

Introduction à



G.Ménier

ADMINISTRATION



▼ Administration du cours

Activer le mode édition

Paramètres

► Utilisateurs

Filtres

► Rapports

Notes

Configuration du carnet de notes

Sauvegarde

Restauration

INF1405

Projet Génie Logiciel

Objectifs : Introduction au génie logiciel et consolidation du niveau en programmation et en algorithmique à travers un projet informatique



Introduction

- 1995 Mocha -> LiveScript -> ECMAScript -> Javascript
- Norme
- Rien à voir avec Java
 - Java : compilé / Javascript interprété
 - Javascript : navigateurs + node.js
 - Langage proche de c / c++ / Java
 - Non (fortement) typé
 - Une variable peut changer de type
 - Entier / réel / chaîne de caractères etc..
 - Langage objet MAIS...
 - Prototypes

Javascript



<http://www.ecma-international.org/>

Utilisé ailleurs sous d'autres noms

Application	Dénomination
Navigateurs de type Gecko avec le moteur embarqué SpiderMonkey , dont Mozilla Firefox	JavaScript
Internet Explorer	JScript
Opera	ECMAScript, avec des extensions JavaScript et JScript
KHTML based browsers, including KDE's Konqueror	JavaScript
Framework .NET de Microsoft	JScript .NET et Managed JScript
Adobe Flash	ActionScript
Adobe Acrobat	JavaScript
General purpose scripting language	DMDScript
OpenLaszlo Platform	JavaScript
iCab	InScript
Implémentation d' XML dans les navigateurs basés sur Gecko et les programmes embarqués comme SpiderMonkey	E4X

Javascript



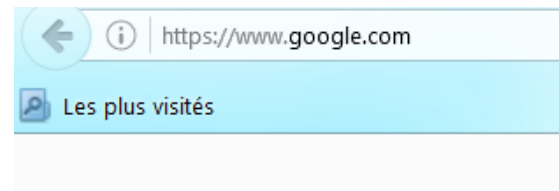
- LE langage web et internet
 - Ordinateurs
 - Téléphones
 - Jeux videos
- Fonctionne dans tous les navigateurs
 - Firefox
 - Chrome
 - IE
 - Opera
 - Etc..
- Liens avec HTML
- Rappels sur HTML

Javascript



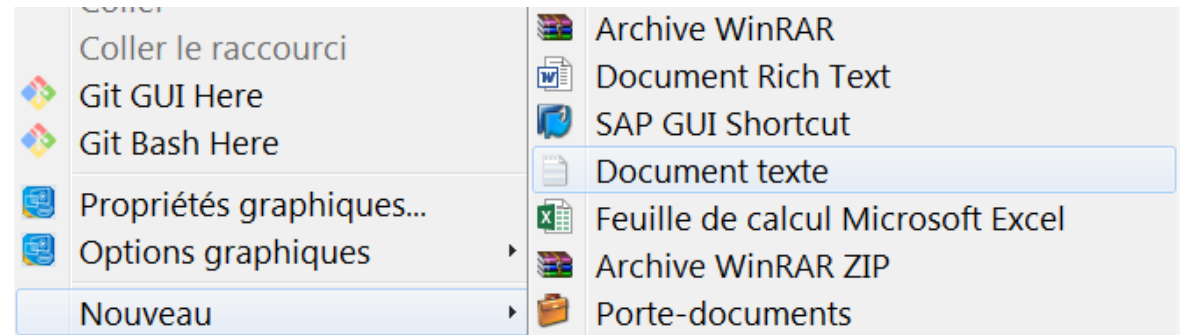
• Navigateur (Firefox)

- Adresse de la page :
 - `http:// ...` sur internet
 - `file:// ...` en local, sur l'ordinateur courant

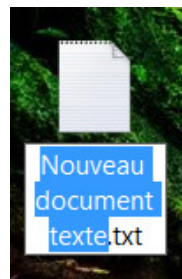


• Création simple d'une page Web :

- Création d'un fichier
- Renommage du fichier
- Extension .html
- HTML = HyperText Markup Language



- Click droit bureau windows
- Renommer en .html



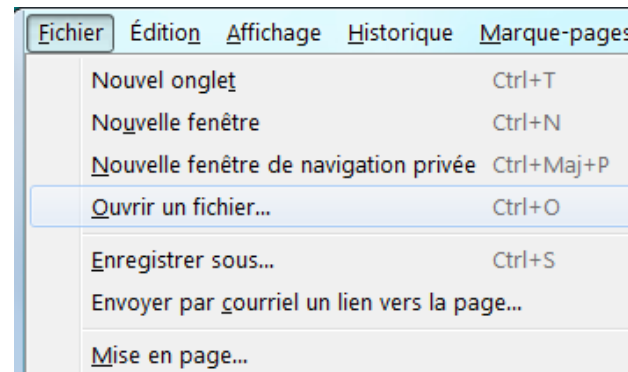
Javascript



- Ouverture de la page par le navigateur

- Ou double-click sur le fichier

- Fichier vide -> page vide



Javascript



- Modifier page.html (click droit editer avec notepad++)

- Cycle

- Modifier la page
- Sauvegarder la page
- Recharger la page dans le navigateur



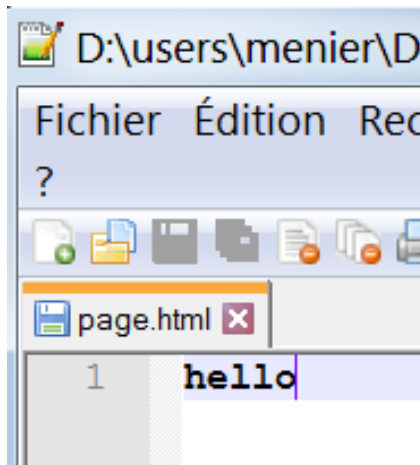
- Création d'une page Web : texte, image etc..
- Rajouter un programme Javascript à cette page
- Le code javascript modifie la page
 - Animation, sounds, web, communication, calculs etc..
- Javascript fonctionne DANS une page web

Javascript

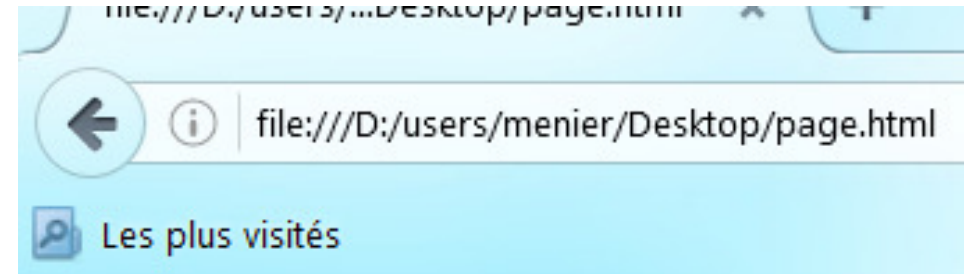


- HTML

- Codage avec des balises
- Exemple : Hello



- page.html dans l'éditeur



hello

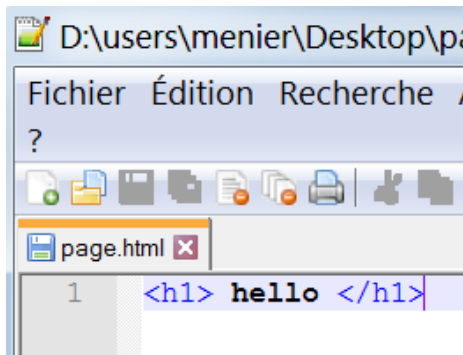
- page.html dans le navigateur

Javascript

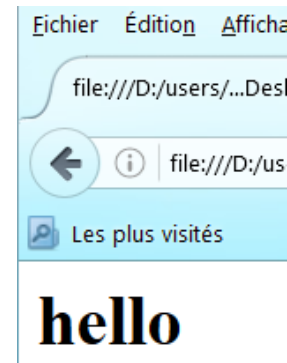


- HTML

- Codage avec des balises
- Exemple : `<h1> hello </h1>`



- page.html dans l'éditeur



- page.html dans le navigateur

Javascript



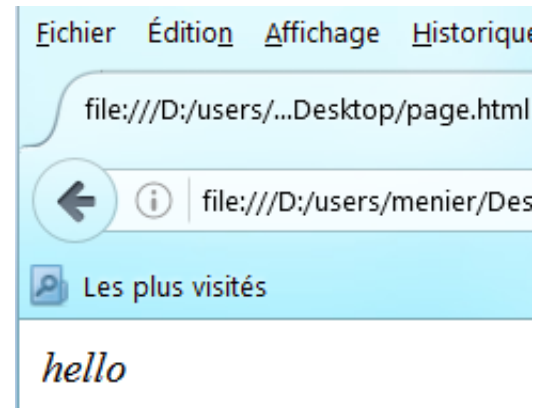
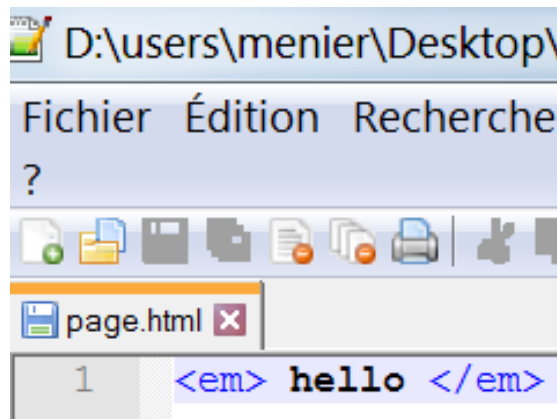
- Balise H1

- Balise de taille
- Ce qui est entre `<h1>` et `</h1>` est écrit en plus gros
- Le texte `<h1>` n'apparaît pas (`</h1>` non plus)
- H1 ou h1, ici c'est pareil pour HTML
- Attention, pour HTML, indifférent
 - MAIS PAS POUR JAVASCRIPT
 - Majuscules / minuscules importantes pour Javascript
 - Pas pour HTML

Javascript



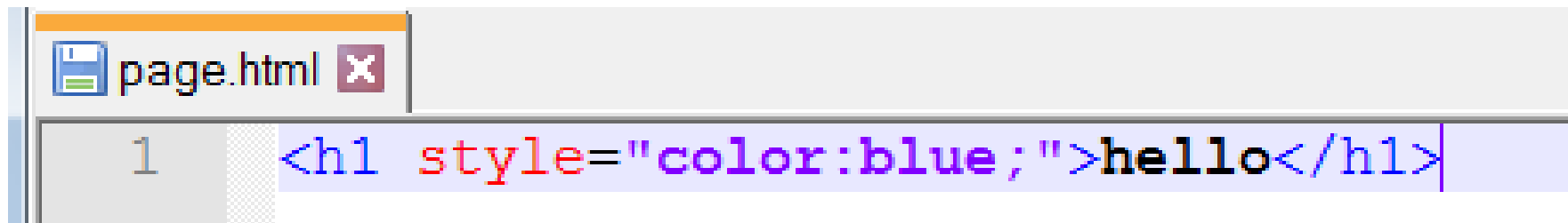
- Balises H1, H2, H3, H4 etc..
- H1 le plus gros
- D'autres balises
- Par exemple : em



Javascript



- On peut modifier le comportement des balises avec des attributs

A screenshot of a web browser window. The title bar shows "page.html" with a close button. The address bar is empty. The main content area displays the HTML code: `<h1 style="color:blue;">hello</h1>`. The code is color-coded: the opening tag is blue, the attribute value is red, and the word "hello" is black. The browser's rendering of the code is shown below the code editor.

- Donne dans le navigateur :

hello

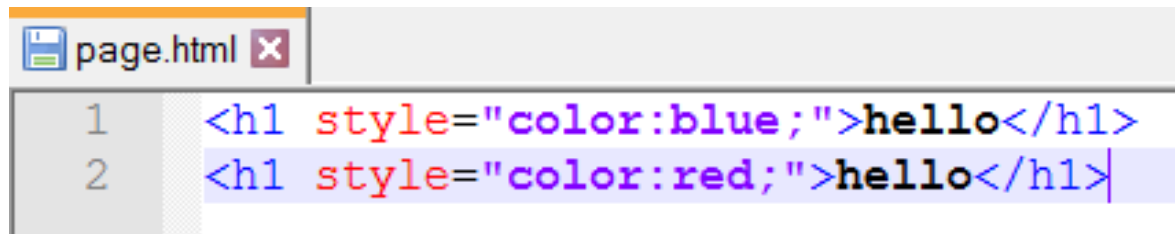
- Attention à la place des attributs : DANS < > après le nom balise

Javascript



- **Attributs HTML**

- `<h1 style="color:blue;">hello</h1>`
- H1 est le nom de la balise
- style est le nom de l'attribut de la balise H1
- "color:blue;" est une chaîne de caractères, c'est la valeur de l'attribut style
 - Seulement pour cette balise H1
- Chaque balise peut avoir ses propres attributs

A screenshot of a code editor window titled "page.html". It shows two lines of HTML code. Line 1: `<h1 style="color:blue;">hello</h1>`. Line 2: `<h1 style="color:red;">hello</h1>`. The code is syntax-highlighted with blue for tags, red for the style attribute, and black for the text.

hello

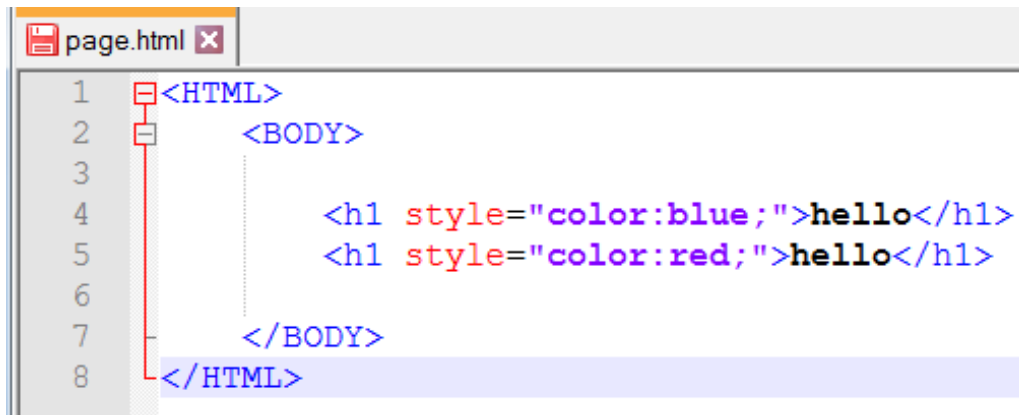
hello

Javascript



- Organisation d'une page HTML

- Pour des raisons d'organisation, une page HTML est structurée de la manière suivante :

A screenshot of a web browser or code editor window titled "page.html". The code is as follows:

```
1 <HTML>
2   <BODY>
3
4       <h1 style="color:blue;">hello</h1>
5       <h1 style="color:red;">hello</h1>
6
7   </BODY>
8 </HTML>
```

hello

hello

- La balise HTML indique un document web
 - La balise BODY indique le contenu du document web
 - Normalement obligatoires (mais *souvent* les navigateurs acceptent sans)
 - Prendre l'habitude de les utiliser

Javascript



- Javascript

- HTML décrit un document
 - Ce n'est pas un langage de programmation
 - C'est un langage de DESCRIPTION
- Javascript est un langage de programmation
 - Il peut calculer
 - Il peut modifier le document HTML
 - Il peut communiquer avec le Web et Internet
- Javascript existe dans une balise SCRIPT
 - À la fin du contenu HTML
 - De préférence
 - Une fois que la page est construite

Indentation !

```
1 <HTML>
2 <BODY>
3
4 <h1 style="color:blue;">hello</h1>
5 <h1 style="color:red;">hello</h1>
6
7 <script>
8 // le code Javascript habite ici :-)|
9 </script>
10
11 </BODY>
12 </HTML>
```


Javascript



```
<HTML>
  <BODY style="background-color:green;"
    <h1 style="color:blue;">hello</h1>
    <h1 style="color:red;">hello</h1>

    <script>
  </script>
  </BODY>
</HTML>
```



Javascript



- LE

```
<HTML>
  <BODY>
    <h1 style="color:blue;">hello</h1>
    <h1 style="color:red;">hello</h1>

    <script>
      document.body.style.backgroundColor ="green"
    </script>
  </BODY>
</HTML>
```



Javascript



- Javascript n'a pas la même syntaxe qu'HTML
- Javascript interagit avec HTML

```
<HTML>
  <BODY style="background-color:green;">
    <h1 style="color:blue;">hello</h1>
    <h1 style="color:red;">hello</h1>

    <script>
  </script>
</BODY>
</HTML>
```

```
<HTML>
  <BODY>
    <h1 style="color:blue;">hello</h1>
    <h1 style="color:red;">hello</h1>

    <script>
      document.body.style.backgroundColor ="green"
    </script>
  </BODY>
</HTML>
```

Javascript a modifié la valeur de l'attribut style de la balise BODY

Javascript



- `document.body.style =`
 - Notation objet hiérarchique
 - L'attribut `style` du `body` du `document`
 - Notation des chemins des fichiers
 - Repérage hiérarchique
 - `rueMainguy`
 - Plein de rues Mainguy : laquelle ?
 - `france.bretagne.vannes.tohannic.rueMainguy`
 - Un seul `body`
 - Repérage facile
 - Si plusieurs éléments identiques : comment repérer ?

Javascript



- Changer la couleur du second h1 (en vert par exemple) par javascript

```
<HTML>
  <BODY>
    <h1 style="color:blue;">hello bleu</h1>
    <h1 style="color:red;">hello rouge</h1>

    <script>
      document. ???????
    </script>
  </BODY>
</HTML>
```

hello bleu

hello rouge



en vert ?

Javascript



On rajoute un attribut pour donner un nom à la balise qu'on veut repérer

```
<h1 style="color:blue;">hello bleu</h1>
```

```
<h1 style="color:red;" id="le rouge">hello rouge</h1>
```

id = " "

id veut dire **I**dentifiant

(le nom donné n'a pas d'importance)

Javascript



- En javascript, il faut retrouver la balise qui a un identifiant particulier :
- `document.getElementById()`

```
<HTML>
  <BODY>
    <h1 style="color:blue;">hello bleu</h1>
    <h1 style="color:red;" id="le rouge">hello rouge</h1>

    <script>
      document.getElementById("le rouge").style = "color:green;"
    </script>
  </BODY>
</HTML>
```

Javascript



- getElementById

```
<HTML>
  <BODY>
    <h1 style="color:blue;">hello bleu</h1>
    <h1 style="color:red;" id="le rouge">hello rouge</h1>

    <script>
      document.getElementById("le rouge").style = "color:green;"
    </script>
  </BODY>
</HTML>
```

hello bleu

hello rouge

Le texte n'a pas changé, mais sa couleur, si !

Javascript

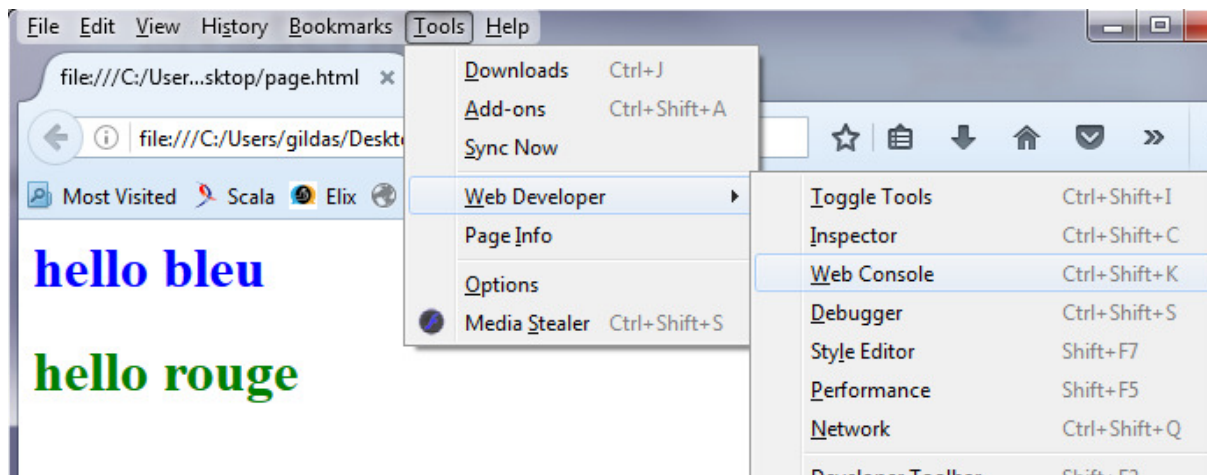


- `document.getElementById(valeur)`
- Recherche une balise qui possède un attribut id de `valeur`.
- Javascript modifie l'attribut style :
- `document.getElementById("le rouge").style = "color:green;"`
- La modification est réalisée sur l'écran, dans le navigateur
- Javascript peut modifier du HTML
 - Master
 - Javascript peut créer du HTML
 - Javascript peut envoyer ou recevoir du HTML sur internet
- On peut donc utiliser ce mécanisme pour voir les résultats de calcul de Javascript
- Mais il y a aussi une console pour les tests

Javascript



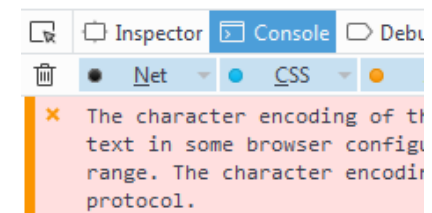
- Console de développement pour les tests



Ouverture console HTML

hello bleu
hello rouge

Choisir « Console » :



Javascript



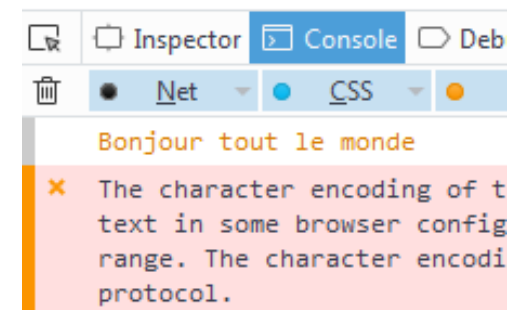
- Javascript peut écrire dans la console :

```
<HTML>
  <BODY>
    <h1 style="color:blue;">hello bleu</h1>
    <h1 style="color:red;" id="le rouge">hello rouge</h1>

    <script>
      document.getElementById("le rouge").style = "color:green;"
      console.log("Bonjour tout le monde")
    </script>
  </BODY>
</HTML>
```

hello bleu

hello rouge



Javascript

hello bleu

hello rouge

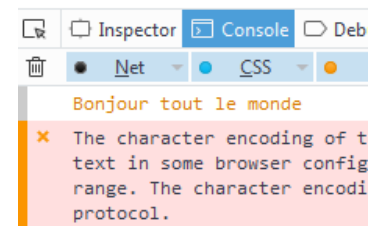


- Erreur : rajouter <meta> ...

```
<HTML>
<meta charset="UTF-8">
<BODY>
  <h1 style="color:blue;">hello bleu</h1>
  <h1 style="color:red;" id="le rouge">hello rouge</h1>

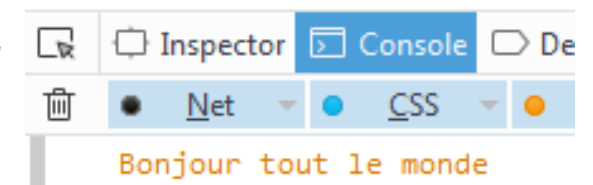
  <script>
    document.getElementById("le rouge").style = "color:green;"
    console.log("Bonjour tout le monde")
  </script>
</BODY>
</HTML>
```

Indique le codage des caractères de la page
(cf Master)



hello bleu

hello rouge



Javascript



- `console.log("texte")`
 - Attention : pas `Console.log`, ni `Console.Log`, ni `console.Log`
 - Majuscules / minuscules importantes

- Affiche dans la console

```
<script>  
    document.getElementById("le rouge").style = "color:green;"  
    console.log("Bonjour tout le monde")  
</script>
```

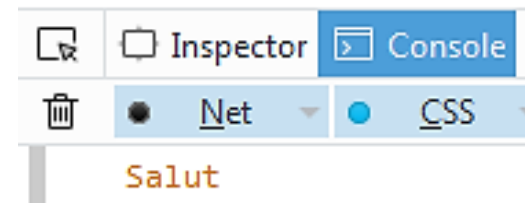
- Points virgules pas importants en fin de ligne
- Les mettre ou pas (différent de Java et de C)
- Une ligne = une instruction
- Décalage dans `<script>`

Javascript



- Variables Javascript

```
var a = "Salut"  
console.log(a)
```



- Création avec var et initialisation (première valeur)
- Nombres : entiers ou réels
- chaînes de caractères
- booléens (true, false)

Javascript



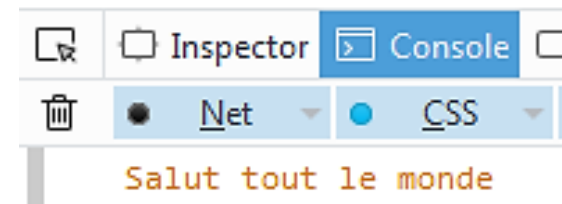
- Variables Javascript : chaînes

```
var a = "Salut"  
var b = "tout le monde"  
console.log(a + b)
```



- Concaténation : on met une chaîne au bout de l'autre

```
var a = "Salut"  
var b = "tout le monde"  
console.log(a + " " + b)
```



Javascript



- Variables Javascript : pas de type fixé
- Attention
 - `var a = "bonjour"`
 - `a = 5`
 - `a = true`
 - `a = 12`
- NE PAS CHANGER DE TYPE
- `a = a + 3`
- ?

Javascript



- Conditionnelle

```
var mot = "sel"  
var grandPetit = "court"
```

```
console.log("le mot " + mot + " est "+ grandPetit )
```

le mot sel est court

- if (condition) instruction

```
var mot = "locomotive"  
var grandPetit = "court"  
if (mot.length > 5) grandPetit = "long"  
console.log("le mot " + mot + " est "+ grandPetit )
```

le mot locomotive est long

mot.length : la longueur de la chaîne de caractères contenue dans mot

Javascript



- conditionnelle

```
If ( condition) instruction1
```

```
If ( condition) {  
    instruction1  
    instruction2  
    instruction3  
}
```

```
If ( condition) {  
    instruction1  
    instruction2  
    instruction3  
} else {  
    instructiona  
    instructionb  
    instructionc  
}
```

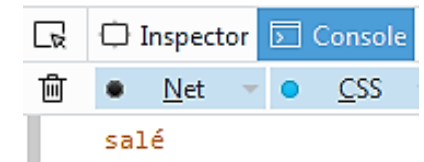
Javascript



- Condition

- if (a==5) ... // ==
- if (a > 5) ...
- if (a < 5) ...
- if (a <= 5) ...
- if (a != 5) ... // différent
- if ((a==5) && (b==5)) ... // et logique
- if ((a == 5) || (b ==7)) .. // ou logique
- if (! (a == 5)) ... // négation logique
- **If (a%2 == 0) .. // si a est pair (le reste de la division de a par 2 est zéro)**

```
var mot = "sel"  
if (mot=="sel") console.log("salé")
```



- Attention, ne pas confondre = et == !!!!

- a = 5
- if (a == 5) ...

Javascript



- Commentaires

```
// ceci est un commentaire sur une ligne
/*
    voilà un commentaire sur
    plusieurs lignes
*/
var mot = "sel"
console.log(mot)
```

Javascript



- Boucles

```
<script>
  var i = 0
  while( i< 10) {
    i = i +1
    console.log(i)
  }
</script>
```



Javascript



- while : tant que

```
<script>  
    var i = 0  
    while( i< 10) {  
        i = i +1  
        console.log(i)  
    }  
</script>
```

Le test est réalisé avant d'entrer dans la boucle

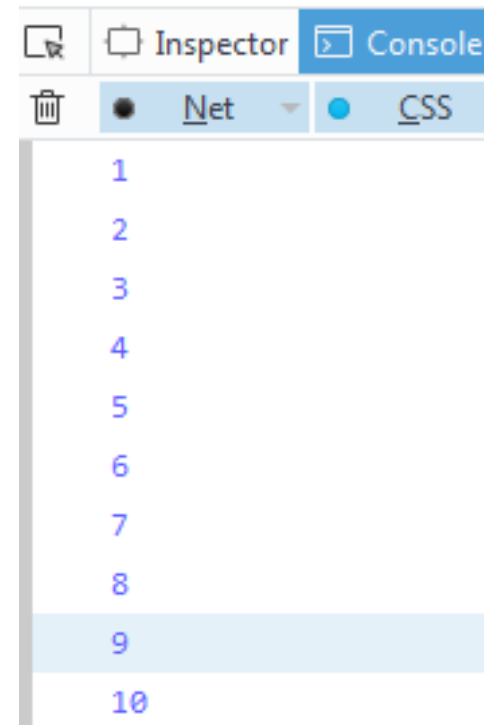
Javascript



- do .. while

```
var i = 0
do {
    i = i + 1
    console.log(i)
} while (i < 10)
```

Le test est réalisé à la fin de la boucle



Javascript



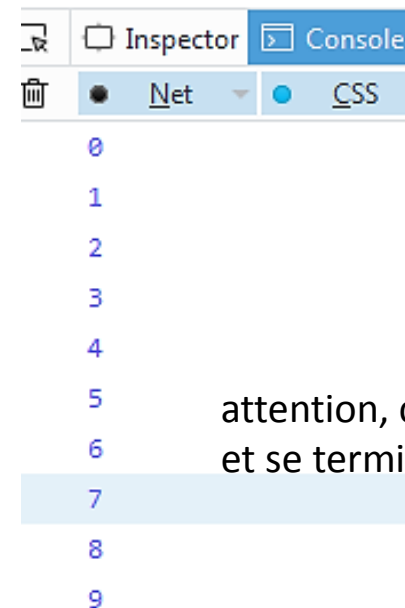
- Boucle for

```
for(var i=0; i< 10; i++) {  
    console.log(i)  
}
```

La boucle commence avec $i=0$ et ne s'exécute tant que $i < 10$
 i est augmenté de 1 à chaque itération ($i++$)

`for(var i=0; i<10; i = i+1) { // est ok également`

For (**initialisation**, **continuation**, **progression**)



attention, commence ici à zéro
et se termine à 9

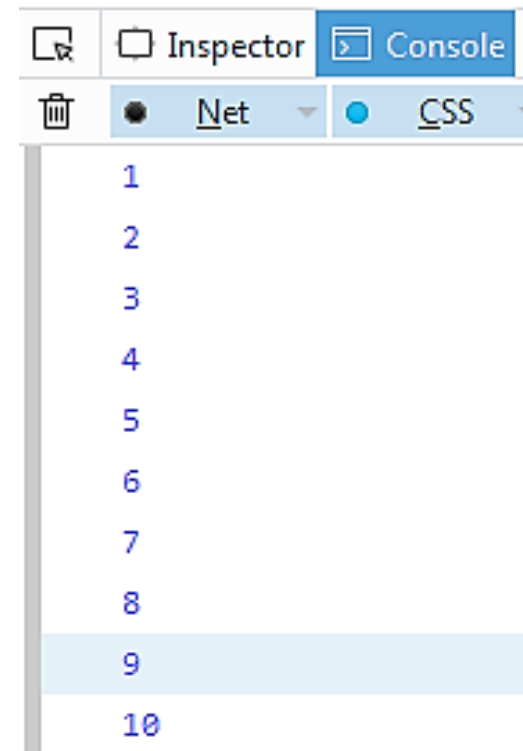
Javascript



- Boucle for

```
for(var i=1; i<= 10; i++) {  
    console.log(i)  
}
```

pour aller de 1 à 10 :

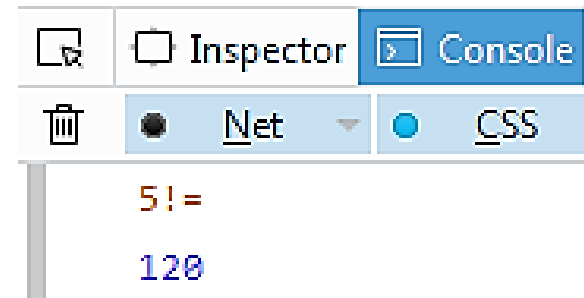


Javascript



- Exemple : factorielle
 - $5! = 5 \times 4 \times 3 \times 2 \times 1$

```
var n = 5
console.log(n+"!=")
var resultat = 1
while (n > 0) {
    resultat = resultat * n
    n = n-1
}
console.log(resultat)
```

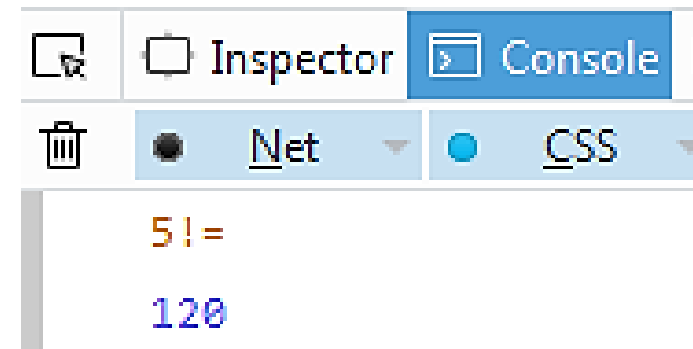


Javascript



- Exemple : factorielle

```
var n = 5
console.log(n+"!=")
var resultat = 1
for(var i=1; i<= n; i = i+1) {
    resultat = resultat *i
}
console.log(resultat)
...
```

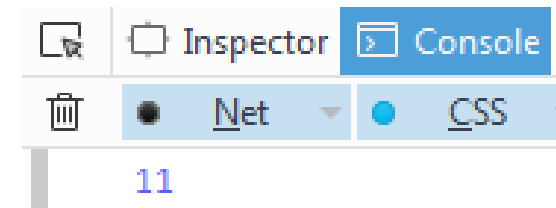


Javascript



- Fonctions
 - Permet de simplifier l'écriture en nommant du code
 - Exemple

```
var a = 5  
var b = 6  
var c = a+b  
console.log(c)
```



Javascript



- Fonction

```
var a = 5
var b = 6

function somme(x,y) {
    return x+y
}

var c = somme(a,b)
var d = somme(c,b)
var e = somme(d,c)

console.log(e)
```



somme(a,b) est remplacé par somme(x,y)
avec x = a = 5 et y = b = 6
Le corps de la fonction est exécuté puis
Le résultat est calculé avec return

Javascript

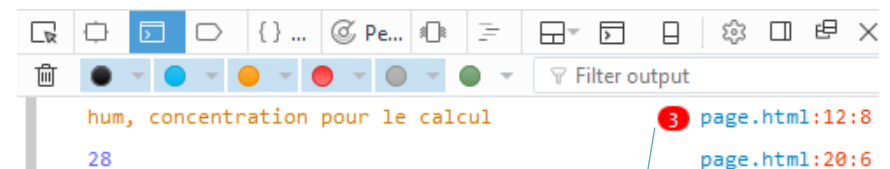


- Fonction

```
function somme(x,y) {  
    return x+y  
}
```

- X et y sont appelés les arguments de la fonction
- On peut avoir autant d'arguments que nécessaire
- Ou pas d'arguments
- La fonction calcule et donne la valeur qui suit return
- Il peut y avoir du code dans la fonction

```
function somme(x,y) {  
    console.log("hum, concentration pour le calcul")  
    return x+y  
}
```



Répété 3 fois

Javascript



- Exemple : fonctionnelle

```
function fact(n) {  
  var resultat = 1  
  for(var i=1; i<= n; i = i+1) {  
    resultat = resultat *i  
  }  
  return resultat  
}  
  
var calcul = fact(5)  
console.log(calcul)
```



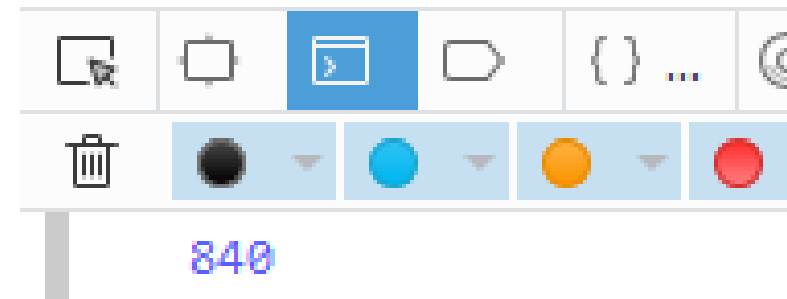
Attention : une fonction calcule un résultat
Une fonction n'affiche pas un résultat
On affiche le résultat d'une fonction
La fonction ne fait QUE calculer une (nouvelle) valeur

Javascript



- Pourquoi ? Supposons que je souhaite calculer $\text{fact}(5) + \text{fact}(6)$

```
function fact(n) {  
  var resultat = 1  
  for(var i=1; i<= n; i = i+1) {  
    resultat = resultat *i  
  }  
  return resultat  
}  
  
var calcul = fact(5) + fact(6)  
console.log(calcul)
```



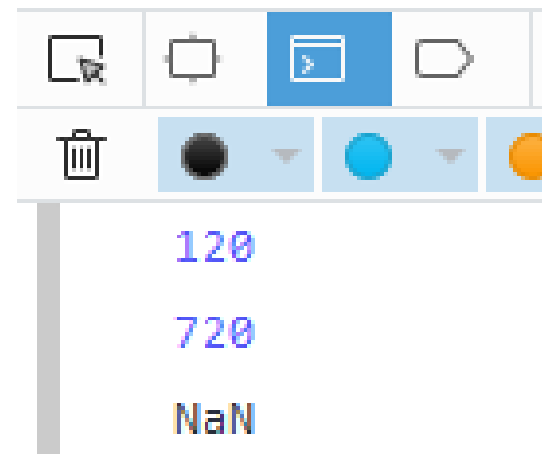
ok

Javascript



- Pas ok

```
function fact(n) {  
  var resultat = 1  
  for(var i=1; i<= n; i = i+1) {  
    resultat = resultat *i  
  }  
  console.log(resultat)  
}  
  
var calcul = fact(5) + fact(6)  
console.log(calcul)
```



Javascript



- Tableaux

- Un tableau est une manière de conserver plusieurs valeurs indicées
- `var a = 5`
 - Une seule valeur
- `var t = []`
 - Un tableau vide (pas d'élément à l'intérieur)
- `var t = ["lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche"]`
- Pour changer une case
 - `t[4] = "Friday"`
 - Commence à l'indice 0
- Pour la longueur :
 - `t.length`

Javascript



- Exemple de tableau

```
var t = ["lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche"]  
for(var i=0; i<t.length; i++) {  
    console.log(t[i])  
}
```

Les jours de la semaine sont contenus dans le tableau t
avec les indices qui vont de 0 à 6 (donc 7 jours)

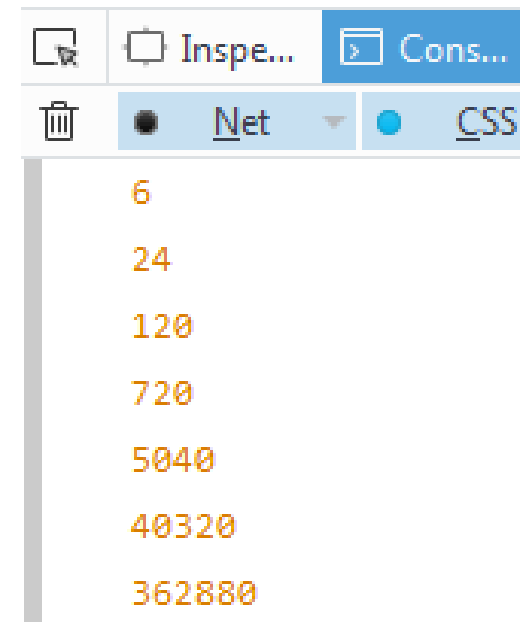


Javascript



- Calculer dans un tableau les factoriels de 3 à 9

```
function fact(n) {  
    var resultat = 1  
    for(var i=1; i<= n; i = i+1) {  
        resultat = resultat *i  
    }  
    return resultat  
}  
  
var t= []  
var indice = 0  
for(var i=3; i<10; i = i+1) {  
    t[indice] = fact(i)  
    indice = indice+1  
}  
  
for(var i=0; i<t.length; i = i+1) {  
    console.log(t[i]+" ")  
}
```



Javascript



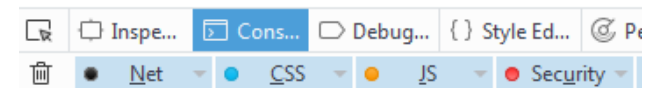
- Javascript a été créé pour interagir avec HTML
 - console.log() limité
- Graphismes
- Balise CANVAS
- Région HTML réservée pour le dessin
- Réserver une taille par exemple 500 x 500

```
<HTML>
  <meta charset="UTF-8">
  <BODY>
    <canvas id="dessin" width="500" height="500">
      Message pour les navigateurs qui ne connaissent pas canvas
    </canvas>

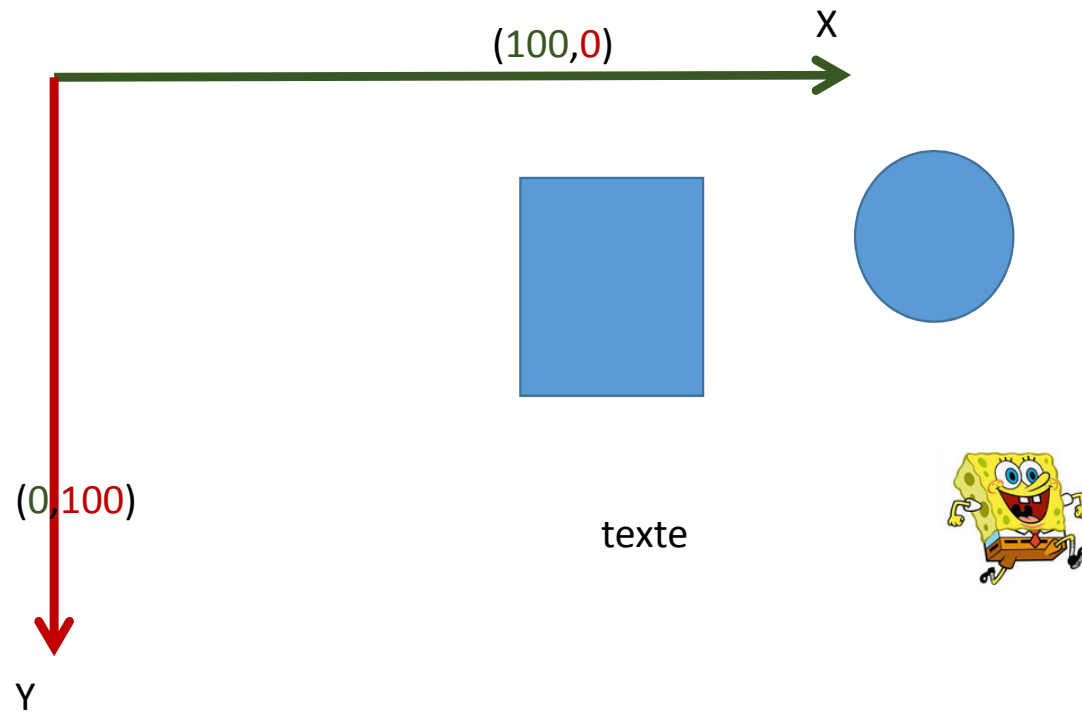
    <script>

  </script>
</BODY>
</HTML>
```

cadre 500x500 invisible (blanc)



Javascript



Javascript

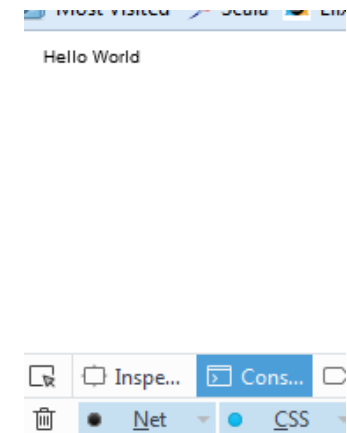


- Dessiner du texte

```
<HTML>
  <meta charset="UTF-8">
  <BODY>
    <canvas id="dessin" width="500" height="500">
      Message pour les navigateurs qui ne connaissent pas canvas
    </canvas>

    <script>
      var cahier = document.getElementById("dessin").getContext("2d");
      cahier.fillText("Hello World",10,10)
    </script>
  </BODY>
</HTML>
```

La variable **cahier** fait le lien entre le **canvas** et le code javascript
L'attribut **id** indique à quel **canvas** on fait référence
(il peut y avoir plusieurs **canvas**)



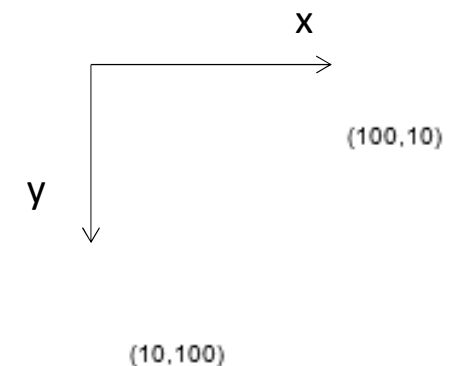
Javascript



- `cahier.fillText("Hello World",10,10)`
- Dessine le texte à la position 10,10 sur le canvas
- exemple

```
<HTML>
  <meta charset="UTF-8">
  <BODY>
    <canvas id="dessin" width="500" height="500">
      Message pour les navigateurs qui ne connaissent pas canvas
    </canvas>

    <script>
      var cahier = document.getElementById("dessin").getContext("2d");
      cahier.fillText("(100,10)",100,10)
      cahier.fillText("(10,100)",10,100)
    </script>
  </BODY>
</HTML>
```



Javascript



- `cahier.fillStyle = "red"`

```
var cahier = document.getElementById("dessin").getContext("2d");
cahier.fillStyle = "green"
cahier.fillText("(100,10)",100,10)
cahier.fillStyle = "red"
cahier.fillText("(10,100)",10,100)
```

`fillStyle` change la couleur du crayon courant

La couleur reste tant qu'on ne la change pas

`cahier.font = "10px Helvetica";` // changer taille et police

Most Visited Scala Elix

(100,10)

(10,100)

Inspe... Cons...
Net CSS

Javascript



- Dessins de base : fillRect
- fillRect(x, y, longueur, hauteur)

```
var cahier = document.getElementById("dessin").getContext("2d");
cahier.fillStyle = "green"
cahier.fillText("(100,10)",100,10)
cahier.fillRect(100,20, 40,10)
cahier.fillStyle = "red"
cahier.fillText("(10,100)",10,100)
cahier.fillRect(10,120, 50, 80)
```

(100,10)



(10,100)



Javascript

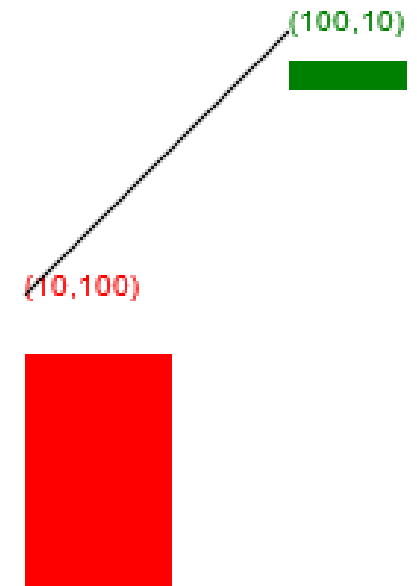


- Dessiner des traits

```
var cahier = document.getElementById("dessin").getContext("2d");
cahier.fillStyle = "green"
cahier.fillText("(100,10)",100,10)
cahier.fillRect(100,20, 40,10)
cahier.fillStyle = "red"
cahier.fillText("(10,100)",10,100)
cahier.fillRect(10,120, 50, 80)
```

```
→ cahier.fillStyle = "blue"
cahier.beginPath()
    cahier.moveTo(100,10)
    cahier.lineTo(10,100)
→ cahier.stroke()
```

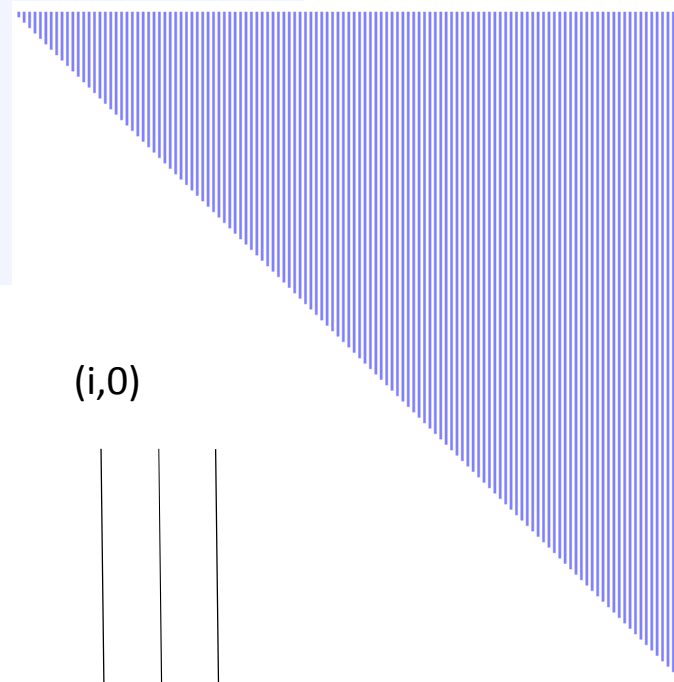
beginPath : début du tracé
moveTo : place le crayon (sans dessiner)
lineTo : déplace le crayon en dessinant
stroke : exécute le tracé



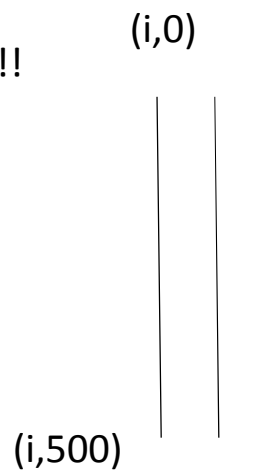
Javascript



```
var cahier = document.getElementById("dessin").getContext("2d");  
  
for (var i=0; i< 500; i = i+4) {  
    cahier.strokeStyle = "blue" ←  
    cahier.beginPath()  
        cahier.moveTo(i,0)  
        cahier.lineTo(i,i)  
    cahier.stroke()  
}
```



Attention, pour la couleur d'un trait (stroke),
Il faut utiliser **cahier.strokeStyle** et non pas cahier.fillStyle !!



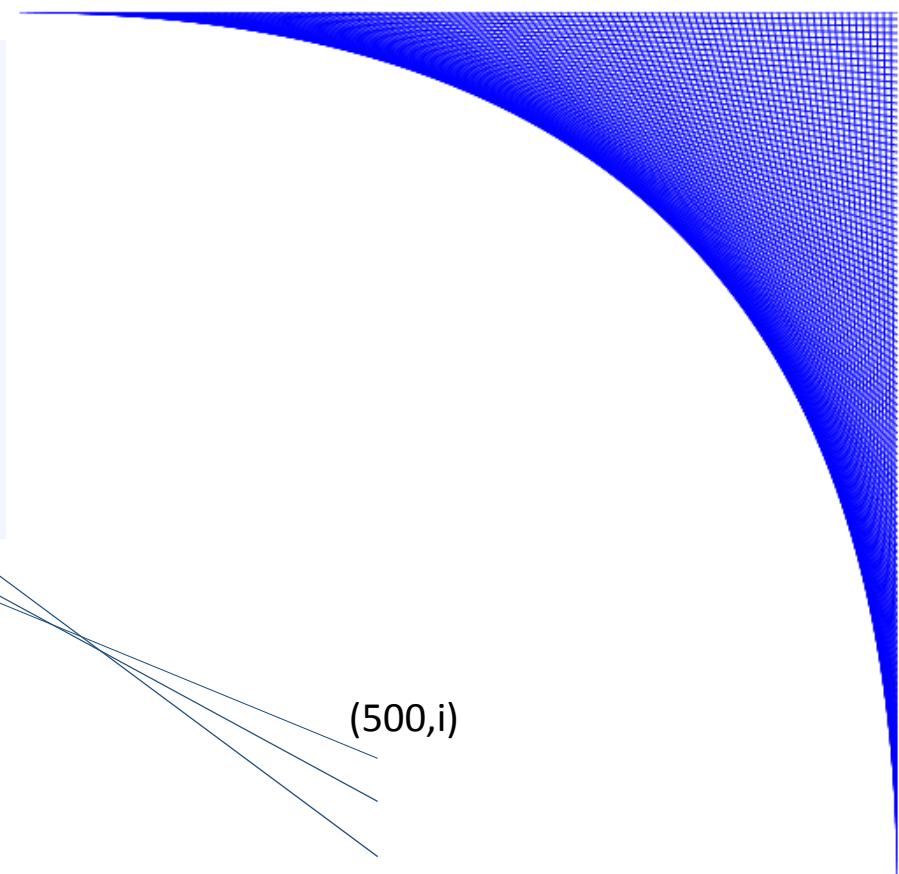
Javascript



```
for (var i=0; i< 500; i = i+4) {  
    cahier.strokeStyle = "blue"  
    cahier.beginPath()  
        cahier.moveTo(i,0)  
        cahier.lineTo(500,i)  
    cahier.stroke()  
}
```

(i,0)

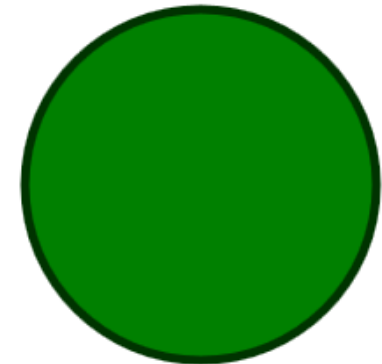
(500,i)



Javascript



```
var cahier = document.getElementById("dessin").getContext("2d")
cahier.beginPath();
cahier.arc(200, 200, 100, 0, 2 * Math.PI, false);
cahier.fillStyle = 'green';
cahier.fill();
cahier.lineWidth = 5;
cahier.strokeStyle = '#003300';
cahier.stroke();
```



arc(centreX, centreY, rayon, angleDebut, angleFin, sensInverseAiguilleMontre)

fill : remplissage

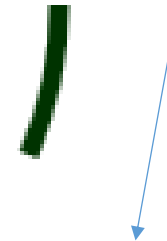
stroke : trait

Javascript



```
var cahier = document.getElementById("dessin").getContext("2d")

cahier.beginPath();
  cahier.arc(200, 200, 100, 0, 0.4, false);
  cahier.fillStyle = 'green';
  cahier.fill();
  cahier.lineWidth = 5;
  cahier.strokeStyle = '#003300';
cahier.stroke();
```



arc(centreX, centreY, rayon, angleDebut, angleFin, sensInverseAiguilleMontre)

fill : remplissage

stroke : trait

Javascript



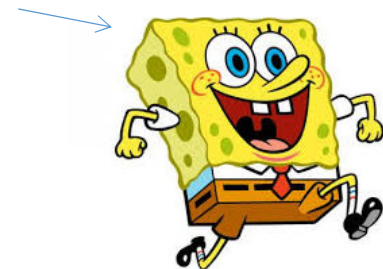
- Dessiner une image

- Création d'un objet par Javascript
- Image n'est pas un type de base
- C'est un objet de la classe Image
- On utilise new pour créer une nouvelle valeur

```
var cahier = document.getElementById("dessin").getContext("2d");  
  
var mon_image = new Image();  
mon_image.src = "bob.jpg";  
cahier.drawImage(mon_image, 100, 100);
```

- L'image bob.jpg doit exister dans le répertoire du fichier html

Position (100,100)

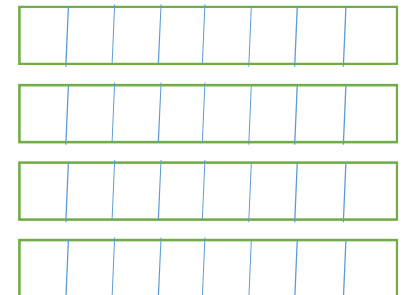
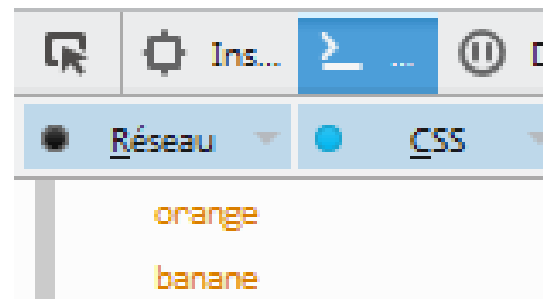


Javascript



- Tableau à 2 dimensions
 - Un tableau a une dimension peut contenir une ligne
 - Un tableau a deux dimensions contient plusieurs lignes
 - Un tableau de tableaux

```
var t = []  
for(var i=0; i<4; i++) {  
    t[i] = []  
}  
  
t[0][0] = "orange"  
t[1][0] = "banane"  
  
console.log(t[0][0])  
console.log(t[1][0])
```



Javascript



- Tableau à 2 dimensions

- Un tableau a deux dimensions contient plusieurs lignes
 - Un tableau de tableaux

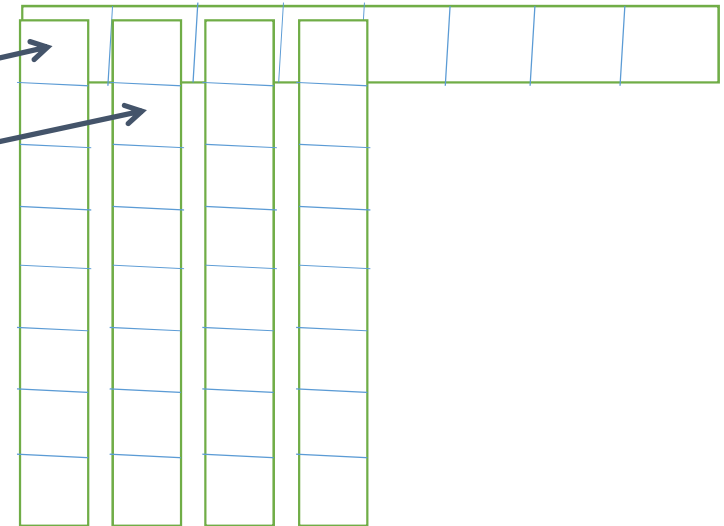
```
var t = []  
for(var i=0; i<4; i++) {  
    t[i] = []  
}
```

```
t[0][0] = "orange"
```

```
t[1][0] = "banane"
```

```
console.log(t[0][0])
```

```
console.log(t[1][0])
```



Javascript



- Programmation orientée objet
 - Javascript :
 - langage à prototypes
 - Surcouche langage objet à classe
 - Un objet contient des **propriétés**
 - C'est une boîte qui contient des variables qui lui sont propres
 - `menier.adresse`
 - `menier.age`
 - Chaque propriété est une variable définie par *var*
 - On peut demander à un objet d'exécuter des **méthodes**
 - `menier.faisLaVaisselle()`
 - `menier.chante("la marseillaise")`
 - Chaque méthode est une fonction définie par *function*

Javascript



- Programmation orientée objet
 - Une classe
 - Est le plan d'un objet
 - Quelles sont les **propriétés** d'un objet ?
 - Quelles sont ses **méthodes** ?

```
class Personne {  
    ...  
}  
  
var g = new Personne ()
```

- Il faut définir une classe AVANT de l'utiliser
- Pour créer une instance de classe, il faut utiliser *new*
- Exemple : création d'une nouvelle instance
- Classe vide : pas de propriété, pas de méthode

Javascript



- Programmation orientée objet
 - Quand on crée un objet par *new*, on peut fabriquer son contenu
 - *constructor*
 - Fabrique les propriétés de l'objet

```
class Personne {  
    constructor(leNom, lePrenom) {  
        this.nom = leNom  
        this.prenom = lePrenom  
    }  
}
```

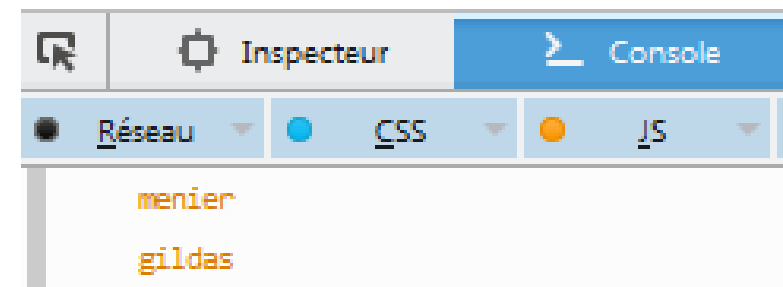
- `var g = new Personne("menier", "gildas")`
- g possède les propriétés *nom* et *prenom*
- Ces propriétés ont les valeurs "menier" et "gildas"

Javascript



- Programmation orientée objet

```
class Personne {  
    constructor(leNom, lePrenom) {  
        this.nom = leNom  
        this.prenom = lePrenom  
    }  
}  
  
var g = new Personne("menier", "gildas")  
  
console.log(g.nom)  
console.log(g.prenom)
```



Javascript



- Programmation orientée objet
 - Les propriétés se créent dans le *constructor*
 - *this* indique que ces propriétés ne concernent que l'objet créé

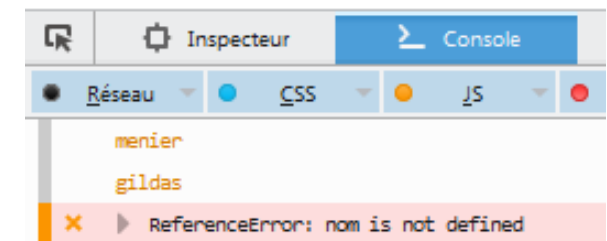
```
class Personne {  
    constructor(leNom, lePrenom) {  
        this.nom = leNom  
        this.prenom = lePrenom  
    }  
}  
  
var g = new Personne("menier", "gildas")  
  
console.log(g.nom)  
console.log(g.prenom)
```

Javascript



- Programmation orientée objet
 - Les propriétés se créent dans le *constructor*
 - *this* indique que ces propriétés ne concernent que l'objet créé

```
class Personne {  
  constructor(leNom, lePrenom) {  
    this.nom = leNom  
    this.prenom = lePrenom  
  }  
}  
  
var g = new Personne("menier", "gildas")  
  
console.log(g.nom)  
console.log(g.prenom)  
  
console.log(nom)  
console.log(prenom)
```



Javascript



- Programmation orientée objet
 - Les méthodes se définissent comme des fonctions
 - SANS le mot clé *function*
 - À l'intérieur de *class*

```
class Personne {
  constructor(leNom, lePrenom) {
    this.nom = leNom
    this.prenom = lePrenom
  }

  sePresente() {
    console.log("bonjour, je suis "+ this.nom + " " +this.prenom)
  }
}

var g = new Personne("menier", "gildas")

g.sePresente()
```



Javascript



- Programmation orientée objet

- obj.a

- Veut dire : la propriété 'a' de l'objet 'obj'
 - On peut lire cette propriété
 - `console.log(obj.a)`
 - `var t= obj.a`
 - On peut modifier cette propriété
 - `obj.a = 56`

- obj.b()

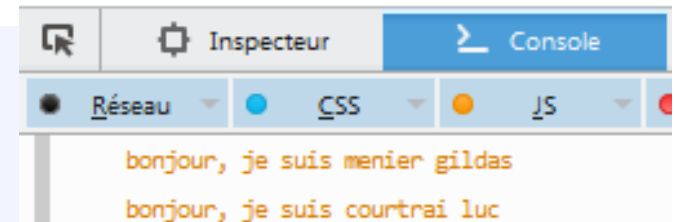
- Veut dire : exécuter la fonction ou la méthode `b()` de l'objet obj
 - Comme une autre fonction, peut renvoyer un résultat `var t= obj.b()`
 - Si elle est définie avec plusieurs arguments : `obj.b(45, "hello", false)`

Javascript



- Programmation orientée objet
 - La programmation objet permet de simplement créer plusieurs instances
 - Réutilisation du code de la classe

```
class Personne {  
  constructor(leNom, lePrenom) {  
    this.nom = leNom  
    this.prenom = lePrenom  
  }  
  
  sePresente() {  
    console.log("bonjour, je suis "+ this.nom + " " + this.prenom)  
  }  
}  
  
var g = new Personne("menier", "gildas")  
g.sePresente()  
var p = new Personne("courtrai", "luc")  
p.sePresente()
```

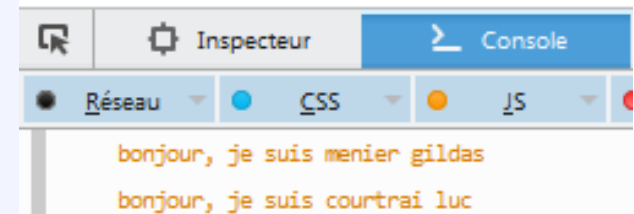


Javascript



- Programmation orientée objet
 - Il est possible de placer les instances dans un tableau

```
class Personne {  
  constructor(leNom, lePrenom) {  
    this.nom = leNom  
    this.prenom = lePrenom  
  }  
  
  sePresente() {  
    console.log("bonjour, je suis "+ this.nom + " " +this.prenom)  
  }  
}  
  
profs = []  
profs[0] = new Personne("menier", "gildas")  
profs[1] = new Personne("courtrai", "luc")  
  
for(var i=0; i<2; i++) {  
  profs[i].sePresente()  
}
```



Javascript



- Exemple : création d'un objet graphique

```
var chat = new Image()  
chat.src = "chatmarche.png"  
cahier.drawImage(chat, 20, 20)
```



chatmarche.png

Javascript



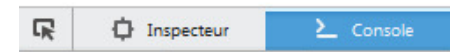
- Exemple : création d'un objet graphique

```
var cahier = document.getElementById("dessin").getContext("2d")

class Lutin {
  constructor(nomImage) {
    this.image = new Image()
    this.image.src = nomImage
    this.x = 0
    this.y = 0
  }

  dessineEn(positionX, positionY) {
    this.x = positionX
    this.y = positionY
    cahier.drawImage(this.image, this.x, this.y)
  }
}

var minou = new Lutin("chatmarche.png")
minou.dessineEn(50, 50)
```

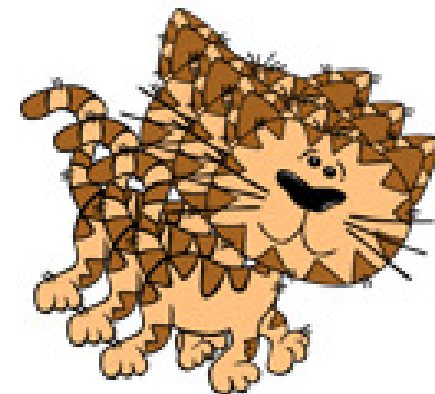


Javascript



- Exemple : création d'un objet graphique

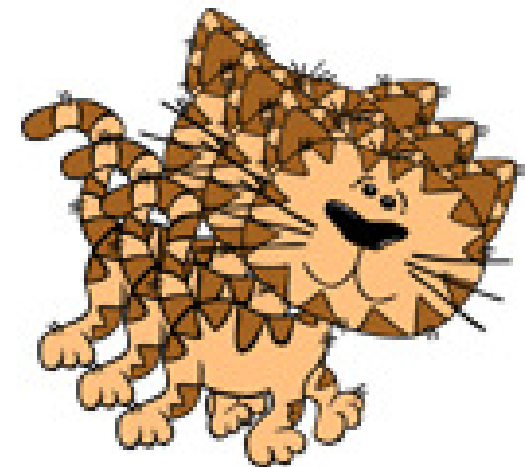
```
var minou = new Lutin("chatmarche.png")  
minou.dessineEn(50,50)  
minou.dessineEn(60,60)  
minou.dessineEn(70,70)
```



Javascript



- Remarque
 - Transparence
 - Format png
 - On peut rendre une couleur transparente
 - Sinon, Jpg : pas de transparence
 - Voir Photoshop / gimp etc..

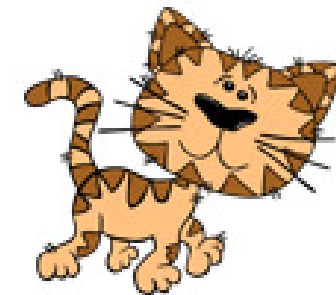


Javascript



- Exemple : création d'un objet graphique

```
class Lutin {  
  constructor(nomImage) {  
    this.image = new Image()  
    this.image.src = nomImage  
    this.x = 0  
    this.y = 0  
  }  
  
  position(positionX, positionY) {  
    this.x = positionX  
    this.y = positionY  
  }  
  
  dessine() {  
    cahier.drawImage(this.image, this.x, this.y)  
  }  
}  
  
var minou = new Lutin("chatmarche.png")  
minou.position(20, 20)  
minou.dessine()
```



une méthode : une seule action

(dessineEn : déplace et dessine)

(position : déplace)

(dessine : une seule action)

Javascript



- Exemple : création d'un objet graphique

```
var dort = new Lutin("chatdort.png")  
dort.position(100,100)  
dort.dessine()
```

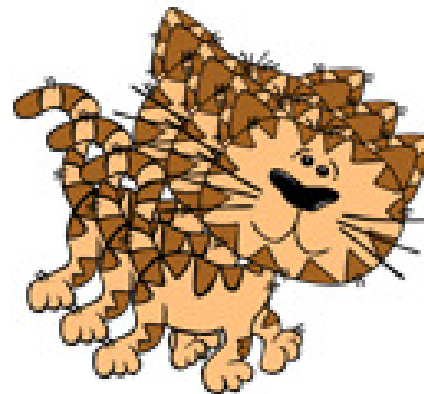


Javascript



- Animation

- Dessiner le lutin
- Changer sa position
- Redessiner le lutin
- Changer sa position
- Etc..



- Effacer l'ancienne position

- Effacer l'écran
- Dessiner tous les lutins
- Efface l'écran
- Dessiner tous les lutins etc..

Javascript



- Effacer le *canvas*
 - Tracer un rectangle blanc de la taille du canvas
 - clearRect

```
var minou = new Lutin("chatmarche.png")
for(var i=0; i< 10; i++) {
    cahier.clearRect(0,0,500,500)
    minou.position(i,0)
    minou.dessine()
}
```

in du programme seulement

- Pas pendant !
- On ne voit que la version finale de l'image
- Pourquoi ? Navigateur

Javascript



- Solution

- Relancer régulièrement une fonction
- *Spécifique* à Javascript
 - *setInterval*
 - Lance une fonction à intervalles réguliers

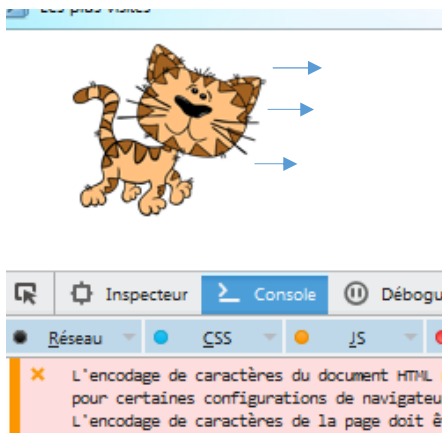
```
function ticTac() {  
    console.log("tic tac")  
}  
  
setInterval(ticTac, 2000)
```

- Affiche "tic tac" toutes les 2s (2000 ms)
- Ne s'arrête jamais

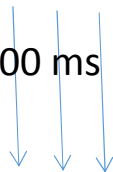
Augmente tout seul



- Animer le chat



toutes les 1000 ms



```
class Lutin {
  constructor(nomImage) {
    this.image = new Image()
    this.image.src = nomImage
    this.x = 0
    this.y = 0
  }

  position(positionX, positionY) {
    this.x = positionX
    this.y = positionY
  }

  dessine() {
    cahier.drawImage(this.image, this.x, this.y)
  }

  avance() {
    this.x = this.x + 5
  }
}

var minou = new Lutin("chatmarche.png")

function animeChat() {
  cahier.clearRect(0, 0, 500, 500)
  minou.avance()
  minou.dessine()
}

setInterval(animeChat, 1000)
```



Chaque appel de 'avance'
déplace le chat vers la droite
de 5 pixels

Toutes les 1000 ms
déplace le chat vers la droite
de 5 pixels



Javascript



- *setInterval*(fonctionSpeciale, n)
 - Appel régulier, toutes les n millisecondes
- *setTimeout*(fonctionSpeciale2, n)
 - Un seul appel au bout de n millisecondes
- On peut en faire plusieurs en même temps (pas conseillé > 3)
 - var chrono = setInterval(fonctionSpeciale, 1000)
 - clearInterval(chrono) : arrêt du chrono
 - var compteRebours = setTimeout(fonctionSpeciale2, 1000)
 - clearTimeout(compteRebours)

Javascript



- Attention
 - Plus le délais est court, plus le navigateur est ralenti !
 - Ne pas mettre un délais à 1ms ou inférieur
 - Blocage du navigateur
 - Plus assez de temps pour se gérer tout seul !

Javascript



- Jouer un son

- Créer une balise audio dans la partie HTML
- On peut en créer plusieurs et les rendre invisibles

```
<audio id="sonMiaou" src="miaou.mp3" preload="auto" autobuffer>  
</audio>
```

- Faire un lien Javascript
- Bien nommer chaque son différemment

```
var lecteurMiaou = document.getElementById("sonMiaou")
```

- Jouer le son

```
lecteurMiaou.play()
```

Javascript



- Jouer une musique (avec boucle)
 - Rajouter **loop** dans la balise <audio>
 - Dès que le son est joué, boucle automatiquement

```
<audio id="musique" src="musiqueDeFond.mp3" preload="auto" autobuffer loop >  
</audio>
```

- Ne pas oublier de faire **play()** pour lancer la musique
- Autres méthodes :
 - stop()
 - pause()
- mp3, wav, ogg