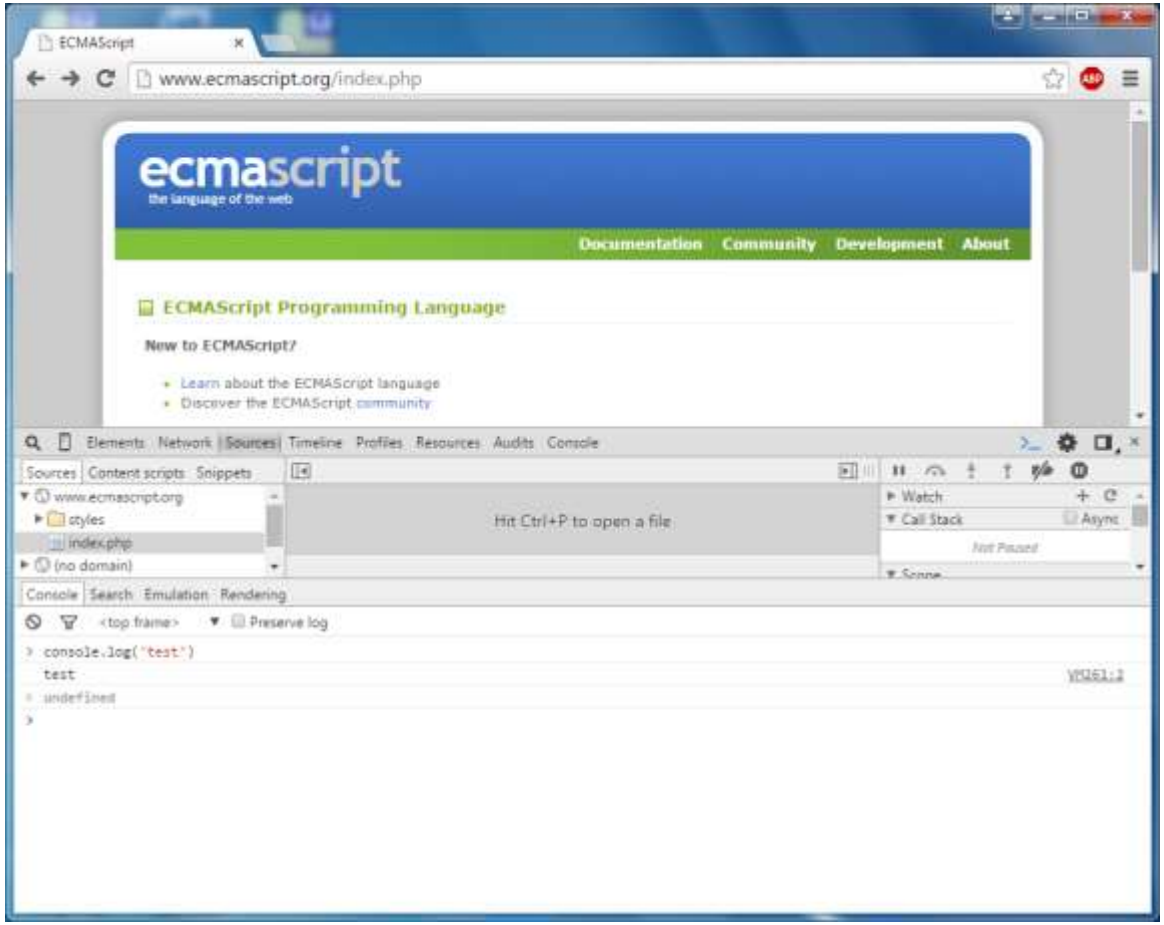
# Javascript dans le browser

# Javascript est évidemment le langage de programmation du browser.

- Son véritable nom est "EcmaScript"
  - Nom sans marque associée
  - Organisme de standardisation :
    - http://www.ecmascript.org/index.php
- Version actuelle : EcmaScript 6
  - Mais la version la plus répandue : EcmaScript 5
  - Ce cours se concentre sur EcmaScript 5

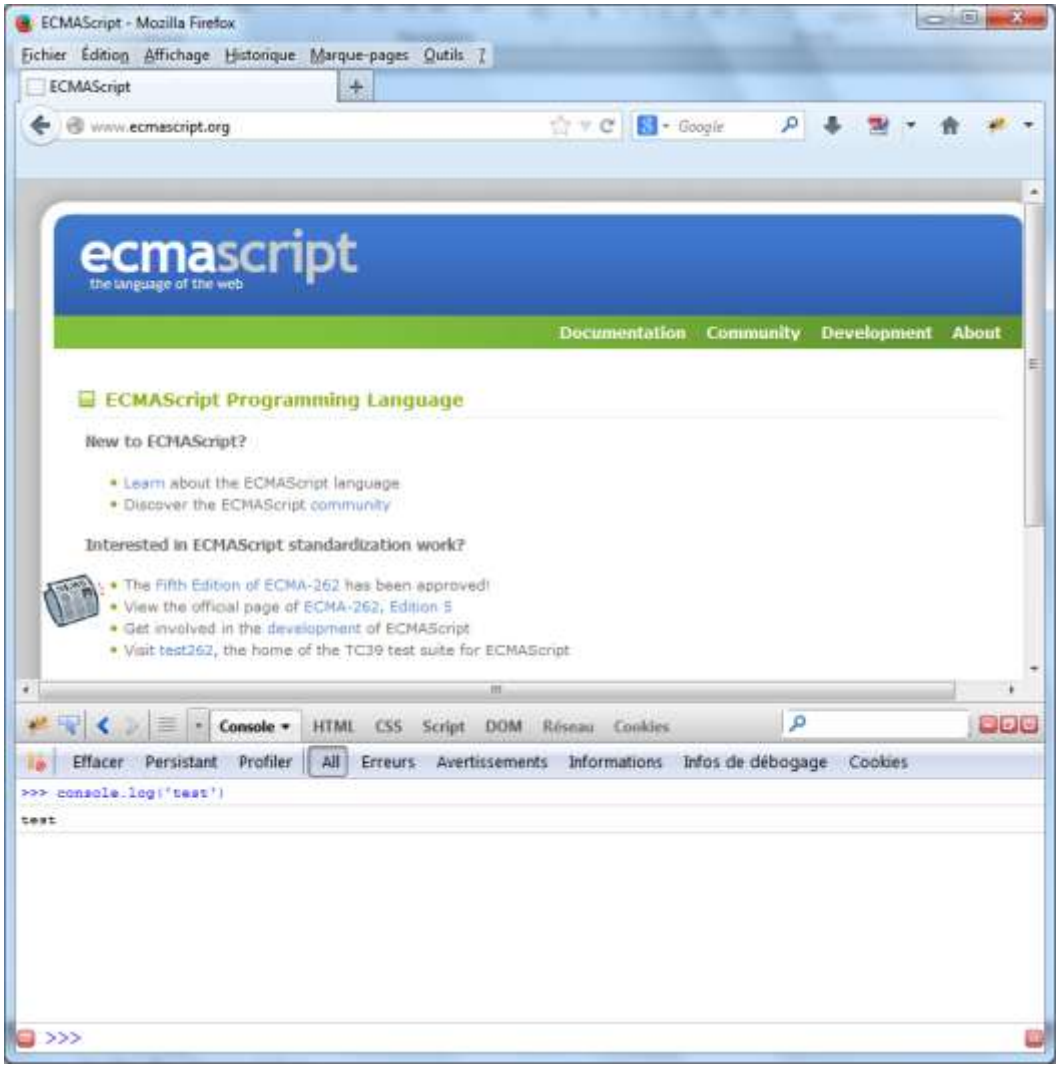
# Javascript dans le browser

- Dans Chrome : F12



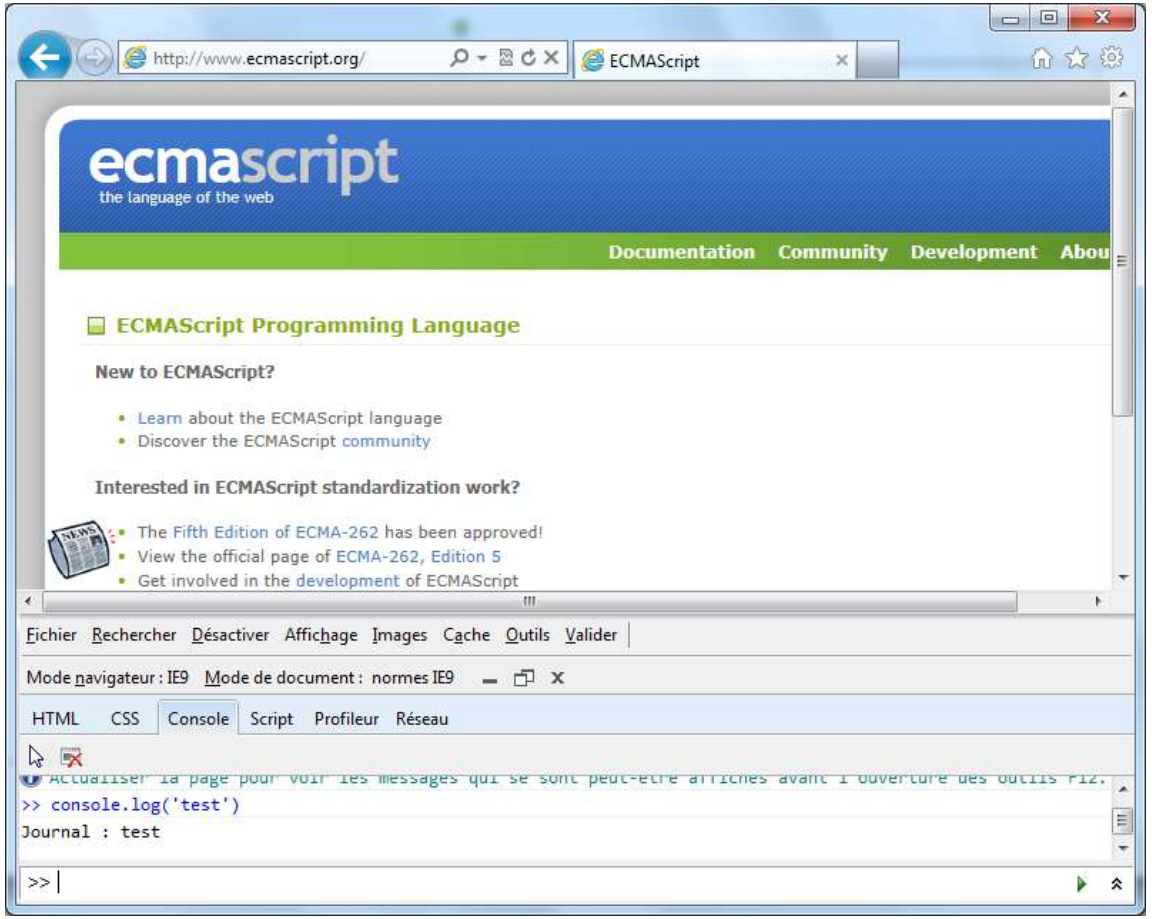
# Javascript dans le browser

- Dans Firefox : F12



# Javascript dans le browser

- Internet Explorer : F12



# Javascript dans le browser

```
<html>
...
<script type="application/javascript">
console.log('test');
</script>
...
</html>
```

# Javascript dans le browser

<html>

...

<script type="application/javascript" src="test.js"></script>

...

</html>

Avec test.js qui contient

console.log('test')

# Javascript dans le browser

- console.log ne va pas nous mener très loin
- Le store de Javascript est prérempli de variables permettant d'effectuer des choses utiles:
  - navigator : objet contenant de l'information sur le browser lui-même, permettant de l'identifier.
  - window : objet manipulant la fenêtre du browser même.
  - document : (===window.document) objet manipulant le corps de l'HTML de la page.
  - location : (===window.location) objet représentant l'URL de la page.

# Exemples location

- location : objet qui représente l'URL de la page.
  - location.href===cette URL.
  - location.reload() : recharge la page.
  - location.replace('https://www.google.com') : navigue à cette URL.

- Documentation : http://www.w3schools.com/jsref/obj_location.asp

# Exemples navigator

- navigator : objet qui représente le navigateur.
  - navigator.appName :
    - "Netscape" pour Chrome et Firefox
    - "Microsoft Internet Explorer"  pour IE
  - navigator.appVersion :
    - Chrome : "5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.130 Safari/537.36"
    - Firefox : "5.0 (Windows)"
    - IE : "5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)"
- Documentation : http://www.w3schools.com/jsref/obj_navigator.asp

# document

- document s'adresse au contenu HTML de la page.
  - document.writeln('coucou') : remplace le contenu de la page (<html>...</html>) pour y écrire la chaîne donnée en paramètre

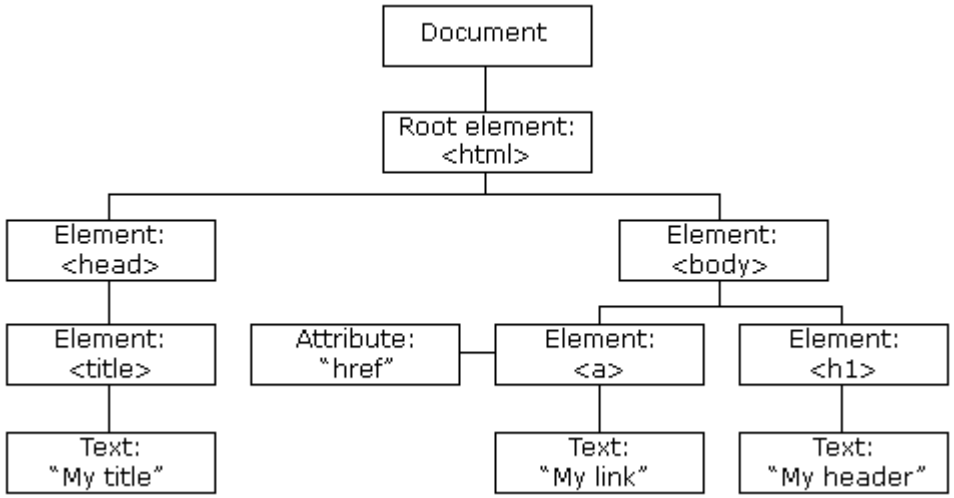
# **D**ocument **O**bject **M**odel

http://www.w3schools.com/js/js_htmldom.asp

- Le navigateur crée une représentation objet de la page.
- Quand cet objet est manipulé en Javascript, la page est automatiquement mise à jour.

# DOM

- document.body.style.backgroundColor='red'



document.body.style.backgroundColor='red'

- document => point de départ du DOM
- document.body => objet représentant le body de l'HTML
- document.body.style => objet représentant l'attribut style du body.
- document.body.style.backgroundColor => attribut qui représente l'attribut CSS correspondant, après camel-case-ation de ce dernier

document.body.children[0].style.backgroundColor='blue'

- document.body.children : tableau contenant les objets DOM des descendants de body
- dans cet exemple, on change la couleur de fond du premier descendant du body

- http://www.w3schools.com/jsref/dom_obj_document.asp

# DOM

- Le DOM a donc une structure en arbre.
- Chaque noeud représente un élément HTML.
- Certains attributs correspondent aux attributs HTML.

```
document.body.onclick=function() {console.log('click');}
```

- D'autres attributs/fonctions permettent de manipuler de diverses manières le noeud, et/ou d'accéder aux autres noeuds.

# En pratique...

- Traverser un arbre pour trouver un noeud particulier est très peu souple.

  document.body.children[1].children[0]...

  - Si la structure de l'HTML change, ceci devra être changé..
- Il existe donc des raccourcis facilitant l'accès à des noeuds particuliers.

# document.getElementById(...)

- Le paramètre est l'id d'un élément de la page.
  - Attention d'après la norme, les ids doivent être unique au sein d'une page !

```
<div id='toto'>...</div>

var toto=document.getElementById('toto');
....
```

# document.getElementsByName(...)

- Le paramètre est la valeur de l'attribut name d'un ou plusieurs éléments de la page.
  - Contrairement à l'id, le name peut être utilisé sur plusieurs éléments.
  - Cette méthode renvoie un tableau d'éléments.

```
<input type='radio' name='radiototo'>
<input type='radio' name='radiototo'>
<input type='radio' name='radiototo'>

var elements=document.getElementsByName('radiototo')
```

# document.getElementsByClassName(...)

- Le paramètre est le nom d'une classe css.
  - Cette méthode renvoie un tableau d'éléments qui ont au moins cette classe CSS.

```
<input type='radio' class='radiototo'>
<input type='radio' class='radiototo'>
<input type='radio' class='radiototo'>

var els=document.getElementsByClassName('radiototo')
```

# document.getElementsByTagName(...)

- Le paramètre est le nom d'un tag HTML.
  - Cette méthode renvoie un tableau d'éléments HTML de ce tag.

```
<input type='radio'>
<input type='radio'>
<input type='radio'>

var els=document.getElementsByTagName('input')
```

# Evénements

- Nous avons vu comment modifier la page dynamiquement.
- Mais ce serait bien de pouvoir réagir aux actions de l'utilisateur.
  - On enregistre des fonctions qui seront appelées lors d'actions spécifiques de l'utilisateur
  - Action spécifique de l'utilisateur = un événement sur un élément du DOM.

# Enregistrement via l'HTML

- http://www.w3schools.com/jsref/dom_obj_event.asp
- Les éléments HTML possèdent des attributs affectables à une fonction javascript.
  - Peut se faire en HTML

    ```
    <body onclick="function() {console.log('click');}">
    ```

  - Peut se faire en Javascript

    ```
    document.body.onclick=function() {console.log("click");}
    ```

# Exemples : événements souris

| Event | Description |
|---|---|
| onclick | The event occurs when the user clicks on an element |
| oncontextmenu | The event occurs when the user right-clicks on an element to open a context menu |
| ondblclick | The event occurs when the user double-clicks on an element |
| onmousedown | The event occurs when the user presses a mouse button over an element |
| onmouseenter | The event occurs when the pointer is moved onto an element |
| onmouseleave | The event occurs when the pointer is moved out of an element |
| onmousemove | The event occurs when the pointer is moving while it is over an element |
| onmouseover | The event occurs when the pointer is moved onto an element, or onto one of its children |
| onmouseout | The event occurs when a user moves the mouse pointer out of an element, or out of one of its children |
| onmouseup | The event occurs when a user releases a mouse button over an element |

# Inconvénients

- La valeur de l'attribut est unique.
  - Si on change cette valeur, la fonction préalablement affectée à cet événement ne sera plus appelée.
  - L'HTML peut référencier une fonction Javascript qui n'est pas encore chargée. Ceci peut conduire à des disfonctionnements si l'utilisateur réagit trop vite.

# Enregistrement via le DOM

- Pour circonvenir à ces problèmes : enregistrement via le DOM

```
document.body.addEventListener("click",
function() {console.log("click")});
```

- http://www.w3schools.com/js/js_htmldom_eventlistener.asp

# Evénements dans le futur

- Les navigateurs sont mono-threadés
  - en apparence du point de vue du Javascript en tout cas
- Quand le JS s'exécute, rien d'autre ne le fait
  - pas de chargement de la page
  - pas de rafraîchissement de la page
  - pas d'autre JS
  - pas d'interaction avec l'utilisateur
  - rien...

# setTimeout

- On pourrait souhaiter faire des actions dans le futur
  - animations
  - timeouts
  - ...
- setTimeout(f,t) : événement temporel
  - f est une fonction qui sera appelée après t millisecondes