Fonction (1/4)

- Une fonction est un ensemble d'instructions (ou morceau de code) qui réalise une tâche spécifique une fois appelée
- Dans un script, on appelle une fonction par son nom nomFonction()
- Il existe deux types de fonctions:
  - Prédéfinies ou natives
  - Personalisées: définies par l'utilisateur

Fonction prédéfinie (2/4)

- parseInt(): convertit une chaine de caractères en nombre entier
- parseFloat(): convertit une chaîne de caractères en nombre flottant
- alert(): affiche le texte passé en argument et un bouton 'ok' dans une boite de dialogue
- console.log(): affiche le texte passé en argument dans la console du navigateur
- prompt(): affiche une boite de dialogue avec un message et un champ de saisie
- etc

Fonction personnalisée (3/4)

- Une fonction doit être définie avant son utilisation
- La définition d'une fonction se fait comme suit:

```
function nom(paramètres) {
                    //Ensemble d'instructions
                    }; //instruction
```

Ex.
```
function somme(a, b){
                    return a+b;
                    }
```

Fonction personnalisée (4/4)

- Une fonction peut aussi être définie comme suit:

```
var nom = function (paramètres){ bloc
d'instructions} // expression de fonction
```

```
Ex. var somme=function (a, b){
                    return a+b;
                }
```

Tableaux (1/5)

- Un tableau est un objet contenant un ensemble d'éléments ou valeurs.
- Les éléments d'un tableau peuvent être de différents types ie nombres, chaines de caractères, objets, etc
- Déclaration et initialisation

```
Ex. let tab=[ ]; // tableaux vide
    let tab=[3,4,5]; // tableau à 3 éléments
    let tab=Array.of(3,4,5);
```

Tableaux (2/5)

- L'accès aux éléments d'un tableau se fait en utilisant la notation crochet
- Ex. let Tab=[3,4,5];

  Tab[0]=3, Tab[1]=4, Tab[2]=5, Tab[3] // undefined
- La taille d'un tableau s'obtient avec la propriété 'length'
- Ex. let Tab=[3,4,5];

  let taille=Tab.length // taille=3
- Un tableau est un **objet** de type **Array**
  - Typeof(Tab) => object
  - Array.isArray(Tab)=> true

Tableaux (3/5)

❑ Quelques méthodes

- push(): permet **d'ajouter** des éléments à la **fin** d'un tableau

Ex. let Tab=[3,4,5];

let x=Tab.push(6,7); / / Tab=[3,4,5,6,7], x=5 (Taille)

- pop(): permet de **supprimer** le **dernier** élément d'un tableau

Ex. let Tab=[3,4,5];

let x=Tab.pop() / / Tab=[3,4], x=5

Tableaux (4/5)

- unshift(): permet **d'ajouter** des éléments au **début** d'un tableau

Ex. let Tab=[3,4,5];

Tab.unshift(1,2); // Tab=[1,2, 3, 4,5]

- shift(): permet de **supprimer** le **premier** élément d'un tableau

Ex. let Tab=[3,4,5];

let x=Tab.shift() // x=3, Tab=[4,5]

Tableaux (5/5)

- slice(): crée une copie d'une partie d'un tableau. Elle prend **deux** arguments facultatifs: indice de début (inclus) et indice de fin (non inclus)

Ex. let Tab=["a", "b", "c", "d"];

let Tab1=Tab.slice(1, 3); // Tab1=["b", "c"]

- splice(): ajoute ou supprime des éléments d'un tableau. Elle prend au moins deux arguments: indice début de la MAJ, nombre d'éléments à supprimer, éléments à insérer (facultatifs)

Ex. let Tab=["a", "b", "c", "d"];

let Tab2=Tab.splice(1, 2); // Tab=["a", "d"]

let Tab3=Tab.splice(1, 0, "b", "c"); // Tab =["a", "b", "c", "d"]

Objets du navigateur

- JavaScript peut contrôler les éléments d'une page web et manipuler les parties de la fenêtre du navigateur
- En JavaScript, un navigateur est un objet window
- Cet objet possède des propriétés et des méthodes
- Ex. propriétés: event, history, location, status
- Ex. méthodes: alert(), close(), confirm(), focus()

Document Object Model (DOM)

- Le DOM (Document Object Model) est une interface de programmation API de documents HTML, XML
- Il permet **d'accéder** et de **manipuler** le contenu HTML (éléments, attributs et textes) et le style CSS d'une page web
- Transforme le document en format compréhensible par les langages tel que JavaScript

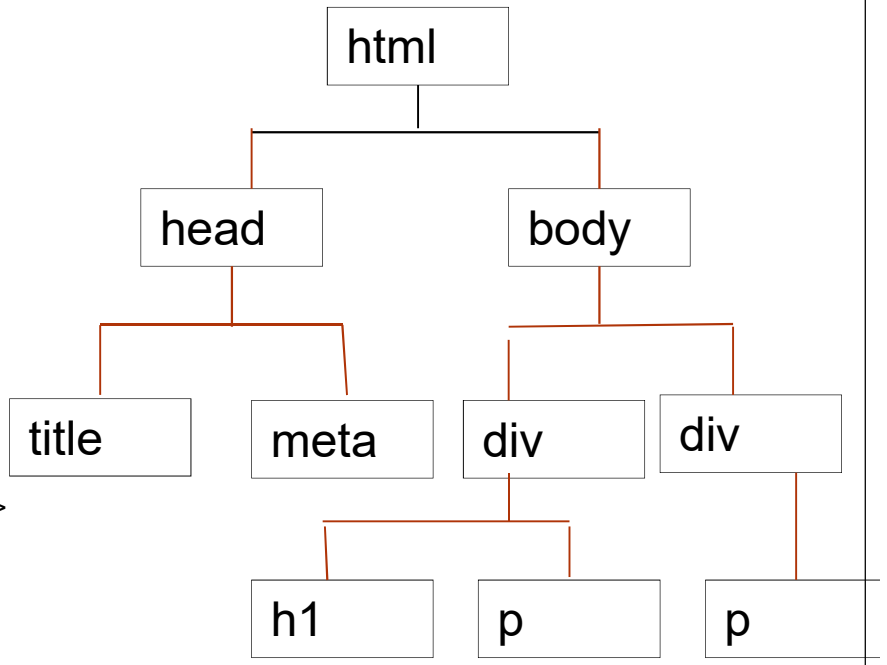
=>Un document HTML est représenté sous forme d'une arborescence d'objets.

Document Object Model (DOM)

- Le DOM est un ensemble de nœuds
- Chaque nœud représente un élément du document tel qu'une balise HTML, un attribut ou un texte
- Avec le DOM, on peut **ajouter**, **modifier** ou **supprimer** des éléments, **modifier** les styles CSS, ou réagir à des événements

Exemple

```
<!DOCTYPE html>
<html>
        <head>
                <title>titre</title>
                <meta charset="utf-8">
        </head>
        <body>
                <div>
                        <h1>titre</h1>
                        <p>Paragraphe</p>
                </div>
                <div>
                        <p>paragraphe.</p>
                </div>
        </body>
</html>
```



Accès au DOM

- L'objet 'document' identifie la page web
- Cet objet possède des propriétés et des méthodes
- Ex. var x = document.getElementById("id1").innerHTML;

Méthodes d'accès aux nœuds

- Plusieurs méthodes permettent d'accéder au DOM à partir d'un script JavaScript. On peut citer:

❑ **getElementById()** permet d'accéder à un élément HTML **spécifique** en utilisant l'attribut id de l'élément. Elle retourne un **seul** élément.

Ex. <img src="photo.jpg" alt="" id="photo1">
var photo = document.getElementById("photo1");

- **getElementsByClassName()** permet d'accéder à **tous** les éléments qui ont une certaine classe CSS. Elle retourne un **ensemble** d'éléments

Ex. Y =document.getElementsByClassName("CLASSE");

elle sélectionne **tous** les éléments de la classe " CLASSE "

- **getElementsByTagName()** permet d'accéder à tous les éléments qui ont un certain nom de balise HTML.

Ex. X=document.getElementsByTagName("p")

Elle retourne **tous** les paragraphes de la page

X[0]=premier paragraphe

- **querySelector()** permet d'accéder à un élément HTML spécifique en utilisant des sélecteurs CSS. Elle retourne le **premier** élément correspondant au sélecteur spécifié.
  - Ex. var y =document.querySelector(".classe1"); Elle retourne le premier élément qui contient la classe CSS 'classe1'
- **querySelectorAll()** permet d'accéder à un ensemble d'élément HTML en utilisant des sélecteurs CSS. Elle retourne une **liste** d'éléments correspondant au sélecteur(s) spécifié(s).
  - Ex. var elts = document.querySelectorAll(".classe2"); elle sélectionne tous les éléments de la classe CSS 'classe2'
- **etc**

Méthodes de manipulation des noeuds

- **setAttribute()** permet de changer la valeur d'un attribut. Elle prend deux arguments: l'attribut et la nouvelle valeur

  Ex. var Image = document.getElementById("image1");
  Image.setAttribute("src", "image.jpg");

- **innerHTML** permet d'accéder et de changer le texte d'un élément

  Ex.var d = document.getElementsByClassName("classe1");
  d[0].innerHTML = " paragraphe ";

- **createElement()** permet l'ajout d'éléments à la page web. Cette méthode prend un seul élément en argument qui est le nom de la balise

  Ex. var d = document.createElement("div");

- **createTextNode()** permet de créer un nouveau nœud de texte.

  Ex. var t = document.createTextNode("Texte.");

  t: noeud de texte

  Texte: son contenu

- **appendChild()** ajoute un nœud à la **fin** de la liste des enfants d'un nœud parent spécifié
- Elle prend un argument qui est le nœud à insérer au DOM

Ex. Ajouter un paragraphe qui contient le texte « paragraphe1 » à l'élément « div ».

```
var d = document.getElementById("div1");
var p = document.createElement("p");
var text = document.createTextNode(" paragraphe1");
p.appendChild(text );
d.appendChild(p);
```

- **replaceChild()** remplace un nœud enfant par un autre. Elle prend deux arguments

```
parent.replaceChild(newChild, oldChild);
```

- **removeChild()** retire un nœud enfant. Elle prend un seul argument

```
noeud.removeChild(enfant);
```

Ex.
```
var parent = document.getElementById("parentid");
var enfant = document.getElementById("enfantid");
parent.removeChild( enfant );
```

- **insertBefore()** insère un nœud avant un autre nœud dit de référence.

  parent.insertBefore(newNode, RefNode)

  Ex. Div.insertBefore(H, P); insère le nœud « H » avant le nœud « P » au niveau du nœud parent « Div »

- **style**

  Le DOM permet d'ajouter, modifier, supprimer un style d'un élément

  Ex. var x= document.getElementById("id1");

  x.style.color = "red";

Evènements en JS

- Un évènement est une action qui peut être détectée avec JavaScript
- Il est identifié par ce qu'on appelle un gestionnaire d'événement ou event handler tels que:
- **onload** lance un script lors du chargement de la page
- **onmouseover** lance un script lorsqu'un utilisateur survole un élément avec la souris

- **onclick** lance un script lorsqu'un utilisateur clique sur un élément
- **onsubmit** lance un script lorsqu'un utilisateur clique sur un button 'envoyer'
- **onkeypress**: lance un script lorsqu'une touche du clavier est enfoncée
- etc

Il existe **trois** méthodes pour appliquer les event handlers aux éléments d'une page

- **Attribut**: spécifier la fonction à exécuter dans un attribut de la page HTML

  Ex. <**h1** onclick="F();"> // la fonction F s'execute lorsqu'un utilisateur clique sur le titre de niveau 1

- **Méthode**: la fonction est attachée à un élément

  window.onclick = F; /* la fonction F s'exécute lorsqu'un utilisateur clique dans la page web */

- **addEventListener(e,f)** elle prend deux arguments:
  - e: évènement
  - f: fonction à exécuter

Ex. window.addEventListener("click", Fonction);

window.addEventListener("click", function(e) { // code de la fonction });