution.

JavaScript contient une bibliothèque standard d'objets tels que STRING, NUMBER, ARRAY etc...

JS est actuellement le langage de programmation le plus utilisé car sa polyvalence favorise le developpement front-end (coté navigateur) et back-end(coté serveur). Utilisé côté serveur, il est capable de remplir l'ensemble des fonctionnalités d'un langage serveur et de retourner du html.

C'est un langage qui dispose d'une grande communauté et d'un large écosystème (librairies et frameworks).

Mais surtout c'est un langage à l'écoute de tous les événements (clic, chargement, hover...), à chaque événement intercepté, une action peut être déclenchée.

Enfin, c'est un langage orienté objet qui utilise la notation pointée pour accéder aux propriétés (attributs) et fonctionnalités (méthodes) de chaque élément.

Copyright- 2023 @ Audrey POEYDOMENGE

Performance et référencement

Le JS a un impact très important sur le temps de chargement et de rendu de la page dans le navigateur.

La part des visiteurs qui ne consultent qu'une page en moins de quelques secondes est appelée taux de rebond.

Les temps de chargement et le taux de rebond influencent le référencement.

On privilégiera les sites qui apportent une réponse satisfaisante en donnant les meilleurs résultats pour fidéliser les utilisateurs.

Chaque ligne contient une seule instruction et le point virgule signale clairement la fin d'une instruction (ainsi que la phase de minification du code).

La minification (par un outil externe) réduit le poids d'un script en supprimant tous les éléments qui ne sont pas utiles à son fonctionnement: espaces, tabulations, retour à la ligne, commentaires.
De plus variables et fonctions sont renommées pour alléger encore plus le fichier.

La minification accélère le chargement sur le réseau.

Un fichier minifié prend la forme suivante : script.min.js

Copyright- 2023 @ Audrey POEYDOMENGE

ECMAscript

ECMAScript est un standard de langage de programmation, édité par Ecma International. il définit la syntaxe, les types de variable, etc ...

Javascript est un des langages qui respecte le standard ECMAScript, tout comme d'autres langages qui respectent ce standard (ActionScript ou JScript).

Le standard ECMAScript évolue, ce qui explique les chiffres 5, 6, 7, etc..
Il évolue pour que les langages basés dessus comme le Javascript soient de plus en plus puissants, plus simples et plus riches. A chaque version, on ajoute des fonctionnalités, on simplifie la syntaxe, etc..

ES5: publié en 2009
ES6: publié en 2015, aussi appelé ES2015 *Beaucoup de choses ont évolué en ES6... Fonctions Fléchées, Classes, Modules, ...*
les versions suivantes apportent peu de nouveautés :
ES14: publié en 2023, (ES2023)

Si le navigateur ne comprend pas le code (nouvelle fonctionnalité qu'il ne supporte pas), on utilise des polyfills et des transpilers. Un polyfill est un bout de code que l'on ajoute pour que la nouvelle fonctionnalité marche dans les navigateurs.
Un transpiler (ex: babel) est un traducteur de code qui va convertir la syntaxe ES6 vers ES5.

Copyright- 2023 @ Audrey POEYDOMENGE

Utilisation de JavaScript

Utilisé côté client, JS permet de dynamiser et d'ajouter de l'interactivité utilisateur/page :

-HTML étant un langage descriptif, JS va lui permettre de modifier le comportement du code en réalisant des services dynamiques, strictement cosmétiques ou à des fins ergonomiques (effets stylistiques)
- contrôle des données saisies dans les formulaires HTML (validation formulaire)
- interaction avec le document HTML via l'interface DOM du navigateur (ajout/suppression d'éléments, modification etc...)
- modification du contenu des pages web en effectuant des appels vers le serveur avec la méthode Ajax (Asynchronous Javascript And XML)
- sauvegarde de données sur le poste local
- gestion des nombres, dates et heures
- création d'animations graphiques et CSS
- création de diaporamas, infobulles
- etc ...

Il n'est pas possible de cacher son code source. Toutefois des système d'obfuscation (obscurcissement) existent pour rendre le code difficilement compréhensible par un humain, en le compactant au maximum et en changeant entres autres le nom des fonctions et des variables.

Copyright- 2023 @ Audrey POEYDOMENGE

Code JS

En interne les scripts JS (autant que souhaité) se placent entre les balises <script> </script> (l'attribut type="text/javascript" n'est plus obligatoire) dans le head ou avant la fermeture du body ou les deux. Une page web est analysée de manière séquentielle. Donc si un JS volumineux est déclaré en début de page, celui-ci va être chargé avant les éléments à afficher, ce qui risque de ralentir la navigation. Il est donc recommandé de placer vos balises script en fin de page. Néanmoins les navigateurs modernes ont tendance à charger le JS en dernier, quelque soit sa position dans la page.

<script> code JS </script>

En réalité il est trés fortement recommandé de placer les scripts dans des fichiers externes dont l'extension est .js
En modularisant le code JS (en le séparant du html), il est plus facile à lire et à maintenir, et surtout le script peut servir à plusieurs pages web différentes.
De plus, les fichiers mis en cache dans le navigateur, accélèrent le chargement des pages!
On appelle notre fichier grâce à la balise script dont on renseigne l'attribut src, que l'on place soit dans le head soit dans le body.
! Attention, les scripts externes ne doivent pas contenir de balises <script> !

<script src="carousel.js"></script> L'avantage d'un fichier séparé est que le navigateur le télécharge et le stocke dans son cache. Les autres pages qui font référence au même script le prendront dans le cache au lieu de le télécharger, cela réduit le trafic et rend les pages plus rapides.

Copyright- 2023 @ Audrey POEYDOMENGE

Code JS

Des instructions JS peuvent être déclenchées grâce à des événements associés à certains de nos éléments HTML.

<button onclick="alert('encore coucou');">push me</button>
survolez-moi

Il est donc possible de placer du JS directement dans un tag HTML, dans les attributs dédiés à la gestion des événements : onclick, onload, onmouseover, etc...

Enfin, il est aussi possible d'appeler un script hébergé sur un autre domaine .
<script src="https://cdnj.cloudflare.com/libs/jquery.min.js">
Ici on appelle la bibliothèque jQuery

CDN (content delivery network)
Un réseau de diffusion qui améliore la vitesse de transfert grâce à des implantations dans toutes les zones géographiques, donc au plus près des utilisateurs.

Et pour les navigateurs obsolètes ?
<noscript> Votre navigateur n'accepte pas le JS </noscript>
Si le JS est activé, le bloc HTML est ignoré, si le JS est désactivé le bloc HTML est affiché.

Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

Module

Nous découpons notre application en modules qui sont autant de fichiers sources séparés. Puis nous rendons visibles les modules à l'aide d'export. Nous n'avons plus qu'à récupérer les parties qui nous intéressent dans d'autres modules à l'aide d'import. Permet une meilleure maintenance ainsi que la réutilisation du code.

Création module d'export, ex : word.js

```
function countWords (text){                                    // fonction traditionnelle
    console.log(text.split(' ').length);
}
const txtReplace = (text) => {                                 // fonction fléchée dans variable
    console.log(text.replace('lorem', 'HOHOHO').trim());
}
export {countWords, txtReplace};
```

Création module d'import, ex : script.js

```
import { countWords, txtReplace } from './word.js';

countWords("lorem ipsmum dolor");
normalizeSpacing("lorem ipsum lorem dolor");
```


Copyright- 2023 @ Audrey POEYDOMENGE

Affichage

JS permet d'afficher des données de différentes manières :

- directement dans le document HTML grâce à la méthode write() de l'élément document.
  document.write("hello world"); // si utilisé après chargement de la page supprime tout le code existant.

- dans une boîte d'alert grâce à la méthode alert() de l'élément window.
  window.alert("hello world"); // une boîte de dialogue (modale) est bloquante, elle impose à l'utilisateur une action pour qu'il puisse continuer sa navigation.

- dans la console du navigateur en utilisant la méthode log() de l'élément console.
  console.log("hello world"); // utiliser à des fins de test et de débogage.

- dans un élément HTML grâce à la propriété innerHTML de l'élément (définie le contenu textuel)

On utilise la notation pointée pour appeler les fonctions (méthodes) propres à un objet.

Ici window, document et console sont des objets.
document et console sont des objets rattachés à l'objet global window.
Étant l'objet racine, on peut omettre l'écriture de l'objet window. alert("hello world");
Par défaut toutes les propriétés (variables) et méthodes (fonctions) appartiennent à l'objet window.

Copyright- 2023 @ Audrey POEYDOMENGE

Script et instructions

Un programme informatique c'est une simple liste d'instructions à exécuter, comme une recette en cuisine. En HTML, les programmes JS sont exécutés par le navigateur web.

Une instruction peut être composée de valeurs, d'opérateurs, de mots-clés etc...

Les instructions sont exécutées une par une, dans l'ordre ou elles sont écrites.

Une instruction se termine généralement par un point virgule, ce n'est pas obligatoire mais trés fortement recommandé, surtout en cas de minification de fichier.

On écrit généralement une instruction par ligne mais il est possible d'en écrire plusieurs sur la même ligne à condition de bien les séparer par des points virgule.

JS tolère les espaces, ils sont même recommandés pour une meilleure lisibilité du code, une bonne pratique consiste à en mettre autour des opérateurs.

Les instructions JS peuvent être regroupées dans des blocs, à l'intérieur d'accolades { }. et plus précisément à l'intérieur de ce que l'on appelle fonction.
Cela signifie que ces instructions doivent être exécutées ensemble.

Copyright- 2023 @ Audrey POEYDOMENGE

Commentaires et indentation

Commentaire

Comme dans tous les langages informatiques, il est possible de commenter son code :

// commentaire sur une ligne
/* commentaire sur plusieurs lignes */

Un commentaire est une portion de code non interprétée.
Celle-ci peut servir à simplifier la lecture par un agent humain, à documenter, ou à désactiver temporairement une ou plusieurs instructions.

Les commentaires permettent une maintenance corrective et évolutive.

Chaque fichier JS doit débuter par une intro avec auteur, date, but du fichier, version...

Indentation

Permet de regrouper des éléments de même niveau de profondeur.
L'indentation ajoute quelques octets à la taille du code généré, mais l'augmentation est infime par rapport au gain en lisibilité.
La tabulation n'occupe qu'un seul octet au lieu de 2 ou 4 pour les espacements!

Copyright- 2023 @ Audrey POEYDOMENGE

Outils de developpement

Comme pour le HTML et CSS il existe des outils de developpement pour le JavaScript et ce à travers plusieurs onglets :

console

Cet onglet affiche par défaut l'ensemble des erreurs détectées.
Mais c'est aussi un outil d'aide à la programmation et au débogage.
On peut directement écrire dedans pour exécuter du code, effectuer des tests, interagir avec la page en cours (notation pointée pour accéder aux éléments du document).
C'est une sorte de bac à sable.
Attention tout disparaît lorsque l'on rafraîchit la page !

network

Cet onglet affiche l'enchaînement des appels réseaux/requêtes des différentes ressources de la page, de tout ce qui est appelé pour être chargé : img, css, js, vidéos, font etc...

stockage

Cet onglet permet de visualiser toutes les ressources déposées sur notre machine : storage et cookie... et permet également de les supprimer !

Copyright- 2023 @ Audrey POEYDOMENGE

Variables

La programmation consiste en la manipulation de données, afin de produire un resultat en sortie de traitement. Pour stocker ces données on crée et utilise des variables, plus précisément des conteneurs/boîtes ou des zones de mémoire (en mémoire vive RAM) destinées à stocker ces données/valeurs. A savoir qu'une variable peut varier au cours du temps, lors de l'exécution d'un script.

Pour se servir de quelque chose, il faut que ce quelque chose existe, qu'il soit créé / fabriqué. La création d'une variable s'appelle aussi déclaration d'une variable , elle se fait à l'aide d'un opérateur (mot-clé) suivi du nom de la variable et éventuellement d'un opérateur d'affectation avec son initialisation (valeur initiale). En effet, si la déclaration est obligatoire, l'initialisation ne l'est pas et peut se faire plus tard.

```
mot-clé + nom = valeur ;  // = est l'opérateur d'affectation
```

Un nom peut être alphanumérique, peut contenir des dollars et underscore, mais ne doit jamais commencer par un nombre ni contenir de tiret (réservé à la soustraction), d'espace ou @. L'utilisation du signe dollar n'est pas très courante en JS, mais les programmeurs professionnels l'utilisent souvent comme alias pour la fonction principale dans des bibliothèques JavaScript. Le JS est sensible à la casse (fait la différence entre minuscule et majuscule), prenom et PRENOM sont deux variables différentes. La convention veut que l'on code en camelCase (majuscule pour la 1ere lettre des mots suivants le premier).

Il est possible de déclarer plusieurs variables avec un seul opérateur en séparant les variables par des virgules.

Copyright- 2023 @ Audrey POEYDOMENGE

Valeurs et types

Différentes valeurs peuvent être affectées à des variables :

- Des chaînes de caractères : type string
  Elles doivent être placées entre guillemets simple ', double " ou backtick `.
  Attention aux chaînes de caractères qui contiennent des apostrophes ou guillemets ! Il faut penser à échapper le caractère avec un antislash : ' le temps aujourd\'hui est ensoleillé '
  Écrire une valeur sur plusieurs lignes provoque une erreur, il faut donc soit écrire le caractère \n pour provoquer un saut de ligne soit utiliser le backtick qui permet également de faire une interpolation de variable `${a} ${b} `
  On peut additionner/concaténer des chaînes de caractères ensemble.

- Des nombres entiers ou flottants (point pour signifier la virgule) : type number
  Les nombres s'écrivent sans guillemets; un nombre écrit entre guillemets est une chaîne !

- Des valeurs qui sont vraies ou fausses : type boolean

- Des tableaux : type array
  Les tableaux permettent de rassembler plusieurs valeurs dans une même variable
  Une liste de valeurs qui peuvent être de différents types

- Des objets : type object
  Les objets sont des tableaux améliorés

- undefined, une variable est déclarée mais pas définie, pas d'assignement (absence de valeur)

Le JS est un langage à typage dynamique, les variables sont typées au moment de leur initialisation.

Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

Mots-clé : const et let

const
Ce mot clé permet de créer une variable qui ne pourra pas être réassignée/modifiée. Une variable déclarée avec const doit obligatoirement être initialisée. Une constante est donc destinée à être fixe, une valeur stable dans le temps. Une valeur en lecture seule.

```
const PI = 3.1415;
PI = 2;  // Uncaught TypeError: Assignment to constant variable.
```

Recommandation : utiliser const par défaut quitte à modifier par let par la suite.

let
Le mot-clé let permet de créer une variable dont la valeur peut évoluer dans le temps, au fil de l'exécution du script.

```
let compteur = 0;
```

Attention: on ne peut pas re-déclarer une variable déjà déclarée avec let ou const.

```
// SyntaxError: 'x' has already been declared
```

var (obsolète)
Le mot-clé var est presque l'équivalent du let (scope différent), mais son utilisation n'est plus préconisée. var title, author; // la re-déclaration de variable était possible avec var :(

Copyright- 2023 @ Audrey POEYDOMENGE

Référence

Comment sont passées les variables ?

const a = “Bonjour” ; // a vaut “bonjour”

const b = a ; // b vaut “bonjour”

si on change la valeur de b :

b = “hello” ; // b vaut maintenant “hello” et a vaut toujours “bonjour”

Lorsque l’on crée une nouvelle variable à partir d’une autre variable, c’est seulement la valeur qui est passée. Les deux variables ont la même valeur mais ne font pas référence à la même valeur !

Lorsque l’on crée des variables avec des nombres, chaînes de caractères, tableaux ou objets à valeur fixe, on dit que ce sont des objets littéraux.

Copyright- 2023 @ Audrey POEYDOMENGE

typeof

Pour connaître le type d'une variable on utilise l'instruction :

typeof + nomVariable

```
const a = "Hello World!" ;  typeof a ;        // affiche "string"
const b = 48 ;              typeof b ;        // affiche "number"
const mineur = true ;       typeof mineur;    // affiche "boolean"
const users = [ ] ;         typeof users ;    // affiche "object"
const book = { } ;          typeof book ;     // affiche "object"
```

Toutes les variables sont en réalité par défaut des objets !

Chaque objet JS a des propriétés et méthodes déjà définies.

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/String
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Number
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Écrire des variables qui contiennent :
- prenom
- nom
- age
- profession
- obtention du permi de conduire
- hobbie

... Et faire un rendu dans le document.
Essayer de concaténer ces variables avec des chaînes de caractères pour rendre le tout lisible et compréhensible.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Écrire des variables qui contiennent :
- prenom
- nom
- age
- profession
- obtention du permi de conduire
- hobbie

... Et faire un rendu dans le document.
Essayer de concaténer ces variables avec des chaînes de caractères pour rendre le tout lisible et compréhensible.

Correction

```
const firstN = “Paul”, name = “Dubois”, age = 36, job = “tech”, licence = true, hobby = “music”;

document.write(“Mon cousin s’appelle “ + firstN + “ “ + name + “, il a “ + age + “ ans, bosse dans la “ + job + “, et adore la “ + hobby);
```


Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Déclarer plusieurs variables, en leur affectant des valeurs de votre choix, puis les afficher ainsi que leur type dans les interface de votre choix.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Déclarer plusieurs variables, en leur affectant des valeurs de votre choix, puis les afficher ainsi que leur type dans les interface de votre choix.

Correction

```
const name = “Pierre”;
const age = 26;
const majeur = true;

console.log(“name est de type “ + typeof name  + “ et sa valeur est “ + name);
document.write(“age est de type “ + typeof age  + “ et sa valeur est “ + age);
alert(“majeur est de type “ + typeof majeur  + “ et sa valeur est “ + majeur);
```


Copyright- 2023 @ Audrey POEYDOMENGE

Manipulation de variable : opérateurs arithmétiques

Comme en algèbre, les variables contiennent des valeurs et peuvent être utilisées dans les expressions :

```
const x = 5 ;
const y = 20 ;
const z = x +y ;
```

Comme en algèbre, on peut faire de l'arithmétique avec les variables :

additionner	+	
soustraire	-	
multiplier	*	
diviser	/	
modulo	%	// en programmation le modulo sert essentiellement pour savoir si un nombre est paire ou impaire

Et surtout comme en algèbre on oublie pas les parenthèses pour prioriser les opérations !

Il est aussi possible de concaténer des chaînes de caractères et d'additionner un nombre avec une chaîne de caractères, dans ce cas la, le nombre est considéré comme une chaîne et du coup il y a concaténation : console.log("22" + 38); // affiche "2238"

Attention: dans une multiplication, une chaîne de caractères sera quant à elle considérée comme un nombre ! console.log("22" * 38); // affiche 836

Copyright- 2023 @ Audrey POEYDOMENGE

Manipulation de variable : opérateurs de comparaison

Des opérateurs de comparaison :

```
const a = 22, b = 58;

==   égal                 // false
!=   différent             // true
<    inférieur            // true
<=   inférieur ou égal     // true
>    supérieur            // false
>=   supérieur ou égal     // false
```

Les opérateurs de comparaison comparent des nombres ou des expressions qui retourne true si la comparaison réussi ou false si elle échoue. Quand les opérandes sont de types différents, ils sont convertis avant comparaison.

Des opérateurs logiques :

```
&&   et    a < b && a < 30;// true
||   ou    a < b || a > 24;        // true
```

servent à vérifier une ou plusieurs conditions.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Calcul de la surface d'un rectangle.

Afficher dans la page la surface d'un rectangle de 4 mètres sur 10.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Calcul de la surface d'un rectangle.

Afficher dans la page la surface d'un rectangle de 4 mètres sur 10.

Solution

```
const width = 10;
const height = 4;
const surface = width * height;
document.write("La surface du rectangle est de : " + surface + " m2");
```


Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Calcul de la surface d'un rectangle.

Afficher dans la page la surface d'un rectangle de 4 mètres sur 10.

Solution

```
const width = 10;
const height = 4;
const surface = width * height;
document.write("La surface du rectangle est de : " + surface + " m2");
```

Sujet

Conversion de Celsius en Fahrenheit

Afficher à l'écran le resultat de la conversion de 25°C en Fahrenheit.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Calcul de la surface d'un rectangle.

Afficher dans la page la surface d'un rectangle de 4 mètres sur 10.

Solution

```
const width = 10;
const height = 4;
const surface = width * height;
document.write("La surface du rectangle est de : " + surface + " m2");
```

Sujet

Conversion de Celsius en Fahrenheit

Afficher à l'écran le resultat de la conversion de 25°C en Fahrenheit.

Solution

```
const celsius = 25;
const farenheit = (celsius * 9/5) + 32;
document.write("la conversion de 25°C en Farenheit est égal à : " + farenheit + " farenheit");
```


Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Calcul du prix total d'un panier

Calculer le prix d'un panier pour l'achat de 3 articles de 22, 47 et 13 euros.
Ajouter une taxe de 20% et afficher le total à l'écran.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Calcul du prix total d'un panier

Calculer le prix d'un panier pour l'achat de 3 articles de 22, 47 et 13 euros.
Ajouter une taxe de 20% et afficher le total à l'écran.

Solution

```
const price1 = 22;
const price2 = 47;
const price3 = 13;
const taxe = .2;                    // 20%

const totalArticle = (price1 + price2 + price3);
totalTaxe = totalArticle * .2;

const finalPrice = totalArticle + totalTaxe;

document.write("Votre panier s'élève à : " + finalPrice + " euros");  // 98.40
```


Copyright- 2023 @ Audrey POEYDOMENGE

Les fonctions : Définition

On appelle fonction un sous-programme qui permet d'effectuer un ensemble d'instructions par simple appel de cette fonction dans le corps principal du programme.

Une fonction c'est un bloc de code que l'on peut exécuter à la demande.

C'est un peu comme des variables, à la différence qu'elles peuvent contenir plusieurs choses et qu'elles peuvent exécuter du code avec ou sans paramètre.

Les fonctions permettent d'exécuter un même programme dans plusieurs parties d'un programme principal.

Les fonctions permettent une simplicité du code et donc une taille de programme optimisée.

Copyright- 2023 @ Audrey POEYDOMENGE

Les fonctions : syntaxe

Une fonction est un ensemble d'instructions délimitées par des accolades.

```
{
   INSTRUCTION 1
   INSTRUCTION 2   // comme des étapes dans une recette pour réaliser une tâche
   INSTRUCTION 3
}
```

Tout comme les variables, une fonction doit être déclarée pour exister et être utilisable.

Déclaration d'une fonction :

```
function nom( ) { instructions }
```

Utilisation de la fonction :

```
nom( );
```

Une fonction qui retourne un resultat utilise le mot-clé return en fin d'instruction.
Une fonction qui ne retourne pas de résultat est appelée une procédure.

Une fonction peut prendre des paramètres, c'est à dire des variables transmises lors de l'appel à la fonction, pour la faire fonctionner.

Créer des fonctions est une manière de structurer le code en gagnant en clarté mais surtout de le rendre réutilisable dans des projets divers et variés.

Copyright- 2023 @ Audrey POEYDOMENGE

Des fonctions natives

Un certains nombres de fonctions sont dites natives, elles sont déjà implémentées côté navigateur :

console.log( )	affiche un message dans la console du navigateur
console.warm( )	affiche un message d'alerte dans la console du navigateur
console.info( )	affiche un message d'information dans la console du navigateur
console.error( )	affiche un message en rouge dans la console du navigateur
alert()	affiche un message dans une boîte de dialogue
write()	affiche directement dans la fenêtre du navigateur

confirm( )	affiche une boîte qui attend une action utilisateur : true ou false
prompt( )	affiche une boîte de dialogue et attend une saisie utilisateur

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Il s'agit ici de créer une fonction qui permet de dire bonjour dans la console du navigateur.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Il s'agit ici de créer une fonction qui permet de dire bonjour dans la console du navigateur.

Correction

déclaration d'une fonction de présentation d'utilisateur :

```
function sayHello(){
        console.log("Hello world!");
}
```

utilisation de la fonction :

```
sayHello();
sayHello();
```


Copyright- 2023 @ Audrey POEYDOMENGE

Les fonctions : paramètres

Visualisons les fonctions comme une sorte de machine à laver. Lors de l'utilisation de celle-ci il y a trois étapes :

1. en entrée on dépose du linge
2. on traite ce linge
3. en sortie on récupère le linge

Plus concrètement, ce que l'on fournit à la fonction lors de sa déclaration ou de son utilisation, ce ne sont ni plus ni moins que des valeurs en dur ou des variables déclarées en amont qui vont nous être nécessaires pour le traitement de la fonction.

Syntaxe :

Déclaration d'une fonction :

```
function nom(parametres) { instructions }
```

Utilisation de la fonction :

```
nom(arguments);
```

Lors de la déclaration d'une fonction, on parle de paramètres pour parler des variables fournies en entrée. Et lors de l'utilisation de la fonction on parle d'arguments passés à la fonction !

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Fonction avec paramètre.

Il s'agit ici de créer une fonction de présentation d'utilisateur et d'afficher le résultat de différents appels à cette fonction dans l'interface de votre choix.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Fonction avec paramètre.

Il s'agit ici de créer une fonction de présentation d'utilisateur et d'afficher le résultat de différents appels à cette fonction dans l'interface de votre choix.

Correction

déclaration d'une fonction de présentation d'utilisateur :

```
function presentation(name, age, job){
        console.log("L'utilisateur " + name + " a " + age + " ans, sa profession est " + job;
}
```

utilisation de la fonction :

```
presentation("Pierre", 22, "webtech");
presentation("Martine", 43, "fleuriste");
presentation("Claude", 26, "comptable");
```


Copyright- 2023 @ Audrey POEYDOMENGE

ES 6 Les fonctions : paramètres par défaut

Les paramètres de fonctions peuvent prendre des valeurs par défaut :

```
function sePresenter(prenom, nom = 'Wangler'){
    console.log(prenom + " " + nom);
}

sePresenter("Pierre");                  //  1 argument, affiche Eric + le paramètre par défaut Wangler
sePresenter("Martine", "Dupont");  //  le 2eme argument écrase le paramètre par défaut affiche Xavier Dulac
```

Ordre des paramètres

En JS les arguments sont évalués de gauche à droite, la 1ère valeur fournie prend la place du 1er paramètre. On doit toujours placer les paramètres par défaut à la fin !

```
function presentation(prenom = "Claire", nom){
    console.log(prenom + " " + nom);
}

        presentation("Victoria");  // affiche Victoria undefined
```


Copyright- 2023 @ Audrey POEYDOMENGE

Fonction anonyme

Il n'est en rien obligatoire de donner un nom à une fonction. Il est parfaitement possible d'en déclarer une comme ceci :

```
function( ){
        console.log("Hello World !");
}
```

Mais dans ce cas, comment y faire appel ?
Il suffit pour cela de stocker cette fonction dans une variable :

```
const hello = function( ){
        console.log("Hello World !");
};
```

Notez au passage la présence du point virgule après la fermeture de notre fonction, indispensable pour conclure l'affectation.

Nous retrouvons généralement les fonctions anonymes en fonction de callback dans les gestionnaires d'événements.

Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

Fonction fléchée (arrow)

Grâce aux fonctions fléchées nous allons pouvoir écrire un code plus concis.

Syntaxe :
- suppression du mot-clé déclaratif function
- suppression des parenthèses si argument unique
- parenthèses si pas d'argument du tout
- parenthèses si plusieurs arguments
- suppression des accolades en cas d'instruction unique
- suppression du mot clé return si instruction unique
- fat arrow => placé entre la signature et le bloc d'instructions

Ex: argument unique et instruction unique

```
let test = name => console.log("Bonjour " + name);
test("Pierre");
```

Ex: plusieurs arguments et instruction unique

```
let multiplicateur = (number1, number2) => console.log(number1 * number2);
multiplicateur(2,4);
```

Dans la pratique, nous retrouverons les fonctions fléchées principalement dans les gestionnaires d'événements, à la place des fonctions anonymes :

```
const element = document.getElementById('element');
element.addEventListener('mousemove', () => console.log("element survolé"));
```


Copyright- 2023 @ Audrey POEYDOMENGE

IIEF

Il existe un type particulier de fonction anonyme, que nous appelons les IIFE pour Immediately-Invoked Function Expression - expression de fonction immédiatement invoquée.

Comme son l'indique il s'agit d'une fonction qui s'exécute immédiatement après avoir définie sans avoir à l'affecter à une variable pour l'appeler.

Cela permet de créer un environnement de portée locale, en encapsulant le code dans une fonction anonyme immédiatement appelée, en évitant ainsi les collisions de noms avec d'autres parties du code.

Ce genre de fonction nous empêche cependant d'y faire appel ultérieurement.

```
(function( ){
        let message = "Hello World !";
        console.log(message);
})( );

(fonction)( );
```

Les IIFE sont couramment utilisés dans les bibliothèques et les frameworks JavaScript pour créer des modules qui ont leur propre portée locale.

Copyright- 2023 @ Audrey POEYDOMENGE

Récursive

Une fonction récursive en JavaScript est une fonction qui s'appelle elle-même à l'intérieur de sa propre définition.

Cela peut être utilisé pour résoudre des problèmes qui peuvent être décomposés en des sous-problèmes similaires plus petits, en utilisant une approche de diviser pour mieux régner.

Lorsqu'une fonction récursive est appelée, elle continue à s'appeler elle-même jusqu'à ce qu'une condition de sortie soit atteinte, ce qui empêche les appels de fonction de se produire indéfiniment.

```
function sum(number){
        if(number === 1){
                return 1;
        }
        else{
                return number + sum(number - 1);
        }
}
```

Dans cet exemple, la fonction s'appelle elle-même avec un argument inférieur à number jusqu'à ce que la valeur 1 soit atteinte, puis remonte la pile d'appels en additionnant.

Copyright- 2023 @ Audrey POEYDOMENGE

Closure (fermeture)

une closure est une fonction qui a accès aux variables d'une fonction parente, même après que la fonction parente ait été exécutée.

Les closures sont créées chaque fois qu'une fonction est déclarée, ce qui permet à la fonction interne de capturer et de conserver l'état de la portée locale à laquelle elle appartient, même après que la fonction externe ait terminé son exécution.

```
function increment( ){

        let count = 0;

        return function( ){
                count++;
                console.log(count);
        }
}

const addOne = increment();

addOne(); // affiche 1
addOne(); // affiche 2
addOne(); // affiche 3
```


Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Écrire une fonction qui permet via une boîte de dialogue, de demander à l'utilisateur :

- son prenom
- son age

... puis d'afficher ces deux informations dans le document.

Pour appeler cette fonction nous utiliserons le bouton suivant :

<button type = "submit" onclick = "display( )">Afficher mes information</button>

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Écrire une fonction qui permet via une boîte de dialogue, de demander à l'utilisateur :

- son prenom
- son age

... puis d'afficher ces deux informations dans le document.

Pour appeler cette fonction nous utiliserons le bouton suivant :

<button type = "submit" onclick = "display( )">Afficher mes information</button>

Correction

prompt( ) est une fonction native qui permet d'afficher une boîte de dialogue

```
const prenom = prompt("Merci d'entrer votre prenom");
const age = prompt("merci d'entrer votre age");

document.write(prenom + " a " + age + " ans");
```


Copyright- 2023 @ Audrey POEYDOMENGE

TP : la calculette

Sujet

Écrire une ou des fonctions qui permettent de réaliser des :

- additions
- soustractions
- multiplications
- divisions

et d'afficher les résultats en console.

ATTENTION
Quel que soit le texte saisi, l'ordre prompt() renvoit toujours une valeur de type chaîne.
Il faudra penser à convertir cette valeur !

Copyright- 2023 @ Audrey POEYDOMENGE

TP : la calculette

Sujet

Écrire une ou des fonctions qui permettent de réaliser des :

- additions
- soustractions
- multiplications
- divisions

et d'afficher les résultats en console.

ATTENTION
Quel que soit le texte saisi, l'ordre prompt() renvoit toujours une valeur de type chaîne.
Il faudra penser à convertir cette valeur !

Correction

parseInt(string) permet de convertir une variable string en variable number entier
parseFloat(string) permet de convertir une variable string en variable number à virgule

Copyright- 2023 @ Audrey POEYDOMENGE

TP : nombre premier

Sujet

tester si un nombre est premier un nombre premier n'est divisible que par 1 et lui même.

Il s'agit ici de créer une fonction avec paramètre pour mettre en place un petit calcul simple qui permette de tester si cet argument est un nombre, et si ce nombre est premier.

Enfin faites en sorte de pouvoir demander à un utilisateur de rentrer un nombre, pour afficher si celui-ci est premier ou non.

Indice

Les fonctions suivantes pourront vous être utiles :
isNaN(number) : permet de tester si un argument est égal à NaN (Not A Number)
parseInt(arg) : revoit un entier de l'argument (NaN si l'argument n'est pas un nombre)
parseFloat(arg) : renvoi le nombre décimal

Pensez également aux opérateurs booléens, ainsi qu'au reste de la division entière (modulo).

Copyright- 2023 @ Audrey POEYDOMENGE

Structure de contrôle : les conditions

Un programme est composé d'une série d'instructions qui s'enchaînent séquentiellement les unes à la suite des autres. Une structure conditionnelle est une structure qui se caractérise par les différents cheminements (embranchements) possibles en fonction d'un résultat d'un test (conditions) qui renvoit soit true soit false.

- if else

```
if (condition) {
    instructions
}
else if (autre condition) {
    instructions
}
else{
    instructions
}
```

Les conditions sont des tests effectués grâce aux opérateurs de comparaison ET / OU logique.

L'exécution des instructions se fait entre les accolades { } comme pour les fonctions.

Copyright- 2023 @ Audrey POEYDOMENGE

Structure de contrôle : les conditions

Une variable peut prendre différentes valeurs distinctes qui conduisent à des traitements différents.

- switch

```
switch(expression){
        case val1 : instructions;
        break;
        case val2 : instructions;
        break;                          break pour sortir de la boucle une fois la bonne valeur rencontrée
        default : instructions;         instructions exécutées si rien ne correspond
}
```

- ternaire

```
let note = 8;
note > 10 ? "bravo" : "nul!";
      |          |        |
condition    true     false
               if       else
```

Les ternaires sont des conditions trés rapides à écrire mais qui sont difficiles à lire et à débugger.

Copyright- 2023 @ Audrey POEYDOMENGE

Comparateur strict

Lorsque l'on compare une chaîne de caractères avec un nombre, JS va naturellement convertir la chaîne en nombre si cela est possible.

== est un comparateur simple

Avec le comparateur simple, JS convertir la chaîne en nombre si cela est possible et compare les valeurs.

=== est un comparateur strict

JS compare tout d'abord le type. Si celui-ci est identique alors il continue et compare la valeur. Il ne convertit pas.

```
const nombre = 28;
const mot = "28";

nombre == mot;      // true
nombre === mot;   // false
```


Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Demander son âge au visiteur via une boîte de dialogue et en fonction du résultat afficher en console s'il est mineur ou majeur.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Demander son âge au visiteur via une boîte de dialogue et en fonction du résultat afficher en console s'il est mineur ou majeur.

Correction

```
let age = prompt("Merci d'indiquer votre age ?");

if(age >= 18){
        console.log("Vous êtes majeur et pouvez donc poursuivre votre navigation sur le
        site");
}
else{
        console.log("Vous êtes mineur, merci de bien vouloir quitter le site");
}
```


Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Demander son âge au visiteur via une boîte de dialogue et en fonction du résultat afficher en console s'il s'agit d'un adolescent, d'un adulte ou d'un sénior.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Demander son âge au visiteur via une boîte de dialogue et en fonction du résultat afficher en console s'il s'agit d'un adolescent, d'un adulte ou d'un sénior.

Correction

```
let age = prompt("Merci d'indiquer votre age ?");

if(age <= 18){
        console.log("Vous êtes un adolscent");
}
else if(age <= 60){
        console.log("Vous êtes un adulte");
}
else{
        console.log("Vous êtes un sénior");
}
```


Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Demander à un utilisateur de rentrer une couleur dans une boîte de dialogue.

Selon la couleur, plusieurs gages sont envisageables ;)

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Demander à un utilisateur de rentrer une couleur dans une boîte de dialogue.

Selon la couleur, plusieurs gages sont envisageables ;)

Correction

```
let color = prompt("Merci d'indiquer une couleur ?");

switch(color){
        case "rouge" : "Aller chercher le café pour tout le monde !";
        break;
        case "vert" : "Passer un coup de balai dans la salle !";
        break;
        case "bleu" : "Nettoyer tous les postes informatiques !";
        break;
        case "jaune" : "Dire à l'accueil qu'il sont un peu trop sevère sur les horaires !";
        break;
        Default : "Ouf vous avez eu chaud mais finalement pas de gage";
}
```


Copyright- 2023 @ Audrey POEYDOMENGE

Structure itérative : les boucles

Les structures itératives permettent de répéter des traitements tant qu'une expression booléenne (condition) est vérifiée/vraie/true.

- while (tant que)
Exécution d'instructions tant qu'une expression est vraie

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

- do while (tant que)
Semblable à while mais les instructions sont exécutées au moins une fois, le test ayant lieu à l'issu du premier traitement.

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

- for (conseillé pour les tableaux pour parcourir l'ensemble des valeurs)
Les instructions sont répétées un nombre de fois connus.

```
for(let i = min; i < max; i++) {
    instructions;
}
```


Copyright- 2023 @ Audrey POEYDOMENGE

La portée des variables

Jusqu'à l'arrivée d'ES6, les scopes sont uniquement des scopes de fonctions, et le sont toujours tant que l'on utilise le mot-clé var. Les variables créées avec let et const ne respectent pas les scopes de fonctions, elles respectent les scopes de blocs.

Un scope de variable est un périmètre où il est possible d'utiliser cette variable.

- scope global, la variable appartient à la page et peut être utilisée par tout le monde, toutes les fonctions, conditions, boucles etc...

- scope de fonction, la variable est locale à la fonction, elle est créée et détruite à chaque fois que la fonction est exécutée. En dehors de cette fonction, aucun code ne peut y accéder.

- scope de bloc, la variable est locale au bloc
  Un bloc est un bout de code qui débute et termine par des accolades { }
  comme les fonctions, conditions, boucles.

Dans un bloc, les variables déclarées avec le mot-clé var sont accessibles depuis l'extérieur du bloc, elles sont considérées comme globales.
Celles déclarées avec let et const à l'intérieur d'un bloc ne sont pas accessibles depuis l'extérieur, elles sont considérées comme locales.

Copyright- 2023 @ Audrey POEYDOMENGE

Synthèse sur la syntaxe et mots-clés JS

On appelle syntaxe l'ensemble des règles d'écriture pour un langage donné, comment on crée des choses et comment on les utilise. La syntaxe comprend les mot-clé qui permettent eux d'identifier les actions à effectuer :

var pour déclarer une variable
let pour déclarer une variable de bloc
const pour déclarer une constante de bloc

if, else pour identifier un bloc d'instruction à exécuter selon une condition
switch, break, default pour identifier un bloc d'instruction à exécuter selon différents cas

for, while pour identifier un bloc d'instructions à exécuter dans une boucle

function pour déclarer une fonction
return pour retourner une valeur en sortie de fonction

try , catch pour implémenter la gestion des erreurs dans un bloc d'instructions

etc ...

Les mots-clés JS sont des mots réservés que l'on ne peut donc pas utiliser comme nom de variables ni de fonction !

Copyright- 2023 @ Audrey POEYDOMENGE

Les tableaux

Un tableau est une variable mémoire qui stocke plusieurs valeurs indépendantes, indexées par des numéros. L'index d'un tableau commence à 0 !

Déclaration d'un tableau vide
let myArray = [ ] ;

Déclaration d'un tableau avec affectation de valeurs
let myFriends = ["Pierre", "Paul", Jacques]; // ... qu'est ce que tu fais la ?

N'importe quelle valeur valable en JS peut être utilisée comme valeur (string, number, booléen...)

Il existe des fonctions natives JS qui nous permettent d'ajouter, récupérer, modifier, supprimer des valeurs... Les tableaux sont riches de propriétés et méthodes pour manipuler les données :

length	pour connaître le nombre d'éléments dans le tableau
push(data)	ajoute une valeur à la fin
unshift(data)	ajoute une valeur au début
[index]	récupérer/accéder à une valeur à un index donné
[index] =	pour modifier une valeur à un index donné
shift( )	retire la 1ère valeur du tableau
pop( )	retire la dernière valeur du tableau

Copyright- 2023 @ Audrey POEYDOMENGE

Les tableaux

Sujet

À vous de jouer !

Libre à vous de créer un tableau vide ou plein d'amis, de hobbys, destinations de voyage..

... et de vous amuser à en rajouter/supprimer via les méthodes appropriées.

Et surtout ne pas hésiter à afficher tout cela dans le document.

Copyright- 2023 @ Audrey POEYDOMENGE

Les tableaux

Sujet

À vous de jouer !

Libre à vous de créer un tableau vide ou plein d'amis, de hobbys, destinations de voyage..

... et de vous amuser à en rajouter/supprimer via les méthodes appropriées.

Et surtout ne pas hésiter à afficher tout cela dans le document.

Correction

```
let myFriends = ["Pierre", "Paul", Jacques];

myFriends.length;
myFriends.push("qu'est ce que tu fais la?");
console.log(myFriends[1];
```


Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

Structure itérative : for of et for in

```
let fruitsRouges = ["fraise", "framboise", "cerise"];
```

for of : permet de récupérer les valeurs

```
for(let fruit of fruitsRouges) {
  console.log("La " + fruit + " c'est bon !");
}
```

La fraise c'est bon ! La framboise c'est bon ! La cerise c'est bon !

for in : permet de récupérer les index et les valeurs

```
for(let fruit of fruitsRouges) {
  console.log(fruits);
console.log(fruitsRouges[fruit]);
}
```

Un objet est une sorte de tableau associatif et for in est pratique sur ces objets pour récupérer les propriétés et leurs valeurs.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Écrire un programme qui affiche dans le document une liste de films que vous appreciez fortement. Comment faire si je veux changer une valeur par une autre au moment de parcourir le tableau ?

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Écrire un programme qui affiche dans le document une liste de films que vous appreciez fortement. Comment faire si je veux changer une valeur par une autre au moment de parcourir le tableau ?

Correction

On utilise trés souvent la boucle for pour parcourir un tableau.

```
const movies = ["l'histoire sans fin", "scream", "peur bleue", "kill bill"];

for(let i = 0; i < movies.length; i++){
    if(movies[i] === "scream"){
        movies[i] = "scream3";
    }
    document.write(movies[i] + " ");
};
```


Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

Destructuring

L'affectation par décomposition (destructuring assignment) permet d'extraire des valeurs de données stockées dans des tableaux et des objets.

sans décomposition :

```
var users = ["Pierre", "Paul", "Jacques"];
var firstUser = users[0];
var secondUser = users[1];
var thirdUser = users[2];
console.log(firstUser, secondUser, thirdUser);
```

avec décomposition :

```
let [movie1, movie2, movie3] = ["Scream1", "Scream2", "Scream3"];
console.log(movie1, movie2, movie3);
```

OU

```
let books = ["meurtre en automne", "fleurs hivernales", "ecrevisse d'avril"];
let [book1, book2, book3] = books;
console.log(book2);
```

La déstructuration nous permet d'aller rapidement chercher plusieurs propriétés au sein de n'importe quel objet itérable sans avoir à multiplier les déclarations ou affectations.

Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

Rest parameter ...

Une fonction qui peut prendre des valeurs libres, non contenu dans un array
les valeurs sont prises et placées dans un array
Cette nouvelle façon d'écrire va transformer cet ensemble de nombres libres en array
Si on l'applique sur des valeurs libres, cela réunira les valeurs libres dans un array

Un opérateur qui permet de concaténer les objets d'un tableau :

```
let fruits = ["fraises", "pommes"];
let legumes = ["haricots", "poireaux"];
let cinqFruitLegume = [...fruits, "tomates", ...legumes];
console.log(cinqFruitLegume); //Array(5) [ "fraises", "pommes", "tomates", "haricots",
"poireaux"]
```


Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

Spread operator ...

Utilisation dans une fonction :

```
let mix = function(fruit1, fruit2){
    console.log(fruit1 + " se mixe avec " + fruit2);
}
let superFruits = ["bananes", "fraises"];
console.log(mix(... superFruits));   // bananes se mixe avec fraises
```

Utilisation dans le paramètre d'une fonction :

```
let bigMix = function (...superFruits){
    console.log(superFruits.join("-"));
}
bigMix("pommes", "poires");          // pommes-poires
```

L'opérateur SPREAD est l'inverse de l'opérateur REST.
Au lieu de prendre des valeurs libres et de les réunir dans un array, il prend un array et renvoie des valeurs libres.

Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

MAP

L'objet map est un nouveau type de structure de données qui permet de stocker sous forme d'un dictionnaire, une collection de clés / valeurs. N'importe quelle valeur valable en JS peut être utilisée comme clé ou comme valeur (string, number, booléen).

Création d'un objet Map vide :

```
let marques = new Map();
```

Création d'entrées avec différentes valeurs de différents types :

```
marques.set("renault", "clio");
marques.set("citroen", "c4");
marques.set("peugeot", "206");
```

Manipulation de la map

```
console.log(marques);
console.log(marques.size);
console.log(marques.has("renault");     // retourne un boolean
console.log(marques.get('citroen");     // retourne c4
marques.delete("peugeot");
marques.forEach((val, key) => {
        console.log(val);
        console.log(key);
});
```


Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

SET

L'objet Set est également un type de données qui représente une collection. Mais à la différence de Map, il n'y a pas de clé et il ne peut y avoir des doublons, les valeurs stockées doivent être uniques.

Création d'un objet set vide :

```
let marques = new Set();
```

Création d'entrées :

```
marques.add("renault");
marques.add ("peugeot");
```

Manipulation des données :

```
console.log(marques.has("renault"));          // renvoi true
console.log(marques.delete("renault");        // supprime renault du tableau
console.log(marques.clear();                  // supprime tous le tableau SET
for(let value of marques{
        console.log(value);
});
```

Alternative, transformer un tableau en set pour travailler avec des méthodes modernes :

```
let TabMarques = ["citroen", "renault", "bmw", "bmw", "audi",]
let marques = new Set(TabMarques);
```


Copyright- 2023 @ Audrey POEYDOMENGE

Programmation orientée objet

La programmation orientée objet en JavaScript est un paradigme de programmation qui permet de modéliser le monde réel à travers la création d'objets qui possèdent des propriétés et des méthodes.

En JavaScript, les objets sont des entités qui regroupent des données et des comportements liés. Les propriétés sont des variables qui stockent des valeurs, tandis que les méthodes sont des fonctions qui effectuent des actions sur ces valeurs.

Pour créer des objets en JavaScript, on peut utiliser la notation littérale, c'est-à-dire en définissant un objet sous la forme d'un ensemble de paires clé-valeur. On peut également créer des objets en utilisant des constructeurs, qui sont des fonctions qui créent des instances d'objets.

La programmation orientée objet en JavaScript permet de développer des applications plus modulaires, plus flexibles et plus faciles à maintenir. Elle permet de réutiliser du code en créant des classes qui peuvent être instanciées plusieurs fois avec des valeurs différentes, et elle permet également de créer des relations d'héritage entre les classes pour étendre leurs fonctionnalités.

Paradigme

Un paradigme est une manière de penser et de concevoir une solution à un problème donné. En programmation, un paradigme est une approche générale pour résoudre des problèmes en utilisant un ensemble de concepts et de techniques qui permettent d'organiser le code et de le rendre plus facile à comprendre, à maintenir et à évoluer.

Il existe plusieurs paradigmes de programmation, tels que la programmation impérative, la programmation fonctionnelle, la programmation orientée objet, la programmation événementielle, etc. Chacun de ces paradigmes se concentre sur une approche différente de la résolution de problèmes, avec des outils et des techniques spécifiques pour atteindre cet objectif.

En choisissant un paradigme de programmation particulier, un développeur peut mieux organiser son code en fonction de ses besoins et de la nature du problème à résoudre. Cela peut faciliter le développement, la maintenance et l'évolution de l'application en permettant de réutiliser du code et d'appliquer des solutions déjà éprouvées pour résoudre des problèmes similaires.

Les objets

Dans la vie réelle :

Un objet est un élément identifiable du monde réel, quelque chose de concret, un élément palpable que l'on peut manipuler selon des caractéristiques propres :
un livre - une voiture - un client - une facture

En programmation :

En informatique, un objet est avant tout une structure de données, également identifiable à un objet du monde réel sans que l'on puisse le palper :
un film (Allociné - un article (Ouest France)

Que ce soit dans la vie réelle ou sur le web, un objet se caractérise par :

- un état (les données de l'objet : sa composition) OU propriétés
- un comportement (opérations : ce qu'il sait faire) OU méthodes

Les caractéristiques d'un objets

En considérant un objet "voiture" :
Que ce soit dans la vie réelle ou en programmation (jeu vidéo) une voiture présente un certain nombre de caractéristiques.

Des propriétés
Une propriété est une donnée/variable associée à un objet et qui possède un nom et une valeur (chaîne de caractère, nombre, boolean, etc...). Accessible en écriture s'il est possible de modifier sa valeur :
nombre de portes
puissance moteur
couleur
prix
etc...

Des comportements
Les comportements sont des méthodes (fonctions) associées à un objet.
démarrer
accélérer
freiner
etc...

En langage programmatique, lorsque l'on parle de propriétés, on sous-entend des attributs, qui ne sont ni plus ni moins que de simples variables. Et lorsque l'on parle de comportement, on sous-entend des méthodes, qui elles sont de simples fonctions.

Objet littéral

En JavaScript, un objet littéral est une façon de créer un objet en utilisant une notation concise et facile à lire. Un objet littéral est défini en utilisant des accolades ({}) pour délimiter les propriétés et les valeurs associées à un objet.

Tout comme une variable ou une fonction, avant de pouvoir être utilisé, un objet doit être déclaré.
Déclaration d'un objet littéral :

```
let personne = {
        prenom : "Paul" ,
        age : 32 ,
        sePresenter : function(){
                return this.prenom + "a " + this.age + " ans" ;
}
```

Accéder aux propriétés d'un objet :

On accède aux attributs et méthodes d'un objet grâce à la notation pointée.

```
console.log(personne.prenom) ;              // affiche Paul dans la console du navigateur
console.log(personne.sePresenter()) ;
```

Ajout d'une propriété :

```
personne.metier = "Boulanger" ;
```

Les objets littéraux, quoique modifiables, sont gravés dans le marbre, pas d'instanciation possible !

L'objet object

En JavaScript, un élément Object est implémenté de base pour l'ensemble des objets.

let obj = { } ; équivaut à let obj = new Object( );

```
var p1 = new Object( );

p1.prenom="Jacques";
p1.nom="DUPONT";
p1.affiche = function( ){
      return this.nom" "+this.prenom;
    }
```

Tous les objets JS héritent de Object, et ils bénéficient tous des méthodes suivantes de Object :

- toString( ) qui permet de renvoyer une chaîne de caractère de l'objet
- valueOf( ) qui permet de renvoyer la valeur d'un objet

Mini TP

Sujet
Il s'agit ici de déclarer un objet de votre choix : voiture, livre, article, etc...
Assigner à votre objet des propriétés et méthodes lors de la déclaration de celui-ci.
Ne pas hésiter à jouer avec : modifier, ajouter...
Et bien sûr, faite un rendu dans le document.

Mini TP

Sujet
Il s'agit ici de déclarer un objet de votre choix : voiture, livre, article, etc...
Assigner à votre objet des propriétés et méthodes lors de la déclaration de celui-ci.
Ne pas hésiter à jouer avec : modifier, ajouter...
Et bien sûr, faite un rendu dans le document.

Correction

Déclaration : const voiture = { }; objet vide OU

```
const voiture = {
        marque : "BMW" ,
        couleur : "rouge" ,
        nbre Porte : 4,
        klaxonner: function( ) { console.log("BIP BIP");}
};
```

Une fois notre objet déclaré, nous pouvons l'utiliser :

```
ajout d'une propriété
voiture.annee  = 1995;
voiture.modele = "clio 2";
modification d'une propriété
voiture.marque = "clio 1";
console.log(voiture.couleur);
console.log(voiture.klaxonner());
```

Fonction constructeur

Une fonction constructeur est une fonction qui permet de créer des objets en JavaScript en utilisant le mot-clé new. Les fonctions constructeurs sont utilisées pour définir des classes d'objets qui peuvent ensuite être instanciées plusieurs fois pour créer des objets différents avec des valeurs différentes pour leurs propriétés.

En général, une fonction constructeur définit les propriétés et les méthodes qui sont communes à tous les objets d'une classe. Les propriétés sont définies à l'aide du mot-clé this pour se référer à l'objet en cours de création, tandis que les méthodes sont définies en utilisant la syntaxe de fonction normale.

```
function Personne(nom, prenom, age, profession) {
  this.nom = nom;
  this.prenom = prenom;
  this.age = age;
  this.profession = profession;
}

let personne1 = new Personne("Dupont", "Jean", 30, "Développeur");
let personne2 = new Personne("Martin", "Marie", 25, "Designer");
```

Programmation orientée prototype

En JS la programmation orientée objet (POO) est en réalité une programmation orientée prototype (POP).

Les objets sont des fonctions qui ont des propriétés et des méthodes.
Une de ces propriétés est le prototype.
Une fonction constructeur possède par défaut une propriété prototype et il est possible d'ajouter des propriétés et méthodes à un objet à partir de ce prototype.
Le prototype est une propriété associée à un objet pour modifier sa structure interne.

En JS, les objets sont passés par référence, c'est à dire qu'il y a un seul prototype pour tous les objets de la même fonction constructive.

```
function Livre(title,author){
        this.titre = title;
        this.auteur = author;
}

var livre1 = new Livre("Goodbye", "Jeannot");
var livre2 = new Livre("Hello", "Jeannette");

Livre.prototype.genre = "fiction";

console.log(livre1.genre);    affiche fiction
console.log(livre2.genre);    affiche fiction
```

on utilise l'opérateur instanceof pour tester la fonction constructrice d'un objet :

```
console.log(livre instanceof Livre?"oui":"non"));
```

Mini TP

Sujet

Il est temps de définir une fonction constructeur pour construire les objets de votre choix.

Et une fois de plus, parcourir ces objets pour un rendu dans le document.

Mini TP

Sujet

Il est temps de définir une fonction constructeur pour construire les objets de votre choix.

Et une fois de plus, parcourir ces objets pour un rendu dans le document.

Correction

```
function Personnage( ){
        this.nom = "Alphonse";
        this.arme = "épée";
}

var perso1 = new Personnage();
console.log(perso1.nom);
```

Ici nous déclarons et initialisons des valeurs par défaut (valeurs qui sont toutefois modifiables) alors que l'avantage d'utiliser une fonction pour construire des objets, c'est de permettre aux utilisateurs de pouvoir paramétrer eux-même les objets pour les utiliser.

```
function Personnage(p_nom, p_arme){
        this.nom = p_nom;
        this.arme = p_arme;
}

var perso1 = new Personnage("Alphonse", "épée");
```

Les objets natifs

LE LANGAGE DISPOSE D'OBJETS NATIFS QUI ONT DÉJÀ LEURS PROPRIÉTÉS ET MÉTHODES.

Les objets natifs

En JavaScript, un objet natif est un objet qui est intégré dans le langage JavaScript et qui est disponible par défaut sans nécessiter de chargement supplémentaire. Les objets natifs fournissent des fonctionnalités de base pour la manipulation de données, la gestion des erreurs, la manipulation de chaînes de caractères, la gestion des dates, les expressions régulières, les tableaux, etc.

Voici quelques exemples d'objets natifs en JavaScript :

- Object : C'est l'objet racine de toutes les classes d'objets en JavaScript. Il fournit des méthodes pour créer, initialiser et gérer des objets.
- Array : Cet objet est utilisé pour stocker une collection de valeurs dans un tableau. Il fournit des méthodes pour manipuler et accéder aux éléments du tableau.
- String : Cet objet est utilisé pour représenter des chaînes de caractères et fournit des méthodes pour manipuler ces chaînes, telles que la recherche de sous-chaînes, la concaténation et la conversion en majuscules ou minuscules.
- Number : Cet objet est utilisé pour représenter les nombres en JavaScript et fournit des méthodes pour effectuer des opérations arithmétiques et des conversions de format de nombre.
- Date : Cet objet est utilisé pour représenter les dates et les heures et fournit des méthodes pour manipuler les dates et les heures.

Les objets natifs en JavaScript sont très utiles pour effectuer des opérations courantes en programmation et sont disponibles dans tous les environnements JavaScript, y compris les navigateurs web et les serveurs Node.js.

String

Permet le traitement des chaînes de caractères.

Déclaration d'une chaîne de caractères

let nomVariable = new String("valeur") OU let nomVariable = "valeur";

```
let hello = "Hello";
console.log(hello);                          // String {"Bienvenue"}
let everybody = new String("everybody");     // Welcome
console.log(everybody);
```

Propriétés

.length Longueur de la chaîne de caractères

```
console.log(hello.length);// 5
```

Méthodes

.big() formate la chaîne avec la balise
.small() formate la chaîne avec la balise <small> </small>
.toUpperCase() transforme la chaîne en majuscule
.toLowerCase() transforme la chaîne en minuscule
.charAt() retourne le caractère situé à une certaine position
(attention le premier caractère est à la position 0)
.indexOf() retourne la position d'un caractère dans une chaîne
(attention le premier caractère est à la position 0)
.lastIndexOf() retourne la dernière position d'un caractère dans la chaîne
(si le caractère n'est pas présent dans la chaîne, renvoit -1)
.substring() retourne une portion de chaîne
.concat() concatène deux chaînes, équivalent au "+"
.link(url) formate la chaîne pour en faire un lien
.anchor(name) formate la chaîne pour en faire une ancre

```
console.log(hello.big());                    // <big>Hello</big>
console.log(hello.small());                  // <small> Hello </small>
console.log(hello.toUpperCase());            // HELLO
console.log(hello.toLowerCase());            // hello
console.log(hello.charAt(2));                // l
console.log(hello.indexOf("o"));             // 4
console.log(hello.lastIndexOf("l"));         // 3
console.log(hello.substr(2));                // "llo"
console.log(hello.substr(0,2));              // "he"
console.log(hello.concat(everybody));        // Helloeverybody
console.log(hello.link("http:....");         // <a href="http...">Hello</a>
console.log(hello.anchor("hello")            // <a name="hello">Hello</a>
```

Number

Permet le traitement des nombres.

Déclaration d'un nombre :

let nomVariable = new String("valeur") OU let nomVariable = "valeur";

Méthode	Description
number.isInteger()	Renvoie vrai si l'argument est un entier
number.isSafeInteger()	Renvoie vrai si l'argument est un entier sûr
nombre.parseFloat()	Convertit une chaîne en nombre
nombre.parseInt()	Convertit une chaîne en un nombre entier

Array

Un objet qui permet de créer et de manipuler des tableaux. Un tableau c'est une collection de valeurs, indexé à partir de 0.
Il possède des méthodes permettant d'ajouter, supprimer, d'extraire et de trier des éléments d'un tableau.

Déclaration d'un tableau

let nomTableau = new Array( ) OU let nomtableau = [ ];

```
// déclaration d'un tableau vide
let numbers = new Array();
console.log(numbers);                                                    // [ ]
// déclaration d'un tableau rempli
let books = new Array("tome1", "tome2", "tome3", "tome4", "tome5");
console.log(books);                                                      // (5) ["tome1", "tome2", "tome3", "tome4", "tome5"]

OU

// déclaration d'un tableau vide
let colors = [ ];
console.log(colors);                                                     // [ ]
// déclaration d'un tableau rempli
let movies = ["movie1", "movie2", "movie3", "movie4", "movie5"];
console.log(movies);                                                     // (5) ["movie1", "movie2", "movie3", "movie4", "movie5"];
```

Propriétés

length	nombre d'éléments dans un tableau	console.log(movies.length); // 5
constructor	cette propriété contient le constructeur de l'objet Array	
input	permet de faire une recherche dans le tableau à l'aide d'une regex	
prototype	permet d'ajouter des propriétés personnalisées à l'objet	

Array

.concat(tab1, tab2[, tab3, ...]) permet de concaténer plusieurs tableaux
créer un tableau à partir des différents tableaux passés en paramètre
.join() renvoie une chaîne de caractères contenant tous les éléments du tableau
converti un tableau en chaîne de caractères
.reverse() inverse l'ordre des éléments du tableau
.sort() permet de trier les éléments d'un tableau par ordre alphabétique
.slice(D, A) renvoie un tableau contenant une portion d'un tableau (extraction d'éléments)
(debut de données, fin de données sans l'inclure)
(debut de données jusqu'à la fin des éléments du tableau)

.unshift(valeur) permet d'ajouter un ou plusieurs élément au début d'un tableau
.push(valeur1[, valeur2, ...]) ajoute un ou plusieurs éléments à la fin d'un tableau
.shift() : Cette méthode supprime le premier élément du tableau
.pop() supprime le dernier élément du tableau
.splice() : Cette méthode ajoute/retire des éléments d'un tableau
filter() permet à partir d'un tableau initial d'effectuer une recherche pour obtenir un nouveau tableau

.unshift(valeur1[, valeur2, ...]) renvoie le code source qui a créé l'objet Array
.toString() renvoie la chaîne de caractères de l'instruction qui a créé l'objet Array
.valueOf retourne la valeur de l'objet Array auquel elle fait référence

Boolean

L'objet Boolean est un objet du noyau Javascript permettant de créer et de manipuler des valeurs de type booléennes.

La syntaxe à utiliser pour créer une variable booléenne :
let x = new Boolean(expression);
Le paramètre peut-être soit une valeur (True ou False) soit une expression

Dans le cas d'une expression, celle-ci est évaluée en tant que valeur booléenne.
Lorsqu' aucune valeur n'est passée, ou la valeur 0, ou une chaîne de caractères vide, ou null, ou undefined ou NaN, la valeur de l'objet est initialisée à False. Dans tous les autres cas, l'objet Boolean possédera la valeur True.

Les propriétés de l'objet Boolean :
.constructor : contient le constructor l'objet Boolean
.prototype : permet d'ajouter des propriétés personnalisées à l'objet

Les méthodes standards de l'objet Boolean :
toSource() : renvoie le code source qui a permis de créer l'objet Boolean
toString() : renvoie la chaîne de caractères correspond à l'instruction qui a permis de créer l'objet Boolean
valueOf : retourne tout simplement la valeur de l'objet Boolean auquel elle fait référence

Date

L'objet Date permet de travailler et manipuler toutes les variables qui concernent les dates et la gestion du temps.

Les dates en Javascript correspondent au nombre de millisecondes depuis le 1er janvier 1970.
Ainsi, toute date antérieure au 1er janvier 1970 fournit une valeur erronée.

Déclaration d'une date : let now = new Date();

Nom_de_l_objet = new Date(); // permet de stocker la date et l'heure actuelle
OU
Nom_de_l_objet = new Date("jour, mois date année heures:minutes:secondes") // les paramètres sont une chaîne de caractères
OU
Nom_de_l_objet = new Date(année, mois, jour) // les paramètres sont trois entiers séparés par des virgules, ceux omis sont mis à zéro par défaut
OU
Nom_de_l_objet = new Date(année, mois, jour, heures, minutes, secondes[, millisecondes])

Date

Les méthodes dont le nom commence par le radical get permettent de récupérer une partie de l'objet Date :

.getDay() retourne la valeur du jour de la semaine stockée dans la date, un entier 1 et 7 qui correspond au jour de la semaine
.getDate() retourne la valeur du jour du mois stockée dans la date, un entier entre 1 et 31 qui correspond au jour dans le mois
.getMonth() retourne le numéro du mois stocké dans la date, un entier entre 0 et 11 qui correspond au mois
.getFullYear() retourne la valeur de l'année sur 4 chiffres stockée dans la date, un entier qui correspond à l'année (ex : 2007)
.getHours() récupère la valeur de l'heure. L'objet retourné est un entier (entre 0 et 23).
.getMilliseconds() récupère le nombre de millisecondes. L'objet retourné est un entier (entre 0 et 999)
.getMinutes() récupère la valeur des minutes. L'objet retourné est un entier (entre 0 et 59) qui correspond aux minutes
.getSeconds() : Permet de récupérer le nombre de secondes. L'objet retourné est un entier (entre 0 et 59)
.getTime() : Permet de récupérer le nombre de millisecondes depuis le 1er janvier 1970. L'objet retourné est un entier.
Cette méthode est très utile pour convertir des dates, soustraire ou ajouter deux dates, etc.
.getTimezoneOffset() : Retourne la différence entre l'heure locale et l'heure GMT (Greenwich Mean Time).
L'objet retourné est un entier, il représente le nombre de minutes de décalage
.getYear() récupère la valeur de l'année sur 2 chiffres. L'objet retourné est un entier qui correspond à l'année ex :

Les méthodes dont le nom commence par le radical set permettent de définir une partie de l'objet Date :

.setDate() Remplace le jour stocké dans la date
.setFullYear() Remplace l'année stocké dans la date
.setMonth() Remplace le mois stocké dans la date
.toGMTString() Transforme la date avec la convention GMT

```
var today = new Date(1346454000); // ms since 01/01/1970
var today = new Date("September 1, 2012");
var today = new Date(2012, 8, 1); // attention janvier correspond à 0, .., décembre à 11.
```

Timer

Une fonction native de JavaScript.

Elle se décompose en deux partie:

- l'action à réaliser
- et le moment où l'action doit se déclencher (délai)

setTimeout(action, délai)

Math

Propriétés :

E Constante d'Euler (~2.718)
LN2 Logarithme népérien de 2 (~0.6931)
PI Pi (~3.14159)

Méthodes :

min()	renvoyer le plus petit nombre d'une série de nombres passés en arguments
max(),	renvoyer le plus grand nombre d'une série de nombres passés en arguments
floor(n)	arrondir la valeur passée en argument à l'entier immédiatement inférieur (ou égal) à cette valeur
ceil(n),	arrondir la valeur passée en argument à l'entier immédiatement supérieur (ou égal) à cette valeur
round(n)	arrondir la valeur passée en argument à l'entier le plus proche Si la valeur passée est supérieure à 0,5, la valeur sera arrondie à l'entier supérieur Dans le cas contraire, la valeur sera arrondie à l'entier inférieur Si la partie décimale vaut exactement 0,5, la valeur sera arrondie à l'entier supérieur
floor()	arrondir un nombre vers le bas à l'entier le plus proche
trunc(n)	ignorer la partie décimale d'un nombre et ne retourner que sa partie entière
abs(n)	retourne la valeur absolue d'un nombre (le nombre sans signe)
sin(), cos(), tan()	Retourne les sinus, cosinus, tangente d'un nombre
pow()	retourne un nombre à une certaine puissance
random()	retourne un nombre aléatoire entre 0 et 1 (exclu). On va ensuite pouvoir multiplier son résultat par un autre nombre afin d'obtenir un nombre pseudo-aléatoire compris dans l'intervalle de notre choix.
sqrt()	retourne la racine carrée d'un nombre

regexp

Les expressions régulières, ou "regex" en abrégé, sont des motifs de recherche de texte qui permettent de rechercher et de manipuler des chaînes de caractères selon certaines règles.
En JavaScript, les regex sont des objets qui peuvent être créés en utilisant le constructeur new RegEx(motif) ou en utilisant une notation littérale.

Déclaration d'un objet RegExp : let regex = /motif/drapeau OU let regex = new RegExp("motif","drapeau")

Un motif en regex (expression régulière) en JavaScript est une chaîne de caractères qui décrit un modèle de recherche. Il est généralement écrit entre deux barres obliques.

Les motifs en regex peuvent contenir différents caractères spéciaux qui permettent de définir des règles de recherche plus complexes, tels que :

- . : qui représente n'importe quel caractère
- * : qui représente zéro ou plusieurs occurrences du caractère précédent
- + : qui représente une ou plusieurs occurrences du caractère précédent
- ? : qui représente zéro ou une occurrence du caractère précédent
- \d : qui représente un chiffre
- \w : qui représente un caractère alphanumérique

Par exemple, le motif /a.*b/ correspondrait à une chaîne qui commence par la lettre "a" suivie de zéro ou plusieurs caractères de n'importe quel type, et se termine par la lettre "b". Ce motif pourrait correspondre à des chaînes telles que "abc", "axyzb", ou "a12b".

En JavaScript, vous pouvez utiliser la méthode test( ) de l'objet RegExp pour tester une chaîne de caractères par rapport à une expression régulière.

```
const regex = /a/;
const string = "hello world";
const isMatch = regex.test(string);
console.log(isMatch); // false
```

TP : traitement des formulaires

Sujet

Construisons un formulaire d'envoi et veillons à traiter les erreurs possible :

Le formulaire contient :
- un champ nom
- un champ age
- un champ pseudo

Merci donc de veiller à ce que :
- aucun champ ne puisse rester vide
- on ne puisse rentrer que des chiffres dans le champ age
- on ne puisse rentrer que des chiffres, lettres (maj, min), et les caractères spéciaux : $, _ , +

N'oubliez pas que HTML présente aussi des solutions pour éviter les erreurs, donc veillez à utiliser les deux de manière simultanée (pattern, required, min, max ...)

Copyright- 2023 @ Audrey POEYDOMENGE

Les objets du BOM

Greta-Cfa
Formation | Accompagnement | Apprentissage

Copyright- 2023 @ Audrey POEYDOMENGE

Browser Object Model - model objet du navigateur

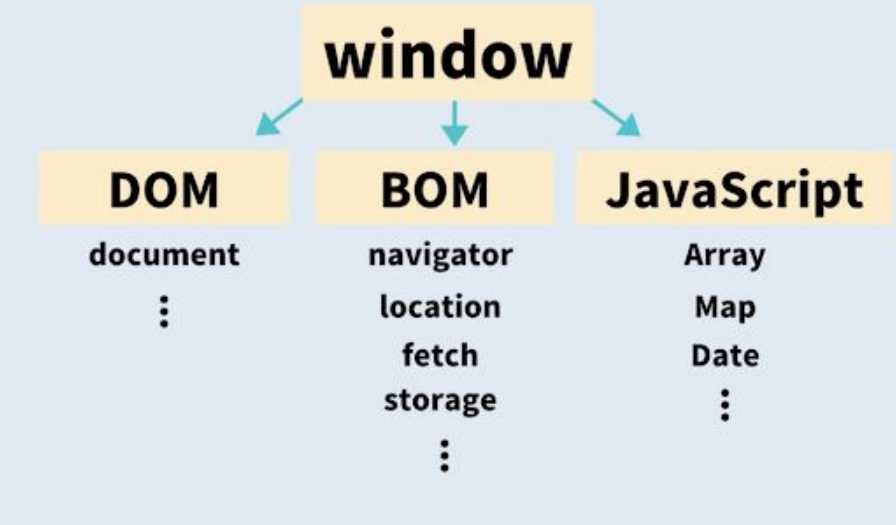
Lorsque l'on parle de JS exécuté dans le navigateur, il s'agit de JS exécuté via l'objet window. C'est donc l'objet le plus haut placé dans la hiérarchie. Cet objet représente la fenêtre du navigateur. Lorsque l'on crée des variables et fonctions, il s'agit de propriétés et méthodes de l'objet window. Et comme c'est l'objet racine, il est implicite et n'a pas besoin d'être spécifié pour utiliser ses méthodes ou ses propriétés.
alert( ) et isNaN ( ) sont des méthodes de l'objet window.

Dans cette fenêtre, il y a un document HTML : c'est l'objet document.
Donc, L'objet window est parent de l'objet document, mais aussi d'autres objets qui vont nous parler du navigateur lui-même.

```
let hello = "hello";
function heelo( ){ return hello; }

console.log(window);
affiche variable et fonction
```

... et affiche surtout toutes les propriétés et méthodes qu'il est possible d'utiliser !



Propriétés et méthodes de window

Des propriétés pour obtenir des informations sur la fenêtre du navigateur.

Des propriétés pour obtenir la taille de la fenêtre du navigateur :

innerWidth	renvoie la largeur (en pixel) interne de la fenêtre.
innerHeight	renvoie la hauteur (en pixel) interne de la fenêtre.
outerWidth	renvoie la largeur (en pixel) externe de la fenêtre.
outerHeight	renvoie la hauteur (en pixel) interne de la fenêtre.

La fenêtre du navigateur n'inclut pas les barres d'outils et les barres de défilement.

Méthodes de l'objet Window :

open( )	pour ouvrir une nouvelle fenêtre
close( )	pour fermer la fenêtre en cours
moveTo( )	pour déplacer la fenêtre en cours
resizeTo( )	pour redimensionner la fenêtre en cours

L'objet document

L'objet document représente le document HTML.

Le DOM est une représentation sous forme d'arbre du code source HTML.
Une sorte d'interface à notre code source.
C'est via le DOM que le JavaScript peut manipuler et modifier le code html et css d'une page web, et plus précisément grâce à l'objet Document.
L'objet Document permet donc d'accéder aux éléments de la page html.

Le window.document objet peut être écrit sans le préfixe window.

Propriétés et méthodes au chapitre DOM

L'objet location

L'objet location représente l'adresse de notre navigation (la barre d'adresse).
Cet objet peut être utilisé pour obtenir :
- l'adresse de page courante (URL)
- pour rediriger le navigateur vers une nouvelle page

Le window.location objet peut être écrit sans le préfixe window.

Propriétés de l'objet Location :

location.href	renvoie le href (URL) de la page en cours
location.hostname	renvoie le nom de domaine de l'hébergeur
location.pathname	renvoie le chemin et le nom de la page en cours
location.protocol	renvoie le protocole web utilisé (http: ou https :)
location.port	renvoie le numéro du port hôte internet de la page en cours

Méthode de l'objet Location :

location.assign( )	charge un nouveau document
location.reload( )	recharge/rafraichit la page
location.replace( )	permet de changer l'adresse dans le navigateur entre double quotes (ne permet pas de revenir en arrière)

L'objet screen

L'objet screen contient des informations sur l'écran de l'utilisateur.

Le window.screen objet peut être écrit sans le préfixe window.

Propriétés de l'objet Screen :

screen.width	retourne la largeur de l'écran en pixel
screen.height	retourne la hauteur de l'écran en pixel
screen.availWidth	retourne la largeur de l'écran en pixel - barre des tâches Windows
screen.availHeight	retourne la hauteur de l'écran en pixel, moins - barre des tâches Windows
screen.colorDepth	renvoie le nombre de bits utilisés pour afficher une couleur
screen.pixelDepth	retourne la profondeur de pixel de l'écran (échantillonnage d'écran)

Pour les ordinateurs modernes, profondeur de couleur et profondeur de pixel sont égaux.

8 bits =256 couleurs différentes « VGA »
16 bits = 65536 couleurs, résolution différente « haute Couleurs »
24 bits = 16777216 couleurs, différents « True Colors »

Mini TP

Sujet

L'objet BOM possède d'autres objets enfants tels que :

History
Navigator
...

A vous d'en chercher l'utilité, leurs propriétés et méthodes, et des cas d'utilisations concrets pour chacun.

Les objets du DOM

Noeud

Selon le W3C, tout ce qui est contenu dans un document HTML est un nœud :

- le document entier est un noeud de document
- chaque élément html est un nœud élément qui peut contenir d'autres noeuds (enfants)
- le texte à l'intérieur des éléments HTML est un nœud textuel (pas d'enfant)
- chaque attribut HTML est un nœud d'attribut (obsolète)
- tous les commentaires sont des nœuds de commentaires

Le DOM est une interface Node, c'est à dire une boîte à outils pour parcourir les différents niveaux de l'arborescence.
On assure le déplacement dans l'arbre grâce à des propriétés et méthodes.

Tous les nœuds sont accessibles en JS et on peut naviguer dans l'arborescence de noeuds pour :

- modifier des noeuds
- créer de nouveaux nœuds
- supprimer des noeuds

Copyright- 2023 @ Audrey POEYDOMENGE

Propriétés de noeud

nodeName retourne le nom du noeud

nodeName d'un noeud d'élément	retourne le nom du noeud (identique à la balise)
nodeName d'un attribut node	retourne le nom de l'attribut
nodeName d'un noeud de texte	est toujours #text
nodeName du noeud de document	est toujours #document

nodeValue retourne la valeur contenue dans le noeud

nodeValue pour les nœuds d'élément	n'est pas défini
nodeValue pour les nœuds de texte	est le texte lui-même
nodeValue pour les nœuds d'attribut	est la valeur d'attribut

nodeType retourne le type d'un noeud

Le type d'un noeud peut être :

ELEMENT_NODE	balise : <h1 class="heading">W3Schools</h1>
ATTRIBUTE_NODE	attribut : class = "heading" (deprecated)
TEXT_NODE	noeud textuel : W3Schools
COMMENT_NODE 8	<!-- This is a comment -->
DOCUMENT_NODE 9	The HTML document itself (the parent of <html>)
DOCUMENT_TYPE_NODE 10	<!Doctype html>

childNodes permet de lister les noeuds enfants d'un élément sous forme d'un tableau

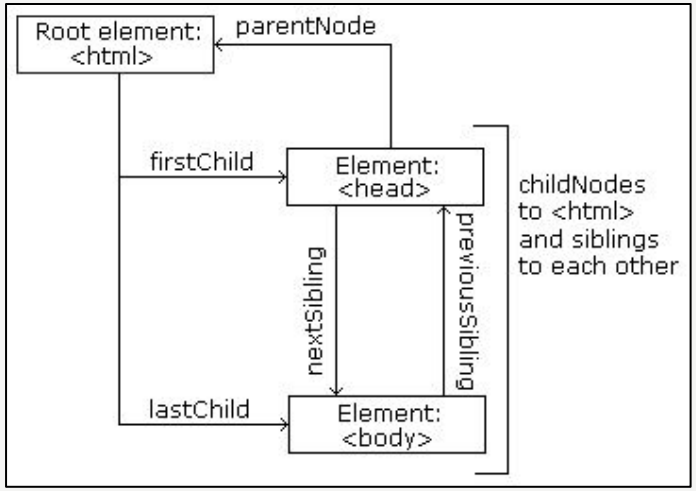
Copyright- 2023 @ Audrey POEYDOMENGE

Hiérarchie des noeuds

Les nœuds d'une arborescence HTML ont des relations hiérarchiques entre eux : parent, enfant et frère.

- le nœud supérieur est appelé racine (ou nœud racine)
- chaque nœud a exactement un parent, sauf la racine (qui n'a pas de parent)
- un nœud peut avoir plusieurs enfants
- les frères et sœurs sont des nœuds avec le même parent

<html> est le noeud racine, il n'a pas de parent
<html> est le parent de <head> et <body>
<head> est le premier enfant de <html>
<head> a un enfant qui est <title>
<title> a un enfant (un nœud de texte) "Tutoriel DOM"
<head> et <body> sont frères
<body> est le dernier enfant de <html>
<body> a deux enfants <a> et <h1>
<a> a un enfant "My link"
<h1> a un enfant "My header"
<a> et <h1> sont frères



Hiérarchiser les éléments du DOM c'est tout simplement les cibler afin de pouvoir les modifier, en ajouter, ou en supprimer

Copyright- 2023 @ Audrey POEYDOMENGE

Des propriétés pour naviguer entre les noeuds

Pour naviguer entre les noeuds avec JS, on utilise les propriétés de nœud suivantes :

parentNode	pour accéder au noeud parent d'un élément
childNodes[*nodenumber*]	pour accéder aux noeuds enfants d'un élément
firstChild	pour accéder au premier enfant d'un élément
lastChild	pour accéder au dernier enfant d'un élément
nextSibling	pour accéder au frère suivant d'un élément
previousSibling	pour accéder au frère précédent d'un élément

Parcourir l'arborescence : document.childNodes[0].childNodes[1];

Il existe une autre manière de cibler les éléments HTML, grâce aux propriétés de hiérarchie des éléments. Il ne s'agit plus de partir de la racine de la page (document) mais depuis un élément. Chaque élément est un objet JS avec ses propres propriétés et méthodes.

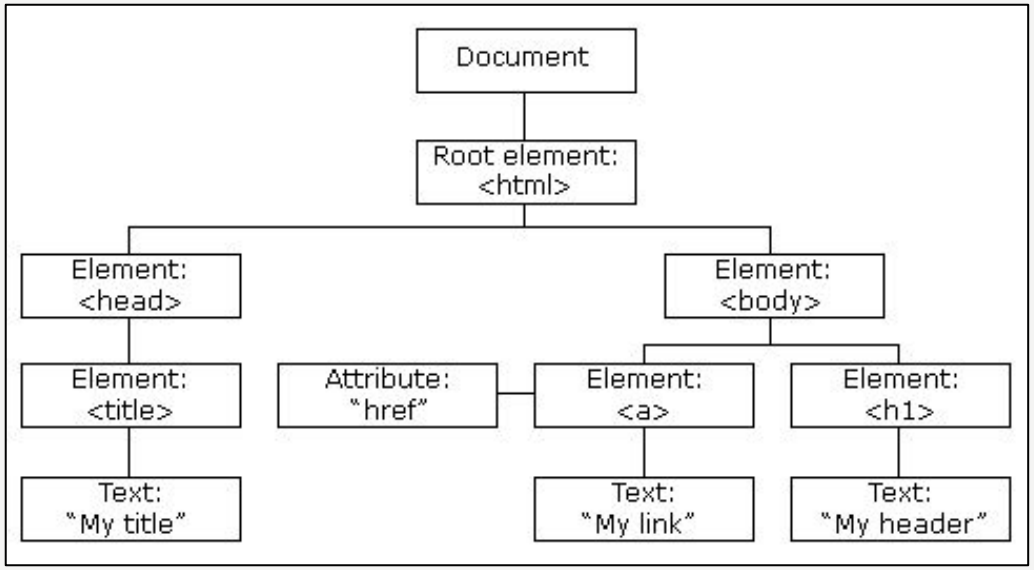
Il existe des propriétés pour parcourir le parent et les enfants de chaque élément.

children	retourne la liste des enfants de l'élément
parentElement	retourne l'élément parent de l'élément
firstElementChild	permet d'accéder au premier enfant d'un noeud
lastElementChild	permet d'accéder au dernier enfant d'un noeud
nextElementSibling	permet de naviguer vers l'élément suivant (frere)
prevElementSibling	permet de naviguer vers l'élément précédent (frere)

Copyright- 2023 @ Audrey POEYDOMENGE

Document Object Model

```
<html>
	<head>
		<title>My title</title>
	</head>
	<body>
		<a href="#">My link</a>
		<h1>My header</h1>
	</body>
</html>
```



Copyright- 2023 @ Audrey POEYDOMENGE

Document Object Model

Lorsqu'une page Web est chargée, le navigateur crée un modèle de document pour cette page. Ce modèle est construit comme une arborescence d'objets, une représentation sous forme d'arbre du code source HTML.

Le DOM est une interface à notre code source, un moyen standardisé (W3C, dernière version publiée en 2004) qui permet de :
- représenter une page HTML sous la forme d'un arbre constitué d'objets nœuds (node)
- d'accéder aux éléments de la page
- de manipuler ces éléments (interaction avec une page html).
- de créer/modifier/supprimer des éléments HTML, leurs attributs, styles et contenu

L'objet document représente la page Web, cet objet est le propriétaire de tous les autres objets présents sur cette page. Pour interagir avec une page HTML, on le fait via l'objet document.

Cibler et atteindre les éléments du DOM c'est en fait cibler du code html.
Puis on utilise des propriétés et méthodes pour interagir avec ce code html.
Les propriétés et méthodes sont visibles en console lorsque l'on cible un élément.

Une propriété est une valeur d'élément HTML que l'on peut obtenir, définir et modifier.
Une méthode est une action que l'on peut effectuer sur un élément HTML.
Tout commence avec le document, point de départ du DOM (balise <html>).

Copyright- 2023 @ Audrey POEYDOMENGE

Propriétés de l'objet document

Des propriétés pour récupérer/cibler des valeurs d'éléments HTML :

document.doctype	retourne le doctype du document
document.documentElement	retourne l'élément <html> et tout son contenu
document.head	retourne l'élément <head>
document.title	retourne l'élément <title>
document.scripts	retourne tous les éléments <script>
document.body	retourne l'élément <body> et tout son contenu
document.forms	retourne tous les éléments <form>
document.anchors	retourne tous les éléments <a> qui ont un attribut name
document.links	retourne tous les éléments <area> et <a> qui ont un href
document.images	retourne tous les éléments <img>
document.embeds	retourne tous les éléments <embed>
document.applets	retourne tous les éléments <applet> (déprécié en HTML5)

Copyright- 2023 @ Audrey POEYDOMENGE

Propriétés de l'objet document

Des propriétés pour obtenir des informations sur l'adresse du document :

document.domain	retourne le nom de domaine du serveur
document.URL	retourne l'URL complète du document
document.baseURI	retourne l 'URI absolue de base du document
document.documentURI	retourne l'URI du document
document.inputEncoding	retourne l'encodage du document (jeu de caractères)

Des propriétés pour obtenir des informations autres :

document.cookie	retourne le cookie du document
document.documentMode	retourne le mode utilisé par le navigateur
document.domConfig	retourne la configuration du DOM (obsolète)
document.implementation	retourne l'implémentation DOM
document.lastModified	retourne la date et l'heure de mise à jour du document
document.readyState	retourne le statut de chargement du document
document.referrer	retourne l'URI du référent (document de liaison)
document.strictErrorChecking	retourne si la vérification d'erreur est appliquée

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

A partir des éléments suivants :

```
<form name="contactForm">
    <input type="text" name="name-box" id="name-box-id" />
</form>
```

Tenter d'accéder aux deux éléments en utilisant des propriétés de l'objet document.

Correction

```
document.forms[0]
document.forms["contactForm"]
document.forms.contactForm
document.contactForm
document.forms.contactForm.elements[0]
document.forms.contactForm.elements["nameBox"]
document.forms.contactForm.nameBox
document.contactForm.nameBox
```


Copyright- 2023 @ Audrey POEYDOMENGE

Méthodes de l'objet document

Il existe une manière plus simple pour sélectionner/récupérer des éléments sans utiliser les propriétés hiérarchiques de nœuds ou d'éléments.

document.getElementById(id)	retourne un élément selon son id
document.getElementsByName(name)	retourne un ou plusieurs éléments selon l'attribut name
document.getElementsByTagName(tag)	retourne un ou plusieurs éléments selon le tag (balise)
document.getElementsByClassName(classe)	retourne un ou plusieurs éléments selon la classe
document.querySelector(sélecteur CSS)	retourne le 1er élément selon un sélecteur CSS
document.querySelectorAll(sélecteur CSS)	retourne tous les éléments selon un sélecteur CSS

Rappel sur les sélecteurs CSS
On a déjà une façon pratique de sélectionner nos éléments en CSS lorsqu'on cherche à leur appliquer un style : article.second p#important { }
On peut aussi utiliser les sélecteurs CSS pour sélectionner les éléments du DOM en Javascript
document.querySelector("article.second p#important"); // sélectionne le premier élément
document.querySelectorAll(".rouge"); // sélectionne tous les éléments et retourne un tableau

Depuis la console du navigateur, on peut voir les nombreuses méthodes et propriétés auxquelles il nous est possible d'accéder.

Pour plus de simplicité, avant d'agir sur les éléments, on peut les récupérer dans des variables.
var titre = document.getElementById("title");
title.textContent = "Welcome everybody";

Copyright- 2023 @ Audrey POEYDOMENGE

Mini-TP

Sujet

Réaliser une page HTML avec différents éléments :

header
    title#title
section
    p*3.para
footer
    small.copyright

Récupérer ces éléments dans des variables puis afficher ces variables dans la console.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini-TP

Sujet

Réaliser une page HTML avec différents éléments :

header
    title#title
section
    p*3.para
footer
    small.copyright

Récupérer ces éléments dans des variables puis afficher ces variables dans la console.

Correction

```
let entete = document.getElementsByTagName("header");
let titre = document.getElementById("title");
let para = document.getElementsByClassName("para");
let pied = document.getElementsByTagName("footer");
let copy = document.querySelector(".copyright");

console.log(entete, titre, para, pied, copy);
```


Copyright- 2023 @ Audrey POEYDOMENGE

Modifier les contenus

```
<div id="balise-div">
        <p>du texte</p>
</div>
```

```
let baliseDive=document.getElementById("balise-div") ;
```

Modifier un contenu HTML

La propriété innerHTML permet de récupérer le HTML enfant d'un élément sous forme de texte, elle retourne des balises :

```
alert(baliseDive.innerHTML) ;
```

Cette propriété permet également de modifier du HTML :

```
baliseDive.innerHTML="<blockquote>blabla</blockquote>" ;   // paragraphe remplacé par blockquote
                                                            visuellement dans le navigateur, pas dans le code !
```

Pour ajouter au code déjà présent :

```
baliseDiv..innerHTML += "<strong> et du texte en plus</strong>" ;
```

Modifier un contenu textuel

La propriété innerText permet de récupérer ou modifier le contenu textuel d'un élément HTML.

Une méthode trés employée est de créer un élément vide, puis de le remplir grâce à ces deux propriétés.

Copyright- 2023 @ Audrey POEYDOMENGE

Modifier les styles CSS

Les éléments du DOM disposent d'une propriété style qui renvoit un objet. Les propriétés de cet objet correspondent aux propriétés css de l'élément. En définissant leurs valeurs depuis JS, on modifie le style de l'élément.

```
<h1 id="title">Mon super titre</h1>
```

```
let title = document.getElementById("title");
```

Récupérer le style d'un élément : element.style.propriété

```
console.log(title.style.color); // affiche une couleur ou rien
```

La propriété style utilisée en javaScript représente l'attribut style interne à l'élément HTML.
La propriété style ne permet pas d'accéder aux déclarations de style situées ailleurs (feuilles.css).

Modifier le style d'un élément : element.style.property = nouveau style

```
title.style.margin = "5px";
```

Certaines propriétés s'écrivent sous la forme d'un nom composé. Pour utiliser ces propriétés en JS, il faut supprimer le tiret et écrire la propriété en camelCase : backgroundColor

La méthode getComputedStyle( ) permet de récupérer la valeur de n'importe quel style css qu'il soit déclaré dans l'attribut style ou dans une feuille de style.
Attention, certaines propriétés ne sont pas récupérables avec cette méthode : offsetWidth (width + padding + border) - offsetHeight - offsetLeft - offsetTop - offsetParent

Copyright- 2023 @ Audrey POEYDOMENGE

Modifier les classes

Une autre façon de modifier le style de nos éléments c'est tout simplement de jouer avec leurs classes CSS.

Texte avec Un lien let lien = document.querySelector("a");

On peut définir une classe pour un élément grâce à la propriété className :

lien.className = "link";

classList	pour accéder à la liste des classes d'un élément

à partir de cette propriété on peut se servir de méthodes pour ajouter/supprimer des classes

classList.add("rouge", "vert")	ajoute 2 classes (rouge et vert) à un élément
classList.contains("rouge")	pour savoir si l'élément possède la classe rouge
classList.remove("rouge")	retire la classe rouge de l'élément
classList.replace("vert", "yellow")	remplace la classe vert par la classe yellow

Copyright- 2023 @ Audrey POEYDOMENGE

Modifier les attributs

Un attribut est une propriété supplémentaire ajoutée à un élément, ex : id, class, href, selected, etc...

lien
let link=document.getElementById("my-link") ;

attributes	permet d'accéder à la liste des attributs d'un élément HTML
setAttribute (attribut, valeur)	permet de définir un nouvel attribut :

link.setAttribute("href", "http://amazone.fr");

getAttribute(attribut)	permet de récupérer un attribut
removeAttribute(attribut)	permet de supprimer un attribut à un élément

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Soit les éléments suivant :

Lorem Ipsum Dolor
Lorem Ipsum Dolor

Spécifier 4 méthodes différentes de ciblage pour récupérer le premier élément.

Puis en modifier le contenu textuel.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Soit les éléments suivant :

Lorem Ipsum Dolor
Lorem Ipsum Dolor

Spécifier 4 méthodes différentes de ciblage pour récupérer le premier élément.

Puis en modifier le contenu textuel.

Correction

document.getElementsByTagName("p")[0]
document.getElementById("first-para")
document.getElementsByClassName("para"[0]
document.querySelector("p#first-para)

.innerText = "blablabla";

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Soit les éléments suivant :

Lorem Ipsum Dolor
Chocolat café thé

Récupérer le contenu du premier élément et le copier dans le second élément.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Soit les éléments suivant :

Lorem Ipsum Dolor
Chocolat café thé

Récupérer le contenu du premier élément et le copier dans le second élément.

Correction

```
let first = document.getElementById("first-para");
let second = document.getElementById("second-para");

second.innerText = first.innerText;
```


Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Soit l'élément suivant :

Lorem Ipsum Dolor

Choisir la méthode de ciblage de votre choix pour récupérer l'élément.

Puis en modifier les styles de votre choix.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Soit l'élément suivant :

Lorem Ipsum Dolor

Choisir la méthode de ciblage de votre choix pour récupérer l'élément.

Puis en modifier les styles de votre choix.

Correction

```
let firstPara = document.getElementById("first-para");

firstPara.style.color = "silver";
firstPara.style.backgroundColor = "yellow";
firstPara.style.fontSize = "2rem";
```


Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Soit les éléments suivant :

```
<img id="img" src="smiley.jpg" />
<input type="text" id="name" value="name" />
```

Choisir la méthode de ciblage de votre choix pour récupérer les deux éléments.

Puis en modifier les attributs de votre choix.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Soit les éléments suivant :

<input type="text" id="name" value="name" />

Choisir la méthode de ciblage de votre choix pour récupérer les deux éléments.

Puis en modifier les attributs de votre choix.

Correction

document.getElementById("img").src = "sun.jpg";

document.querySelector("#name").value = "prenom";

Copyright- 2023 @ Audrey POEYDOMENGE

Créer des éléments HTML

Pour créer un nouvel élément HTML on utilise la méthode createElement(tag)
on place cette élément dans une variable pour pouvoir manipuler l'élément plus facilement :
let child = document.createElement("p");

On ajoute un contenu textuel à cet élément :
child.innerText = "Mon super paragraphe";

On apporte du style à cet élément :
child.style.color = "green";

Un élément créé ne fait pas encore partie du document, il n'est pas visible sur la page. Pour le voir il faut l'ajouter en tant qu'élément enfant d'un élément parent grâce à la méthode appendChild(variable) :
father.appendChild(child);

D'autres méthodes de l'objet document :

document.createElement(tag)	*permet la création d'un élément HTML*
document.removeChild(variable)	*permet la suppression d'un élément HTML*
document.appendChild(variable)	*permet l'ajout d'un élément HTML*
document.replaceChild(new, old)	*permet le remplacement d'un élément HTML*

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Créer de toutes pièces une liste avec du contenu textuel.

Libre à vous d'ajouter des attributs.

Surtout ne pas hésiter à apporter du style à cette liste.

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

Créer de toutes pièces une liste avec du contenu textuel.

Libre à vous d'ajouter des attributs.

Surtout ne pas hésiter à apporter du style à cette liste.

Correction

```
const itemContainer = document.createElement('ul');
itemContainer.classList.add('item-container');
const itemList = document.createElement('li');
item.classList.add('item-list");
itemList.innerText = "Chaussettes éléphant";
itemContainer.appendChild(itemList);
```


Copyright- 2023 @ Audrey POEYDOMENGE

ES 6

Template string : les backtick ``

Les templates permettent de créer des chaînes de caractères sur plusieurs lignes :

Sans template :

```
const description2 = "Lorem ipsum dolor sit amet consectetur adipisicing elit.
                      Deserunt exercitationem delectus soluta voluptas vero adipisci
                      mollitia tempora perferendis,omnis modi dolorem cum non ullam maxime
                      nulla at! Vel, ipsam inventore!";
// Uncaught SyntaxError: "" string literal contains an unescaped line break
```

Avec template :

```
const description2 = `Lorem ipsum dolor sit amet consectetur adipisicing elit.
                      Deserunt exercitationem delectus soluta voluptas vero adipisci
                      mollitia tempora perferendis,omnis modi dolorem cum non ullam maxime
                      nulla at! Vel, ipsam inventore!`;
```

Mais surtout, ils permettent de faire de l'interpolation de variable !

```
const footer = document.getElementById('footer');
const copyright = "Pierre Paul Jacques";
footer.innerHTML = `<small>${copyright}</small>`;      // ${ variable }
```

Greta-Cfa
Formation | Accompagnement | Apprentissage

Copyright- 2023 @ Audrey POEYDOMENGE

TP

Sujet

Vous devez créer une liste d'articles et les insérer dans le document.

Pour chaque article il nous faut :

- titre
- courte description
- prix
- disponibilité

Copyright- 2023 @ Audrey POEYDOMENGE

TP

Sujet

Vous devez créer une liste d'articles et les insérer dans le document.

Pour chaque article il nous faut :

- titre
- courte description
- prix
- disponibilité

Correction

```
let articlesSection = document.querySelector(".articles-section");

let title = "Super titre";
let description = "lorem ipsum description";
let price = "40$";

let article = `
        <article class="article-container">
                <h1 class="article-title">${title}</h1>
                <p class="article-description">${description}<span class="article-price">${price}</span></p>
        </article>
`;

articlesSection.innerHTML = article;
```


Copyright- 2023 @ Audrey POEYDOMENGE

Les évènements

Greta-Cfa
Formation | Accompagnement | Apprentissage

Copyright- 2023 @ Audrey POEYDOMENGE

Fonctionnement

Le DOM permet d'exécuter du code JavaScript lorsqu'un événement se produit.
Le DOM permet à JavaScript de réagir aux événements HTML.

Une fonction JS peut être exécutée lorsqu'un événement se produit, comme lorsqu'un utilisateur clique sur un élément HTML.

Pour exécuter du code lorsqu'un utilisateur clique sur un élément, soit on ajoute du code JS à un attribut d'évènement HTML, soit on crée un écouteur d'événement sur un élément.

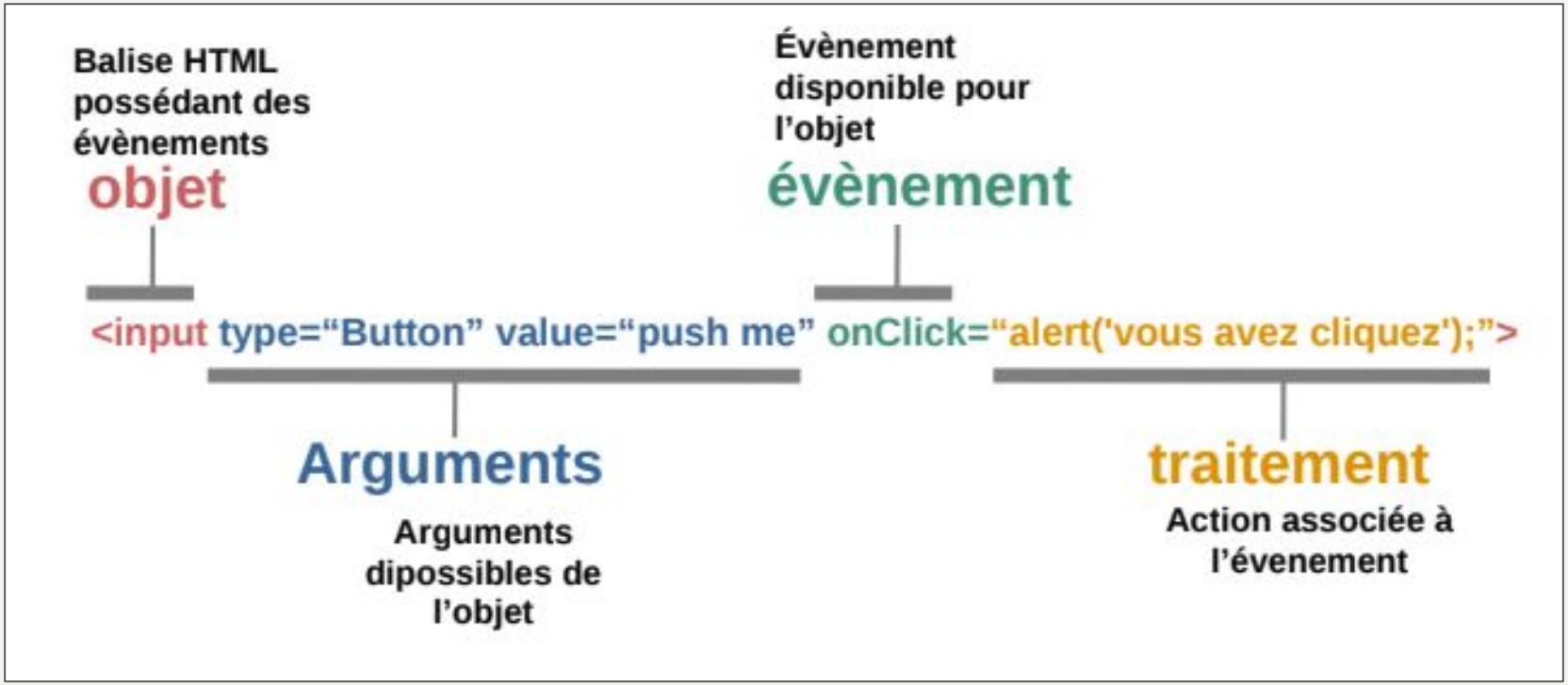
Les événements sont générés par le navigateur quand il se passe des choses avec des éléments HTML :
- un élément est cliqué
- la page a chargé
- les champs de saisie sont modifiés
- la souris passe sur un élément
- un formulaire HTML est soumis
- une touche est enfoncée

Un événement permet de déclencher des traitements (fonctions).

Copyright- 2023 @ Audrey POEYDOMENGE

Attribut d'évènement

Utilisation directement au sein des balise HTML (sans utiliser le DOM) :
attribut on + nom de l'événement



Copyright- 2023 @ Audrey POEYDOMENGE

Écouteur d'événement Dans un fichier.js

Avec le DOM-0

```
var element = document.getElementById('title');

element.onclick = function(){  // Une variable qui prend en propriété une fonction anonyme
        alert('Vous venez de cliquer sur le lien');
}
```

Avec le DOM-2

```
element.addEventListener('click', function(){ // Une fonction anonyme propre à un élément, non réutilisable ailleurs car non identifiée
        alert('Vous venez de cliquer sur le lien');
}
```

ON AJOUTE UN GESTIONNAIRE D'ÉVÉNEMENT SUR UN ÉLÉMENT

La méthode .addEventListener() prend 3 paramètres :
- le nom de l'événement (click, dragstart, keydown...)
- la fonction de retour à exécuter
- (option) un booléen pour spécifier si l'on utilise la phase de capture ou celle de bouillonnement

Enfin, nous pouvons supprimer la gestion d'un événement en faisant appel à la méthode removeEventListener avec les mêmes paramètres utilisés lors de l'ajout.

Copyright- 2023 @ Audrey POEYDOMENGE

Écouteur d'événement Dans un fichier.js

Attention :

* Par défaut un événement se propage, c'est à dire qu'il se transmet à l'élément parent jusqu'à l'élément racine.

stopPropagation( ) est une méthode de l'objet reçu en paramètre de la fonction de callback et qui permet d'empêcher la propagation vers le parent.

* Par défaut, le comportement d'un élément actionné est exécuté.
Si l'on clique sur un lien, le navigateur va nous rediriger vers l'url en question.
Si l'on clique sur un bouton de validation de formulaire, celui-ci sera envoyé.

Il est possible de désactiver ce comportement grâce à la méthode preventDefault() que l'on applique sur un objet que l'on passe en paramètre à la fonction et qui correspond au contenu de l'événement qui vient de se produire.

```
links.addEventListener("click", function(e){
        e.preventDefault();
        e.stopPropagation();
        // autres instructions
})
```


Copyright- 2023 @ Audrey POEYDOMENGE

Des évènements

Événement	Déclenchement	S'applique à
Evénements souris		
onclick	clic de la souris	objets JS (link, document, form) - balises HTML (a, body, area)
ondblclick	double clic	objets JS (link, document, area) - balises HTML (a, body, area)
onmousedown	clic de la souris	objet JS (link, document, form) - balises HTML (a, body, form)
onmouseup	relâchement du clic	objets JS (document, link, form) - balises HTML (body, a, input)
onmouseover	passage de la souris	objets JS (link, area) - balises HTML (a, area)
onmousemove	le curseur passe dessus	objets JS (link, document, form) - balises HTML (a, body, form)
onmouseout	à la sortie de la souris	objets JS (link, area) - balises HTML (a, area)
Evénements clavier		
onkeydown	touche enfoncée	objets JS (document, link, image, form) - balises (body, a, img, textarea)
onkeyup	touche relâchée	objets JS (document, link, image, form) - balises (body, a, img, textarea)
onkeypress	touche pressée	objets JS (document, link, image, form) - balises (body, a, img, textarea)

Copyright- 2023 @ Audrey POEYDOMENGE

Des évènements

Événement	Déclenchement	S'applique à
Evénements page / fenêtre		
onload	chargement de la page HTML	objets JS (image, window) - balises HTML(img, body)
onunload	fermeture de la page HTML	objets JS (window) - balises HTML (body)
onmove	déplacement de la fenêtre	objets JS (window) - balises HTML (body)
onresize	redimensionne	objets JS (window) - balises HTML (body)
ondragdrop	déplacer quelque chose	objets JS (window) - balises HTML (body)
onabort	L'utilisateur a stoppé le chargemt img	objets JS (image) - balises HTML (img)
onerror	erreur de script	objets JS (window, image) - balises HTML (body, img)
Evénements formulaire		
onfocus	Au focus de l'élément	objets JS (window, form) - balises HTML (body, input)
onblur	A la perte du focus	objets JS (window, form) - balises HTML (body, input, textarea, select)
onselect	Quand on sélectionne	objets JS (form) - balises HTML (input, textarea)
onchange	Au changement de la valeur	objets JS (form) - balises HTML (input, textarea, select)
onsubmit	A l'envoi du formulaire	objets JS (form) - balises HTML (form)
onreset	Quand on réinitialise	objets JS (form) - balises HTML (form)

Copyright- 2023 @ Audrey POEYDOMENGE

L'objet event

Lorsqu'un événement se déclenche, la fonction de callback peut prendre en paramètre un objet event qui va contenir des infos sur l'événement lui-même.

Chaque événement implémente l'objet Event, c'est à dire que chaque événement a au minimum les mêmes fonctions et propriétés que l'objet Event :
.preventDefault()
.stopPropagation()

Grâce à cet objet event, il va être possible de récupérer la position de la souris, écouter le clavier, etc...

Pour détecter le mouvement souris par exemple, on écoute l'événement mousemove.
Cet élément va fournir un objet de type MouseEvent avec les propriétés:
clientY et clientX
pageX et pageY
offsetX et offsetY
screenX et screenY
moventX et moventY

Copyright- 2023 @ Audrey POEYDOMENGE

Mini TP

Sujet

créer plusieurs élément HTML :
titre
bouton
lien

récupérer ces éléments dans des variables puis placé des gestionnaires d'événements sur ces éléments (attribut d'évènement ou directement dans le DOM).

Enfin, créer des fonctions de callback afin de modifier les contenus de ces éléments, qui seront déclenchées lors d'événement de votre choix.

Copyright- 2023 @ Audrey POEYDOMENGE