

Javascript

Introduction

- Le JavaScript est un langage de programmation de scripts orienté objet
- Un langage de scripts signifie qu'il s'agit d'un langage interprété
- Possède un interpréteur: Inclus dans les navigateur Web !
 - Internet Explorer: Chakra;
 - Mozilla Firefox: SpiderMonkey;
 - Google Chrome: V8
- Le JavaScript s'inclut dans la page Web ou dans un fichier externe et permet de dynamiser une page HTML, en ajoutant:
 - Des interactions avec l'utilisateur,
 - Des animations,
 - De l'aide à la navigation,
 - ...

Introduction

- JavaScript est un langage dit client-side, c'est-à-dire que les scripts sont exécutés par le navigateur chez l'internaute (le client).
- Les langages de scripts dits server-side qui sont exécutés par le serveur Web. C'est le cas des langages comme le PHP.
- Un script server-side s'occupe de créer la page Web qui sera envoyée au navigateur. Ce dernier va alors afficher la page puis exécuter les scripts client-side.

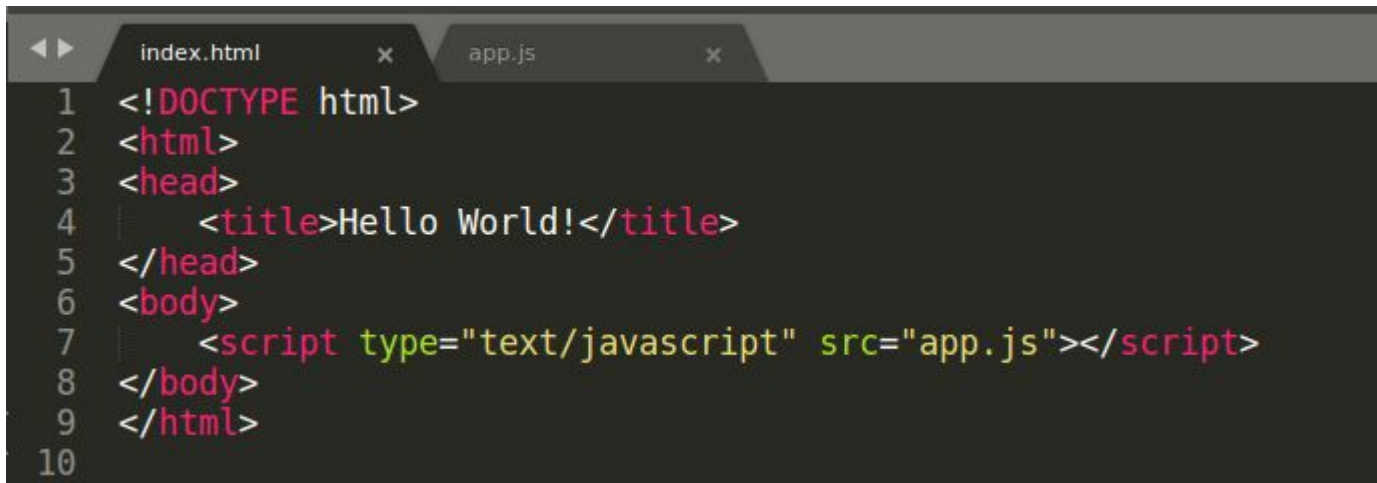
Introduction

- Le JavaScript est utilisé pour réaliser des extensions pour différents programmes, Chrome et Firefox possèdent tous deux des extensions en partie codées en JavaScript.
- JavaScript a la possibilité d'être exécuté sur n'importe quelle machine grâce au projet Node.js.
- Aujourd'hui on a la possibilité de créer une application en JavaScript grâce à Node.js + Electron ou NW.js.

Premier script

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Hello World!</title>
5  </head>
6  <body>
7      <script>
8          //commentaire: n'oublier pas le ppoint vergule
9          // apres chaque insruction
10         alert('Bonjour Tous!');
11     </script>
12 </body>
13 </html>
14
```

Premier script

A screenshot of a code editor with two tabs: 'index.html' and 'app.js'. The 'index.html' tab is active, showing an HTML document structure. The code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Hello World!</title>
5 </head>
6 <body>
7   <script type="text/javascript" src="app.js"></script>
8 </body>
9 </html>
10
```

A screenshot of a code editor with two tabs: 'index.html' and 'app.js'. The 'app.js' tab is active, showing a single line of JavaScript code:

```
1
2 alert("Bonjour Tous!!")
3
4
```

Les Variables

- Le nom d'une variable ne peut contenir que des caractères alphanumériques
- le nom de la variable ne peut pas commencer par un chiffre
- ne peut pas être constitué uniquement de mots-clés utilisés par le JavaScript
- Pour déclarer une variable il suffit de

```
1 var myVariable;  
2 myVariable = 2;
```

Les Variables (les types)

- En JavaScript, les variables sont typées dynamiquement, ce qui veut dire que l'on peut y mettre du texte en premier lieu puis l'effacer et y mettre un nombre sans contraintes.
 - Le type numérique (number)
 - Les chaînes de caractères (string)
 - Les booléens (boolean)

```
1
2  var nbr = 2;
3  alert(typeof nbr); // Affiche : « number »
4  var txt = 'un texte';
5  alert(typeof txt); // Affiche : « string »
6  var bln = false;
7  alert(typeof bln); // Affiche : « boolean »
8  alert(typeof x); // Affiche : « undefined »
9
```


Les opérateurs arithmétiques

```
1  var d = 3, r1, r2, r3;
2  r1 = (16 + 8) / 2 - 2 ; // 10
3  r2 = r1 / d;
4  r3 = r1 % d;
5  alert(r2); // Résultat de la division : 3,33
6  alert(r3); // Reste de la division : 1
7  var hi = 'Bonjour', name = 'toi', r4;
8  r4 = hi + name;
9  alert(r4); // Affiche : « Bonjourtoi »
10 alert(r4+r3); // Affiche : « Bonjourtoi1 »
11 alert(r4*r3); // Affiche : « NaN »
12
```

Les opérateurs de comparaison

Opérateur	Signification
==	égal à
!=	différent de
===	contenu et type égal à
!==	contenu ou type différent de

Opérateur	Signification
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à

Les opérateurs logiques

Opérateur	Type de logique	Utilisation
&&	ET	valeur1 && valeur2
	OU	valeur1 valeur2
!	NON	!valeur

La condition « if else »

```
1  if ( /* condition */ ) {  
2  
3      // Du code...  
4  
5  } else {  
6  
7      // Du code...  
8  
9  }
```

La condition « switch »

```
1  var n = parseInt(prompt('Choisissez le tiroir à ouvrir (1 à 4) :'));
2  switch (n) {
3      case 1:
4          alert('Contient divers outils pour dessiner');
5          break;
6      case 2:
7          alert('Contient du matériel informatique');
8          break;
9      case 3:
10         alert('Ce tiroir est fermé à clé !');
11         break;
12     case 4:
13         alert('Contient des vêtements');
14         break;
15     default:
16         alert("Le meuble ne contient que 4 tiroirs");
17 }
18
```

Exercice

1. L'utilisateur charge la page Web
2. Il est ensuite invité à taper son âge dans une fenêtre d'interaction ;
3. Une fois l'âge fourni l'utilisateur obtient un petit commentaire selon le tableau suivant:

Tranche d'âge	Exemple de commentaire
1 à 17 ans	« Vous n'êtes pas encore majeur. »
18 à 49 ans	« Vous êtes majeur mais pas encore senior. »
50 à 59 ans	« Vous êtes senior mais pas encore retraité. »
60 à 120 ans	« Vous êtes retraité, profitez de votre temps libre ! »

Les boucles

```
while (condition) {  
    instruction_1;  
    instruction_2;  
    instruction_3;  
}
```

```
do {  
    instruction_1;  
    instruction_2;  
    instruction_3;  
} while (condition);
```

```
for (initialisation; condition; incrémentation) {  
    instruction_1;  
    instruction_2;  
    instruction_3;  
}
```

Les fonctions

```
function showMsg() {  
    alert('Et une première fonction, une !');  
}
```

```
showMsg(); // On exécute ici le code contenu dans la fonction
```

```
var msg1="Messge globale"  
function showMsg() {  
    var msg2="Message local"  
    msg3="un autre Message"  
    alert(msg1);  
    alert(msg2);  
    alert(msg3);  
}  
showMsg(); // affiche "Messge globale", "Message local" puis "un autre Message"  
alert(msg1); // affiche "Messge globale"  
alert(msg2); // erreur !! msg2 inconnue  
alert(msg3); // affiche "un autre Message"
```


Les fonctions

```
function somme(a, b) {  
    if ((typeof a === 'undefined') || (typeof b === 'undefined')){  
        alert("besoin de deux arguments");  
        return NaN  
    }  
    return parsInt(a)+parsInt(b)  
}
```

Les fonctions anonymes

- Ces fonctions sont extrêmement importantes en JavaScript :
 - les objets,
 - les évènements,
 - les variables statiques,
 - ...

```
function (arguments) {  
    // Le code de votre fonction anonyme  
}
```

```
var sayHello = function() {  
    alert('Bonjour !');  
};  
  
sayHello();
```

```
// Code externe  
  
(function() {  
  
    // Code isolé  
  
})();  
  
// Code externe
```

La portée des variables (scop)

Var

Let

const

Les objets (string)

- Les objets contiennent :
 - un constructeur ;
 - des propriétés ;
 - des méthodes.

```
var myString = 'Ceci est une chaîne de caractères'; // On crée un objet String

alert(myString.length); // On affiche le nombre de caractères, au moyen de la
propriété « length »

alert(myString.toUpperCase()); // On récupère la chaîne en majuscules, avec la méthode
toUpperCase()
```

Les objets (string)

- .length
- .toLowerCase()
- .toUpperCase()
- .charAt() , []

Les objets (array)

```
var tab1 = [42, 12, 6, 3];
var tab2 = [42, 'blue', 12, 'blanc'];
alert(tab1[0]);
var tab3 = new Array('un', 'deux', 'trois');
tab3[3]='quatre';
tab3.push('cinq','six'); // Ajoute « cinq » et « six » à la fin
tab3.shift();//Retire "un"
tab3.pop(); //Retire "six"
tab3.unshift("un"); // Ajoute « un » au debut du tableau
var s="un dex trois";
tab=s.split(" "); // tab=['un', 'dex', 'trois']
s=tab.join("_"); // s="un_dex_trois"
```

Les objets littéraux

```
1  var etudiant{
2      nom:"Ben bachir",
3      prenom: "reda",
4      notes:[14,12.5]
5  };
6  etudiant.age=20;
7  console.log(etudiat.nom);
8  console.log(etudiat["prenom"]);
9  var p="prenom";
10 console.log(etudiant[p]);
11 for(var key in etudiant){
12     console.log(etudiant[key]);
13 }
```

Les constructeurs d'objets

```
1  function Etudiant(nom, prenom){
2      this.nom=nom;
3      this.prenom=prenom;
4      this.notes=[];
5      this.ajouterNote=function(){
6          this.notes.push(Math.random()*20);
7      };
8  }
9  var e1 = new Etudiant("Alami","Yahya");
10 var e2 = new Etudiant("Ibrahimi","Mouna");
11 var e=[e1,e2];
12 e1.ajouterNote();
13 e2.ajouterNote();
14 console.log(e1);
```


Closures

Définition d'une Fonction f()
dans une autre g().

La fonction f() utilise des
variables locales de la
fonction g()

```
1  function calcul(){
2      var a=5;
3      var b=7;
4      function somme(){
5          var s=a+b;
6          return s;
7      }
8      return somme;
9  }
10 var r=calcul();
11 console.log(r);
12 console.log(r());
```

Closures (application)

```
1  function px2em(px){
2      var basePx=16;
3      function calc(){
4          return px/basePx;
5      }
6      return calc;
7  }
8  var sm=px2em(12);
9  var md=px2em(18);
10 console.log("Small: ", sm());
11 console.log("Medium: ", md());
```

Exercice

Conversion d'un nombre en toutes lettres:

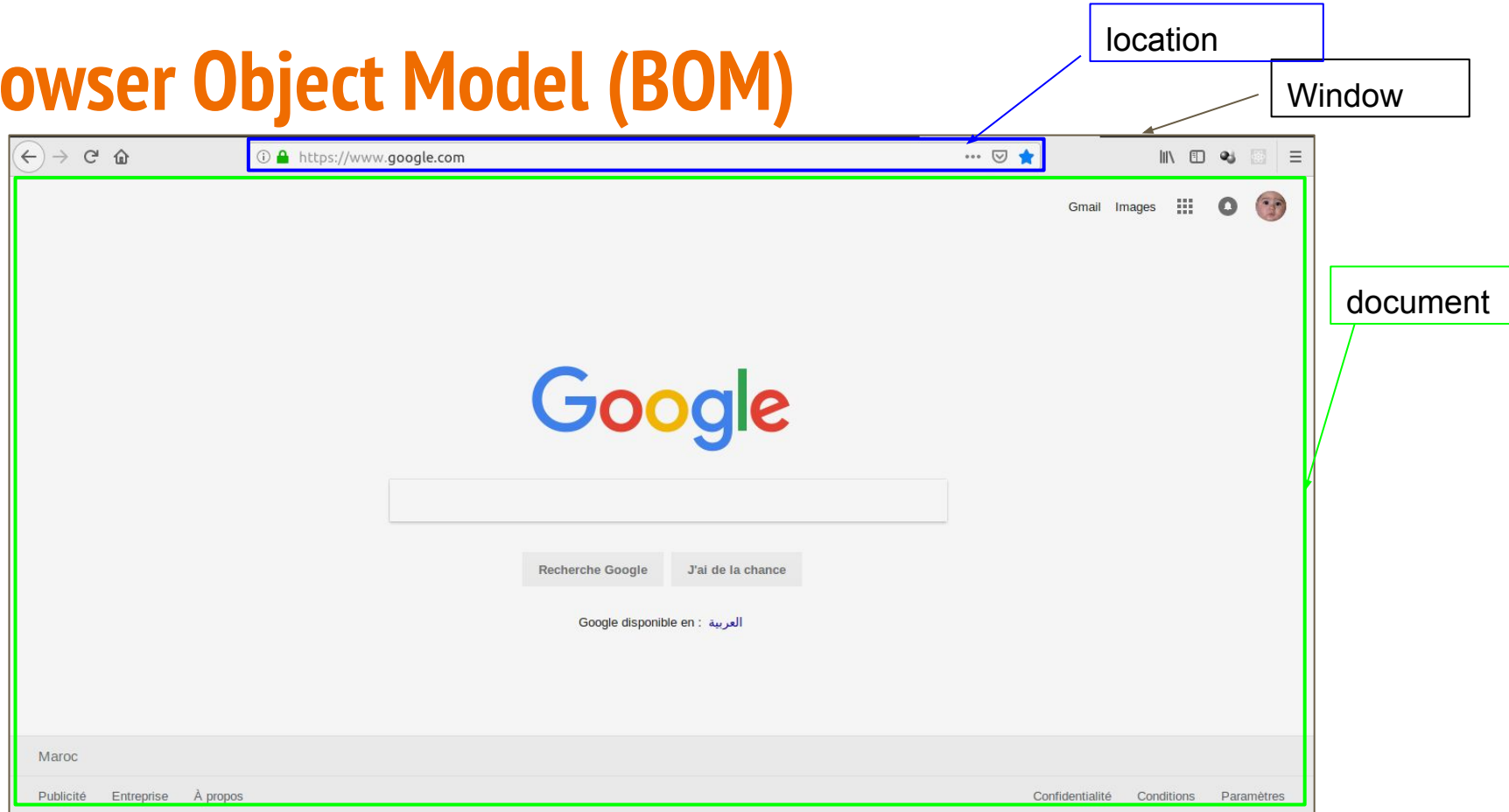
si l'utilisateur entre le nombre « 38 », le script devra retourner ce nombre en toutes lettres : « trente-huit ».

si l'utilisateur entre le nombre « 42 », le script devra retourner ce nombre en toutes lettres : « quarante-et-deux ».

Note: la fonction ***isNaN()*** permet de voir si une variable n'est pas un nombre

```
var test = parseInt('test'); // Contient au final la valeur « NaN »  
alert(typeof test); // Affiche « number »  
alert(isNaN(test)); // Affiche « true »
```

Browser Object Model (BOM)



Browser Object Model (BOM)

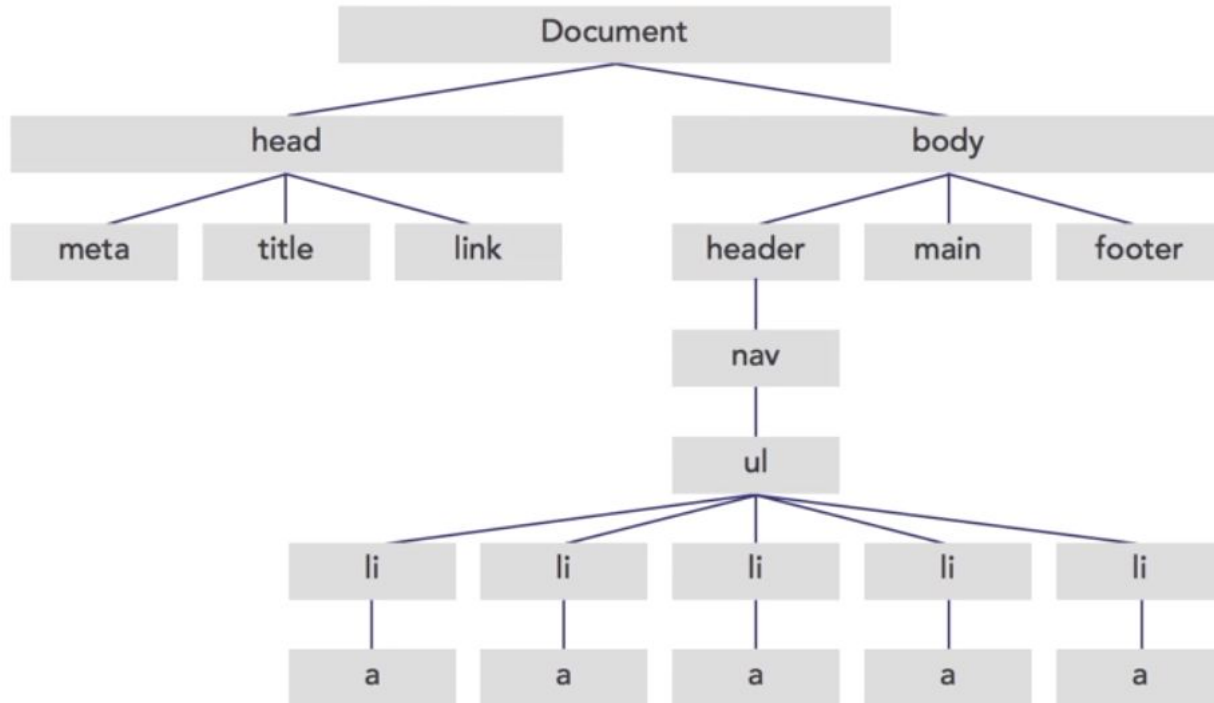
Window est l'objet du plus haut niveau dans le BOM et possède un ensemble de propriétés et méthode pour interagir avec le Browser

- `window.innerWidth`
- `window.open()`
- `Window.location`
- `.....`
- `Window.document`

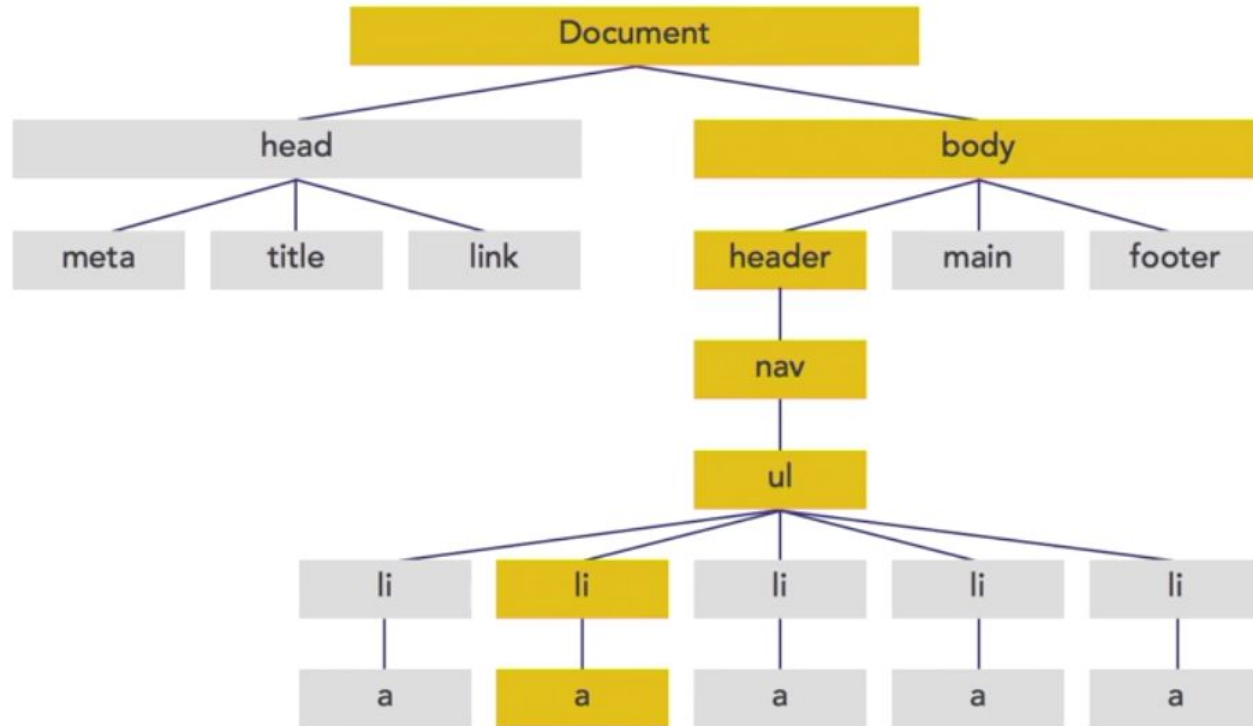
Le Document Object Model (DOM)

- Le DOM est une API qui s'utilise avec les documents XML et HTML, et qui va nous permettre, via le JavaScript, d'accéder au code XML et/ou HTML d'un document.
- C'est grâce au DOM que nous allons pouvoir:
 - modifier des éléments HTML (afficher ou masquer un<div>par exemple),
 - ajouter des éléments HTML
 - déplacer des éléments HTML
 - Supprimer des éléments HTML

Le Document Object Model (DOM)



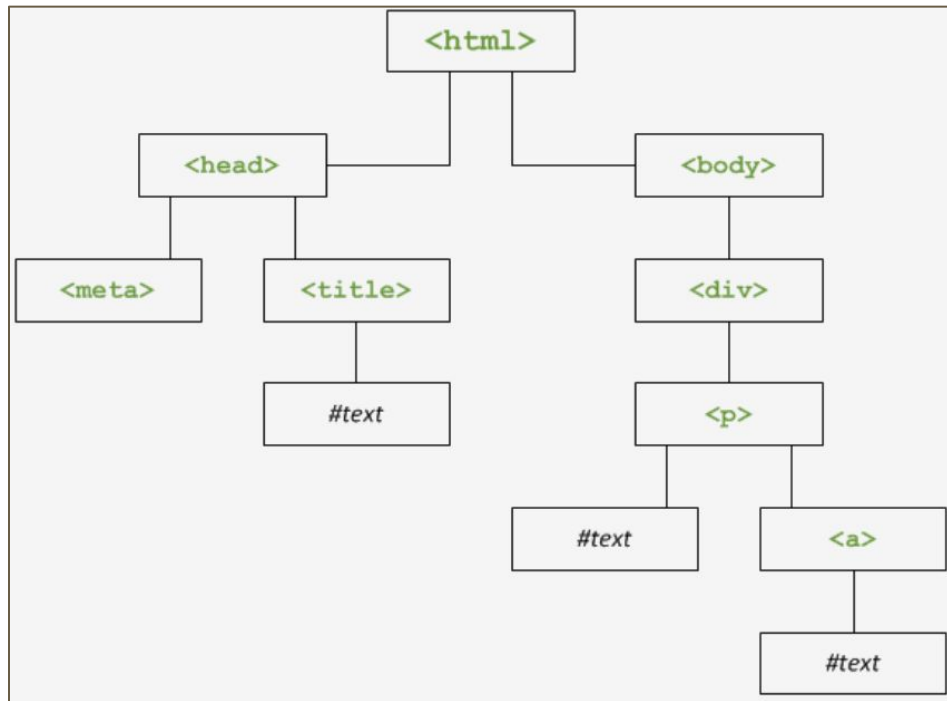
Le Document Object Model (DOM)



Le Document Object Model (DOM)

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Le titre de la page</title>
</head>

<body>
  <div>
    <p>Un peu de texte <a>et un lien</a></p>
  </div>
</body>
</html>
```



Accéder aux éléments

1. `document.body`
2. `Document.title`
3. `document.URL`
4. `document.getElementById("#id"),`
5. `document.getElementsByTagName("tagName")`
6. `document.getElementsByClassName(".class")`
7. `document.getElementsByName("name")`
8. ...

Accéder aux éléments

document.querySelector("css selector") => Le premier element qui verifier le sélecteur css spécifier

document.querySelectorAll("css selector") => Tous les éléments vérifiant le sélecteur CSS

Ces deux méthodes sont les plus utilisées

Accéder aux éléments

1. `querySelector()`
2. `querySelectorAll()`

```
var query = document.querySelector('#menu .item span'),
    queryAll = document.querySelectorAll('#menu .item span');

alert(query.innerHTML); // Affiche : "Élément 1"

alert(queryAll.length); // Affiche : "2"
alert(queryAll[0].innerHTML + ' - ' + queryAll[1].innerHTML); // Affiche : "Élément 1
- Élément 2"
```

```
<div id="menu">

  <div class="item">
    <span>Élément 1</span>
    <span>Élément 2</span>
  </div>

  <div class="publicite">
    <span>Élément 3</span>
    <span>Élément 4</span>
  </div>

</div>

<div id="contenu">
  <span>Introduction au contenu de la page...</span>
</div>
```

Accéder aux attributs d'un éléments

Les attributs des élément HTML peuvent êtres accède en lecture/écriture ou en lecture seule

1. `element.attributes`
2. `element.innerHTML`
3. `element.outerHTML`
4. `element.clientHeight`
5. `element.className`
6. `element.classList`
7. `element.id`
8. ...

Accéder aux attributs d'un éléments

Pour modifier les attributs en lecture seul on utilise de methodes:

1. `element.classList.add("une nouvelle classe")`
2. `element.classList.remove("une classe existante")`
3. `element.classList.contains("une classe")`
4. `element.haseAttribute(attribut)`
5. `element.getAttribute(attribut)`
6. `element.setAttribute(attribut, value)` // modifier ou ajouter un attribut
7. `element.removeAttribute(attribut)`

Exp: **`document.querySelector("a").setAttribute("target","_blank")`**

Accéder aux éléments

getAttribute() et setAttribute()

```
<body>
  <a id="myLink" href="http://www.un_lien_quelconque.com">Un lien modifié
dynamiquement</a>

  <script>
    var link = document.getElementById('myLink');
    var href = link.getAttribute('href'); // On récupère l'attribut « href »

    alert(href);

    link.setAttribute('href', 'http://www.siteduzero.com'); // On édite l'attribut
« href »
  </script>
</body>
```

Accéder aux éléments (Element.className)

```
3 <head>
4   <meta charset="utf-8" />
5   <title>Le titre de la page</title>
6   <style>
7     .blue {
8       background: blue;
9       color: white;
10    }
11  </style>
12 </head>
13 <body>
14   <div id="myColoredDiv">
15     <p>Un peu de texte <a>et un lien</a></p>
16   </div>
17   <script>
18     document.getElementById('myColoredDiv').className = 'blue';
19   </script>
20 </body>
```


Accéder aux éléments (Element.classList)

```
1  var div = document.querySelector('div');
2  // Ajoute une nouvelle classe
3  div.classList.add('new-class');
4  // Retire une classe
5  div.classList.remove('new-class');
6  // Retire une classe si elle est présente ou bien l'ajoute si elle est absente
7  div.classList.toggle('toggled-class');
8  // Indique si une classe est présente ou non
9  if (div.classList.contains('old-class')) {
10     alert('La classe .old-class est présente !');
11 }
12 // Parcourt et affiche les classes CSS
13 var result = '';
14 for (var i = 0; i < div.classList.length; i++) {
15     result += '.' + div.classList[i] + '\n';
16 }
17 alert(result);
18
```

Accéder aux éléments (Element.textContent)

Ajout d'un élément au DOM

- | | |
|--|--|
| 1. Créer l'élément | <code><= document.createElement()</code> |
| 2. Créer le noeud texte de cet élément | <code><= document.createTextNode()</code> |
| 3. Ajouter le noeud texte à l'élément | <code><= document.appendChild()</code> |
| 4. Ajouter l'élément au DOM | <code><= document.appendChild()</code> |

Exercice:

Ajouter un élément `<caption>...</caption>` à l'élément `<figure ..> ...</figure>`

```
9  <figure class="mafig">
10    
11  </figure>
```

Ajout d'un élément au DOM

```
1 const fig=document.querySelector(".mafig");
2 const img=fig.querySelector(".monImg");
3 var altTxt=img.getAttribute("alt");
4 var capElmt=document.createElement("figcaption");
5 var capTxt=document.createTextNode(altTxt);
6 capElmt.appendChild(capTxt);
7 fig.appendChild(capElmt);
8 console.log(fig);
```

Une nouvelle methode : .append()

```
1 const fig=document.querySelector(".mafig");
2 const img=fig.querySelector(".monImg");
3 var altTxt=img.getAttribute("alt");
4 var capElmt=document.createElement("figcaption");
5 capElmt.append(altTxt);
6 fig.append(capElmt);
```

Style CSS Inline

Avec l'attribut style on peut ajouter n'importe quel propriété CSS à n'importe quel élément.

- ***Element.style;*** => **uniquement** le **Inline** CSS de l'élément {attribut:"value", ... } mais pas les autre style définie dans des fichier css au dans le head.
- ***Element.style.color="blue";***
- ***Element.style.backgroundColor="yellow";*** //backgroundColor pas background-color
- ***Element.style.cssText="color: blue; background-color: yellow; ..."***
- ***Element.style.setAttribute("style", "color: blue; background-color: yellow; ...");***

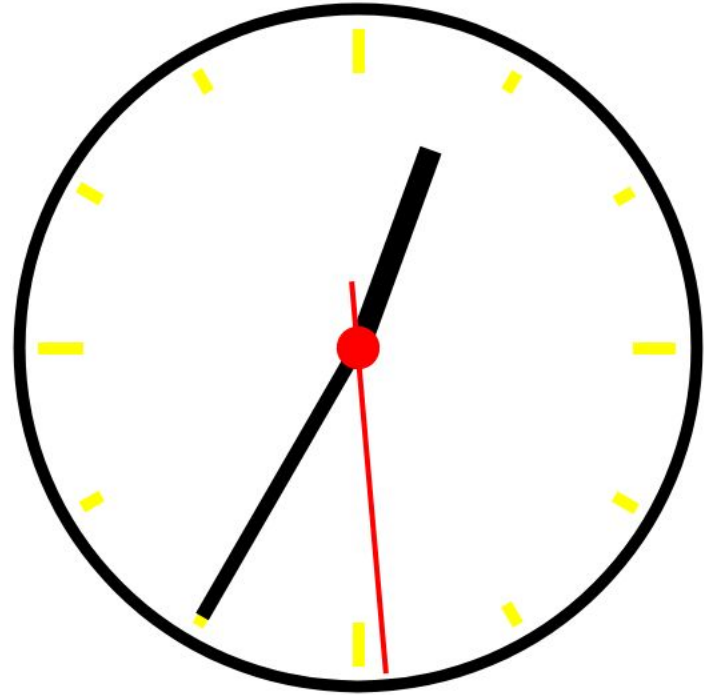
Style CSS Inline

Les style CSS inline remplace les style definie dans les feuille de style.

Dans la plupart des cas il vaut mieu définir des règles CSS et gérer les class avec javascript

Exercice

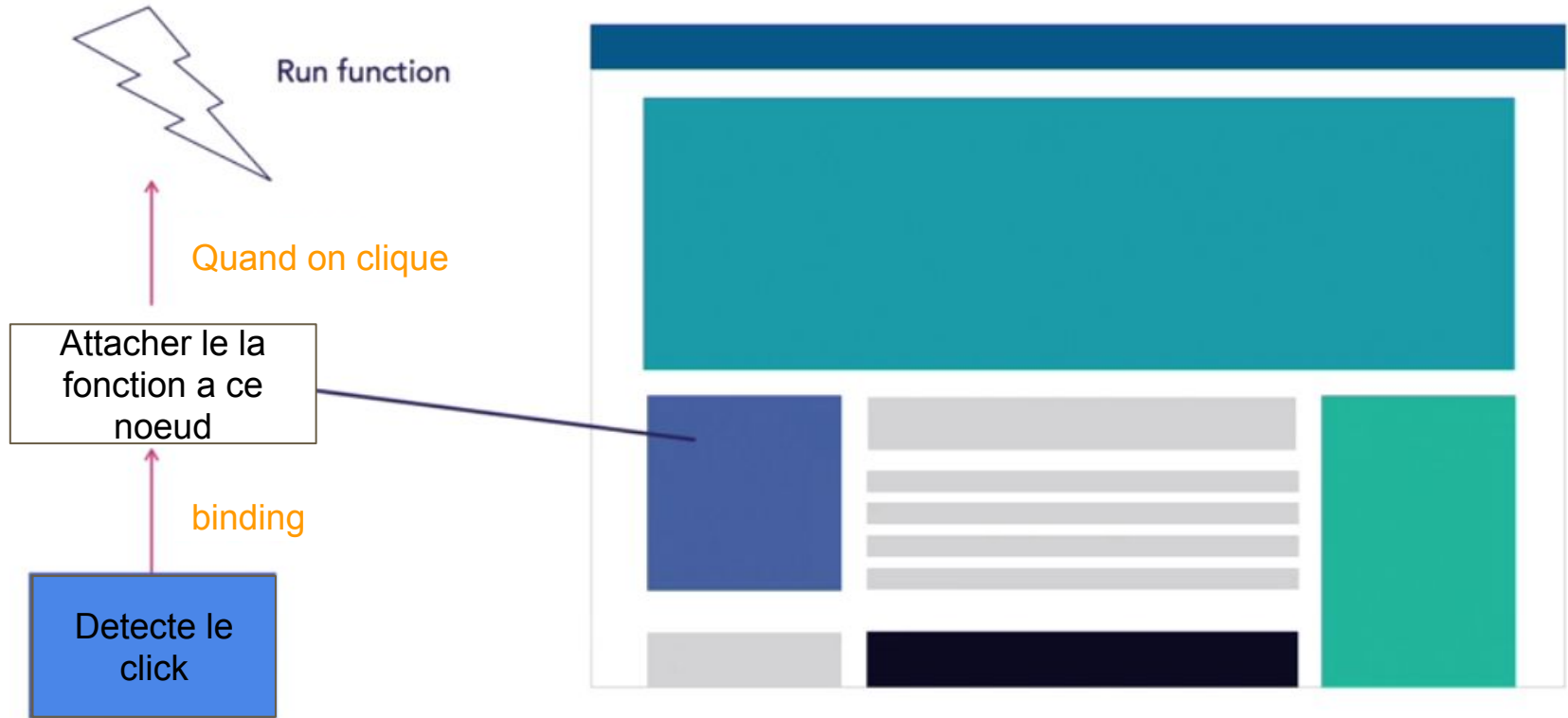
Créer une horloge on utilisant une image SVG
a laide d'un editeur de graphiques vectoriel



Les evenements



Les evenements



Les evenements

load

error

resize

online

Offline

Scrol

...

Click

Dblclick

Mouseover

Mouseout

Mousedown

Mouseup

Mousemove

Keydown

Keyup

KeyPress

Focus

Blur

...

Change

Input

Select

Reset

Submit

...

Les evenements

```
1 function eventclbk(e){
2     e.preventDefault();// annule le comportement par default
3     // autres traitement si l evennement e surgit
4 }
5
6 //element.onevent= eventclbk;
7 element.onclick = eventclbk;
```

Les evenements

```
1 const elmt=document.querySelector(".normal");
2 function eventclbk(e){
3     console.log("cliked");
4     elmt.classList.toggle("normal");
5 }
6 elmt.onclick = eventclbk;
```

event.js

event.html

```
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
<style type="text/css">
    .normal{
        background-color: yellow;
    }
    div{
        background-color: blue;
        text-align: center;
        font-size: 20px;
        height: 40px;
    }
</style>
</head>
<body>
    <div class="normal">click me</div>
</body>
```

Les evenements

```
1  const elmt=document.querySelector(".normal");
2  function eventclbk1(e){
3      elmt.classList.toggle("normal");
4  }
5  function eventclbk2(e){
6      console.log("Div clicked");
7  }
8  elmt.onclick = eventclbk1;
9  elmt.onclick = eventclbk2;
```

Problème: uniquement la fonction eventclbk2 sera exécutée

Les evenements

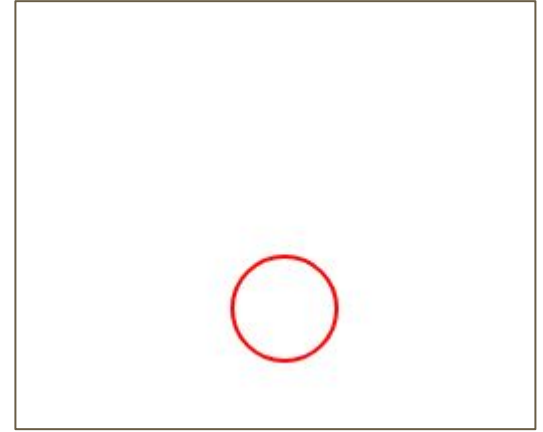
element.addEventListener("event", function_clbck, [true/false]);

```
1  const elmt=document.querySelector(".normal");
2  function eventclbk1(e){
3      elmt.classList.toggle("normal");
4  }
5  function eventclbk2(e){
6      console.log("Div clicked");
7  }
8  elmt.addEventListener("click",eventclbk1,false);
9  elmt.addEventListener("click",eventclbk2,false);
```

Les evenements

Exercice:

1. Le cercle se déplace pour garder son centre en symétrie avec le curseur de la souris.
2. Le cercle change de couleur lorsqu'il touche curseur.
3. Utiliser ***.clientX*** et ***.clientY*** de l'objet ***event***



Les evenements

Exercice:

Typing Speed

Les événements permettent de déclencher une fonction selon qu'une action s'est produite ou non.

Recopie le texte en haut. le chrono démarre avec la saisie.

00:00:00

redémarrer

ES6

Javascript introduit par Netscape

Puis ECMA International pour la standardisation

ES6 == EcmaScript 6 (2015) ES7 (2016) ES8(2017)

⇒ > le code javascript devient plus simple

Mais pas supporter par la plupart des navigateur

=>Babel

ES6

Javascript

```
var a=function(x,y){  
    return x+y;  
}
```

ES6

```
const a=(x,y)=>x+y;
```

ES6: destructring

Javascript

```
1  const etudiant={
2      nom: "Baddi",
3      prenom: "Ahmed",
4      age: 24
5  }
6  const nom=etudiant.nom;
7  const prenom=etudiant.prenom;
8  const age=etudiant.age;
```

ES6

```
1  const etudiant={
2      nom: "Baddi",
3      prenom: "Ahmed",
4      age: 24
5  }
6
7  const {nom, prenom, age}=etudiant;
```

ES6: proprietes des objets

```
1  const a="nom";  
2  const b="pre";  
3  const etudiant={  
4      [a]: "Baddi",  
5      [b+a]: "Ahmed",  
6      age: 24  
7  }  
8  console.log(etudiant.prenom);
```

ES6: proprietes des objets

```
1  const nom="Baddi";
2  const prenom="Ahmed";
3  const age=24;
4  const etudiant={
5      nom: nom,
6      prenom: prenom,
7      age: age
8  }
9  //ES6
10 const etudiant={
11     nom,
12     prenom,
13     age
14 }
15
```

ES6: template Strings

```
1  const nom="Baddi";
2  const prenom="Ahmed";
3  const age=24;
4  let s= "Mon nom est"+nom+"j'ai "+age+"
        ans";
5  //ES6
6  let s=`Mon nom est: ${nom} j'ai ${age}
        ans`;
```

ES6: Arguments par défaut

```
1  function cal(a=0,b=1){  
2      return a/b;  
3  }  
4  
5  calc(); // 0  
6  calc(4); // 4  
7  calc(4,8); // 0.5
```

ES6: Arrow function

```
1  function somme(a,b){  
2      return a+b;  
3  }  
4  //ES6  
5  const somme=(a,b)=> {  
6      return a+b  
7  }  
8  
9  const somme=(a,b)=> a+b;
```


ES6: Closures

```
1 function f1(){
2     var a="Bonjour";
3     function f2(){
4         console.log(a);
5     }
6     return f2;
7 }
8 var f=f1();
9 f();
```

ES6: Closures

```
1 function f1(){
2     var a="Bonjour";
3     function f2(){
4         console.log(a);
5     }
6     return f2;
7 }
8 var f=f1();
9 f();
```

```
//ES6
const f1={()=>{
    const a="Bonjour";
    const f2={()=>{
        console.log(a);
    }
}
var f=f1();
f();
```

ES6: Currying

Transformer une fonction a plusieurs variable en plusieurs fonctions a une seule variable.

```
1  const prod1=(a,b)=> a*b;  
2  prod1(3,4); // 12  
3  
4  const prod2=(a)=>(b)=>a*b;  
5  prod2(3); // ?  
6  prod2(3)(4); // ?
```

ES6: Currying

Transformer une fonction a plusieurs variable en plusieurs fonctions a une seule variable.

```
1  const prod1=(a,b)=> a*b;  
2  prod1(3,4); // 12  
3  
4  const prod2=(a)=>(b)=>a*b;  
5  prod2(3); // ?  
6  prod2(3)(4); // ?
```

ES6: Compose

```
1  const comp = (f, g) => (a) => f(g(a));  
2  let somme = (n) => n+1;  
3  comp(somme, somme)(3); // ?
```

Array

```
1  const a=[1,4,3,8];
2  let b=a.forEach((n)=>{
3      n*2;
4  });
5  console.log(b);// undefined
6  let b=a.map((n)=> n*2);
7  console.log(b);// [2,8,6,16]
8
9  let c=a.filter((n)=> n%2 === 0);
10 console.log(c);// [4,8]
11
12 let d=a.reduce((s,n)=> s+n,2);
13 console.log(d);// 18
14
```

Objets : references

```
1  let a={val: 3};  
2  let b=a;  
3  let c={val: 3};  
4  console.log(a===b); //true  
5  console.log(a===c); //false  
6  a.val=2;  
7  console.log(a.val); //2  
8  console.log(b.val); //2  
9  console.log(c.val); //3
```

Objets: contexte

```
1  const obj={
2      a:function(){
3          console.log(this);
4      }
5  }
6  obj.a();//obj
7  console.log(this);// window
8  function a(){
9      console.log(this);
10 }
11 a()// window
```


Objets: instantiation

```
class Homme{  
  constructor(nam, age){  
    this.nom=nom;  
    this.age=age;  
  }  
  hi(){  
    console.log(`Je suis ${this.nom}`);  
  }  
}  
let h=new Homme("yakoubi",22);
```

Objets: instantiation

```
class Etudiant extends Homme{
  constructor(nom, age, note){
    super(nom, age);
    this.note=note;
  }
  getNote(){
    console.log(`Note: ${this.note}`);
  }
}
let h=new Homme("yakoubi",22,17);
```