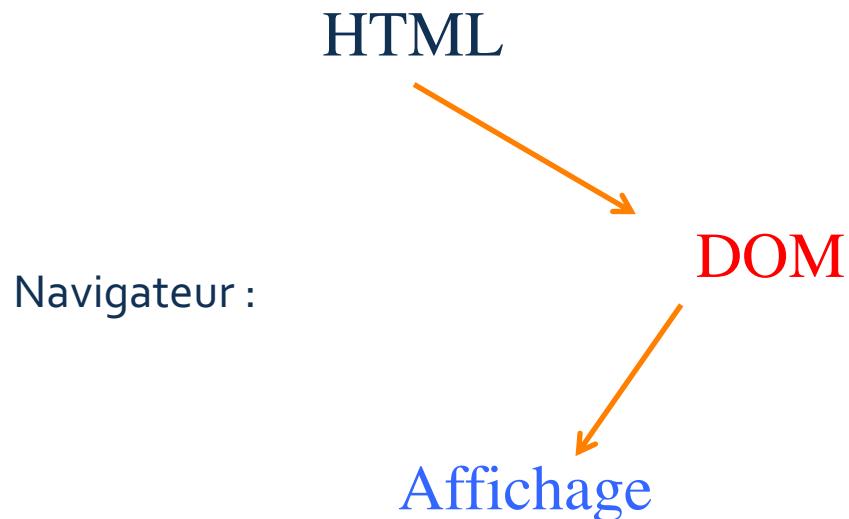




Javascript
AJAX

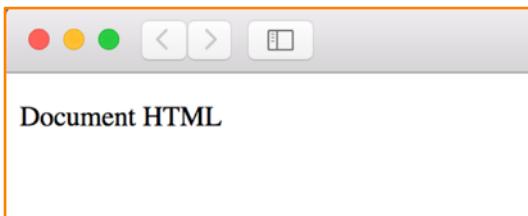
Javascript

- + Comment modifier dynamiquement le contenu d'une page HTML ?
 - + Sur la base d'une action de l'utilisateur
 - + Parce que des informations dans la page ont changé
 - + ...



Javascript

```
<!DOCTYPE html>
<html lang="fr-fr">
  <head>
    <title>Document HTML</title>
    <meta charset="UTF-8"/>
  </head>
  <body>
    <p>Document HTML</p>
  </body>
</html>
```



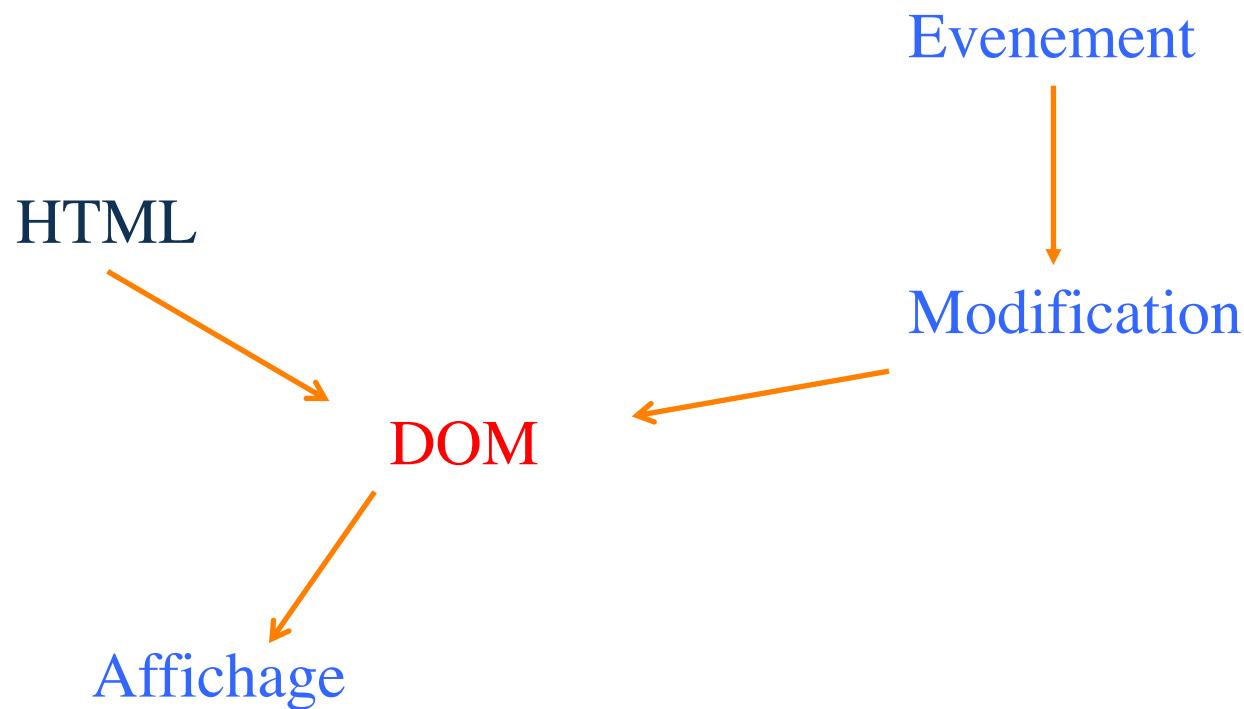
```
<!DOCTYPE html>
<html lang="fr-fr">
  <head>
    <title>Document HTML</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <p>Document HTML</p> = $0
  </body>
</html>
```

▼ Propriétés

▼ <p>

- accessKey: ""
- align: ""
- assignedSlot: null
- ○ attributes: NamedNodeMap {len}
- baseURI: "file:///Users/kwy"
- childElementCount: 0
- ○ childNodes: NodeList [#text]
- ○ children: HTMLCollection []
- ○ classList: DOMTokenList {len}
- className: ""
- ...

Javascript



Javascript

- + Langage de programmation = ECMAScript
 - + Typage faible (pas obligatoire) et dynamique (s'adapte à la situation)
 - + Interprété ou compilé à la volée
 - + Orienté objet à prototype (constructeurs, méthodes)
- + TypeScript (TS) évolution de Javascript proposée par Google.
 - + Typage plus fort
 - + Quasiment compatible avec Javascript

Javascript

- + Dans Javascript
 - + Elements de langages de programmation
 - + Variables
 - + Fonctions
 - + Instructions de contrôle
 - + Tests
 - + Boucles
 - + Fonctions permettant de manipuler le DOM
 - + Fonctions permettant d'interagir avec l'utilisateur
 - + Fonctions permettant de communiquer
 - + Interception d'evenements (trigger / déclencheur)

Syntaxe Javascript



Syntaxe Javascript

+ L'élément de référence : **document**

document permet

- + D'accéder aux informations contenues dans la page
- + D'accéder aux informations sur le navigateur
- + D'accéder aux fonctions permettant de manipuler le contenu du document

Syntaxe Javascript

- + Afficher un message dans une page html :

```
<!DOCTYPE html>
<html lang="fr-fr">
    <head>
        <title>Document HTML</title>
        <meta charset="UTF-8"/>
    </head>
    <body>
        <p>Document HTML</p>
        <script type="text/javascript">
            <!--
                //masquage du script pour les anciens navigateurs
                document.writeln("Nous sommes le 20/01/2019");
            -->
        </script>
    </body>
</html>
```



Document HTML

Nous sommes le 20/01/2019

Syntaxe Javascript

- + Et si je dois faire la même chose à plusieurs endroits ?
 - + Définir une **fonction**

```
function maFonction() {  
    Ce que doit faire la fonction  
}
```
 - + La fonction est définie au choix :
 - + Dans la partie entête (head) de la page
=> utilisation uniquement dans cette page
 - + Dans un fichier à part qui va être inclus dans la page : fichier javascript avec une extension js
=> permet une utilisation là où on le souhaite

Syntaxe Javascript

+ Fonction déclarée dans la page

```
<!DOCTYPE html>
<html lang="fr-fr">
    <head>
        <title>Document HTML</title>
        <meta charset="UTF-8"/>
        <script type="text/javascript"> <!--
            function afficherDate() {
                document.writeln("Nous sommes le 20/01/2019");
            }
        --> </script>
    </head>
    <body>
        <p>Document HTML</p>
        <script type="text/javascript"> <!--
            afficherDate();
        --> </script>
    </body>
</html>
```



Document HTML

Nous sommes le 20/01/2019

Syntaxe Javascript

+ Fonction déclarée dans un fichier distinct

```
<!DOCTYPE html>
<html lang="fr-fr">
  <head>
    <title>Document HTML</title>
    <meta charset="UTF-8"/>
    <script type="text/javascript" src="test.js" ></script>
  </head>
  <body>
    <p>Document HTML</p>
    <script type="text/javascript">
      <!--
        afficherDate();
      -->
    </script>
  </body>
</html>
```



Document HTML

Nous sommes le 20/01/2019

test.js

```
function afficherDate() {
  document.writeln("Nous sommes le 20/01/2019");
}
```

Syntaxe Javascript

+ Les variables en Javascript

- + Une variable porte un nom = suite de caractères sans espaces, et si possible sans accents, commençant par une lettre.
- + Typage dynamique (en fonction du contexte)

+ Définir une variable

```
var maVariable; // visible à partir de la déclaration  
let variable; // idem mais limité au bloc contenant  
const variable; // comme let, mais constante
```

+ Mettre une valeur dans une variable

```
var maVariable = 5;  
var mavariable2 = maVariable + 1  
var maVariable3 = "test = "+ maVariable2;
```

Syntaxe Javascript

- + Opérateur
 - + Utilisation des opérateurs classiques
 - + Arithmétique : + - * /
 - + Compareurs : == != < <= > >=
 - + Comparateur + test de typage : === !==
 - ====
 - vérifier le type et valeur
 - « 1 » === 1 faux
 - + Booléen : && || !

Syntaxe Javascript

+ Instructions

+ Terminées par ;

Le passage à la ligne est parfois suffisant.

+ Commentaires :

```
/* ... */  
//
```

+ Tests :

```
if (cond) {instructions_si_vrai;}
```

```
if (cond) {instructions_si_vrai;} else {instructions_si_faux;}
```

```
switch (element) case ... :... break;... default: ... }
```

+ Boucles

```
for (init; condition; incrément) { ... }
```

```
while (condition) { ... }
```

Syntaxe Javascript

+ Fonctions

+ Création

```
function nomFonction1(arguments) {
```

```
    ...
```

```
}
```

```
function nomFonction2(arguments) {
```

```
    ...
```

```
    return value;
```

```
}
```

+ Appel

```
nomFonction1(paramètres);
```

```
var maVariable = nomFonction2(paramètres);
```

Syntaxe Javascript

+ Javascript et le passage de paramètres

Une fonction javascript peut avoir en paramètre

- + Des entiers, réels, ...
Passage par valeur
- + Des objets
- + Des fonctions

Syntaxe Javascript

+ Objets

Object, Array, Date, ...

+ Création = opérateur new

+ Appel de méthode : objet.methode(paramètres)

+ this = objet manipulé

```
function afficherDate() {  
    var date = new Date();  
    alert("Nous sommes le " + date.getDate() + "/" + (date.getMonth() + 1) + "/" + date.getFullYear());  
}
```

Syntaxe Javascript

- + Créer ses propres objets

- + A partir d'Object

```
var pierre = new Object();
pierre.nom = "Durand";
pierre.prenom = "Pierre";
```

- + A partir d'un bloc
= JSON

```
var pierre = {
    nom : "Durand",
    prenom : "Pierre",
    nomprenom : function() {
        return this.nom+" "+this.prenom;
    }
}
```

- + A partir d'une fonction

```
function personne(n, p) {
    this.nom = n;
    this.prenom = p;
}
var pierre = new personne( "Durand", "Pierre");
```

Syntaxe Javascript

+ Le typage des données dans TypeScript

+ Déclaration de variables

```
var maVariable: string;
```

+ Paramètres et résultats de fonctions

```
function maFonction(data: string) : Object { ... }
```

+ Création de classes

```
class maClasse { ... }
```

+ Constructeurs

```
constructor( parametre) { ... }
```

+ Méthodes

+ extends pour l'héritage

+ ...

Syntaxe Javascript

+ Callback et Promise

- + Callback : une fonction qui sera évaluée en temps opportun, plus tard (par exemple lorsqu'un évènement survient)
- + Promise : un fonction **asynchrone** qui retourne un object de type Promise (promesse de résultat)
 - + Resolve : le résultat est obtenu (utilisation de la méthode then)
 - + Reject : une erreur s'est produite (utilisation de catch)

subscribe

Syntaxe Javascript

+ Exemple utilisation de Promise

```
maFonction()
  .then(function(data) {
    // do something with data
  })
  .catch(function(err) {
    // do something with error
  });

```

Le programme appelle maFonction. Lorsque le résultat est établi

- + S'il est valide on appelle la fonction dans then
- + S'il a généré une erreur, on appelle la fonction dans catch

NB : il est possible d'enchaîner les then et catch pour effectuer des actions en cascade

Syntaxe Javascript

+ Interagir avec la page :

- + Pour l'instant, on n'a pas gagné grand chose : peu d'interaction avec l'utilisateur.
- + Comment changer le comportement de la page en fonction des actions de l'utilisateur ?
Utilisation de la programmation évènementielle
= réagir en fonction des évènements qui surviennent.

Utilisation de **déclencheurs** = demander à un élément de la page de réagir en fonction de ce qui se passe

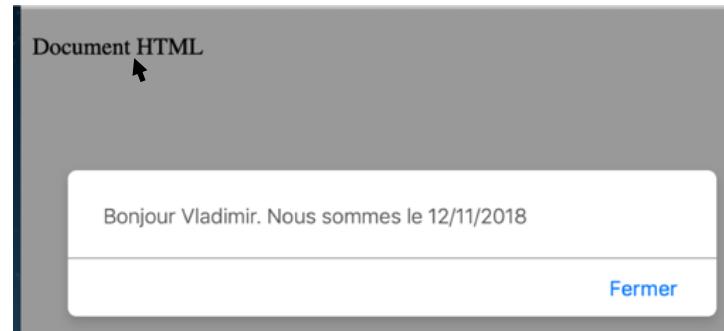
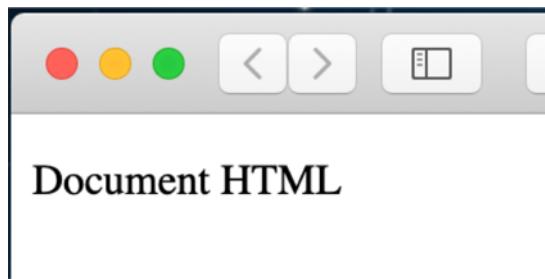
Syntaxe Javascript

```
<!DOCTYPE html>
<html lang="fr-fr">
  <head>
    <title>Document HTML</title>
    <meta charset="UTF-8"/>
    <script type="text/javascript" src="test.js" ></script>
  </head>
  <body>
    <p onmouseover="afficherDate('Vladimir', new Date());">Document HTML</p>
  </body>
</html>
```

onmouseover = lorsque la souris passe au dessus de l'élément

```
function afficherDate(nom, date) {
  alert("Bonjour " + nom + ". Nous sommes le " + date.getDate() + "/" + (date.getMonth() + 1) + "/" + date.getFullYear());
}
```

Syntaxe Javascript



La souris passe au dessus du texte

Syntaxe Javascript

- + Les déclencheurs
 - + Réagissent sur la base d'évènements
 - + Souris (clic, déplacement, ...)
 - + Clavier
 - + ...
 - + Dépendent des éléments sur lesquels ils sont placés
 - + Certains ne sont disponibles QUE en HTML5

Syntaxe Javascript

- + Quelques déclencheurs :
 - + onClick : clic sur le bouton de la souris
 - + onDoubleClick : double-clic sur le bouton de la souris
 - + onMouseDown : bouton de la souris enfoncé sur un élément input
 - + onMouseUp : bouton de la souris relâchée sur un élément input
 - + onMouseOver : curseur survolant l'objet
 - + onMouseOut : curseur quittant le survol de l'objet
 - + onKeyDown : touche du clavier enfoncée
 - + onKeyPress : la touche du clavier enfoncée est relâchée
 - + onKeyUp : touche du clavier relâchée
 - + onLoad : l'élément est chargé dans la page (ouverture du document)
 - + onSelect : du texte a été sélectionné dans l'élément
 - + onSubmit : Soumission d'un formulaire
 - + onChange : l'élément a été modifié (input, select, ...)
 - + ...

Syntaxe Javascript

- + Fonctionnement des déclencheurs :
 - + Utilisé comme un attribut dans la balise
 - + La valeur de l'attribut déclencheur est la suite d'instructions à exécuter => il peut s'avérer utile d'utiliser des fonctions
- + Les déclencheurs **retournent** une valeur (true ou false) indiquant si l'événement doit être traité ou non. Par défaut, on considère que true est retourné.
 - + Si true est retourné, l'événement est considéré comme non traité et le traitement par défaut s'applique
 - + Si false est retourné, l'événement est considéré comme traité et le traitement par défaut ne s'applique pas.

Syntaxe Javascript

- + Et si on modifiait les éléments du DOM ?
 - + **document** : l'objet global qui est à la racine du DOM.
Tout est dans **document**.
 - + **this** : référence à l'objet dans lequel est le déclencheur

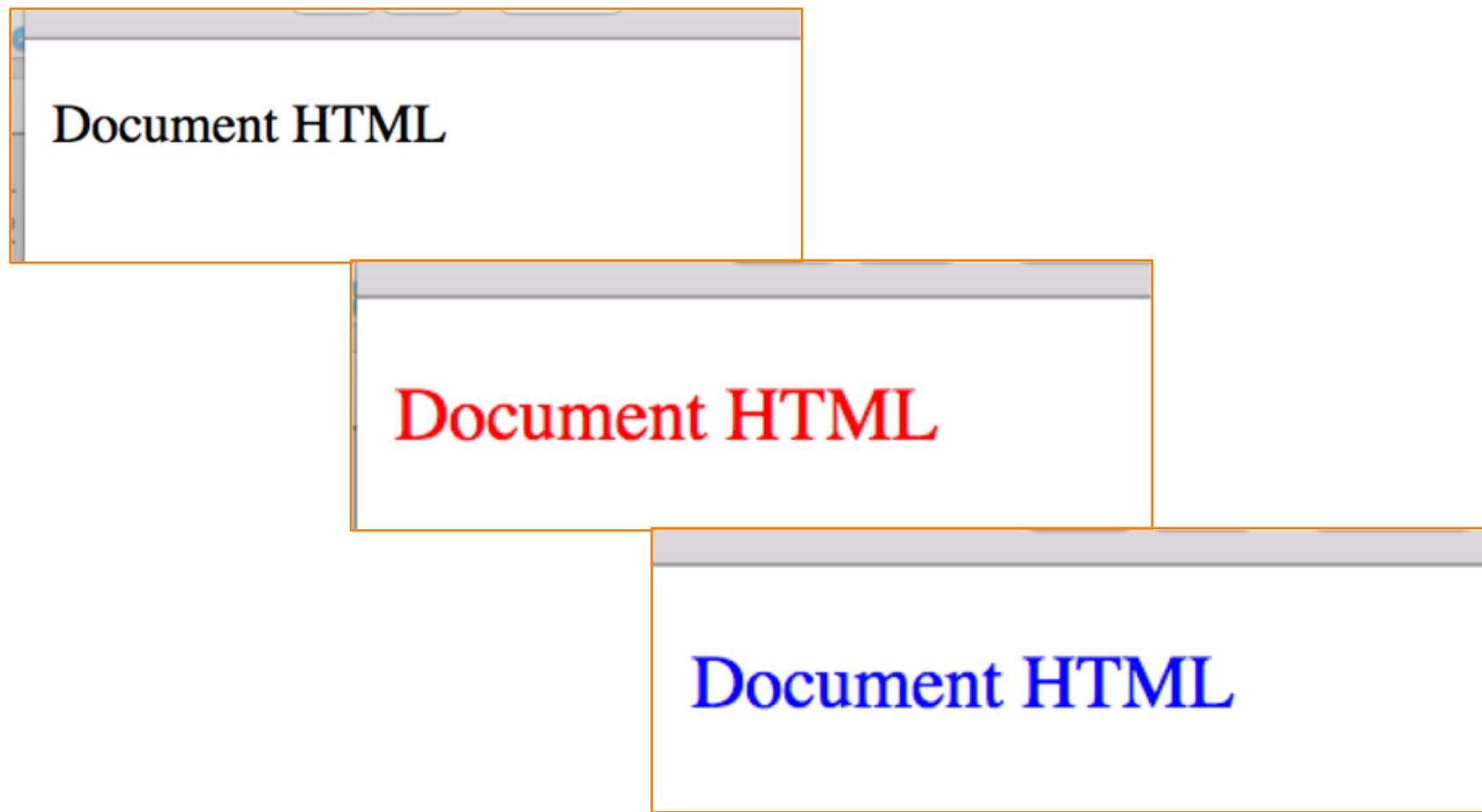
Syntaxe Javascript

```
function afficher(element, couleur) {  
    element.style.color = couleur;  
}
```

```
<!DOCTYPE html>  
<html lang="fr-fr">  
    <head>  
        <title>Document HTML</title>  
        <meta charset="UTF-8"/>  
        <script type="text/javascript" src="test.js" ></script>  
    </head>  
    <body>  
        <p onmouseover="afficher(this, 'red');" onmouseout="afficher(this, 'blue');>Document  
HTML</p>  
    </body>  
</html>
```



Syntaxe Javascript



Syntaxe Javascript

- + Que peut-on utiliser / modifier dans les éléments ?
Pratiquement tous les attributs
 - + class = l'attribut class de l'élément
 - + classList = liste des classes de l'élément
 - + style = le style qui lui est appliqué
 - + id = l'attribut id de l'élément
 - + textContent : le contenu d'une balise de texte
 - + value : la valeur contenue dans un input, un select, ...
 - + selectedIndex= l'index sélectionné dans un select
 - + tagName = le nom de la balise
 - + ...

Syntaxe Javascript

- + Que peut-on utiliser / modifier dans les éléments ?
 - + style = l'attribut style de l'élément
 - + style.color = couleur du texte
 - + style.backgroundColor : couleur de fond
 - + ...
 - + display : la façon dont l'élément est affiché (gestion des bordures)
 - + top, left, right, bottom, width, height, ...

Syntaxe Javascript

- + Accéder / Manipuler les éléments de la page ?
 - + id : élément unique dans la page
`document.getElementById(idDeElement)`
l'objet correspondant à l'id en question
 - + La balise
`document.getElementsByTagName (nomBalise)`
-> tous les objets correspondant à la balise mentionnée
 - + firstChild, lastChild
Le premier / dernier élément fils d'un objet
 - + nextSibling, previousSibling
L'élément de même niveau situé après, avant
 - + ...
 - + null = pas d'objet correspondant

Syntaxe Javascript

```
<!DOCTYPE html>
<html lang="fr-fr">
  <head>
    <title>Document HTML</title>
    <meta charset="UTF-8"/>
    <script type="text/javascript" src="test.js" ></script>
  </head>
  <body>
    <p id="montexte">Document HTML</p>
    <p onmouseover="changeTexte('je change');" onmouseout="changeTexte('je reviens');">je
change tout</p>
  </body>
</html>
```

```
function changeTexte(leTexte) {
  var element = document.getElementById("montexte");
  element.textContent = leTexte;
}
```

Syntaxe Javascript

Document HTML

je change tout

je change

je change tout

je reviens

je change tout

Syntaxe Javascript

- + Modifier le DOM pour associer les déclencheur

Comment sont définis les déclencheurs dans le DOM ?

The screenshot shows a browser's developer tools DOM inspector. On the left, the DOM tree is displayed:

```
<!DOCTYPE html>
<html lang="fr-fr">
  <head>_</head>
  <body>
    <p id="monTexte">Document HTML</p>
    <p onmouseout="changeTexte(2); onmouseover="changeTexte(1);>je change tout</p> = $0
  </body>
</html>
```

Two blue arrows point from the highlighted code in the DOM tree to the properties panel on the right. The properties panel shows the following for the second `p` element:

- `onmouseout: function(event)`
- `onmouseover: function(event)`
- `arguments: null`
- `caller: null`
- `length: 1`
- `name: "onmouseover"`
- `prototype: onmouseover {}`
- `Prototype Function`
- `onmouseup: null`

Syntaxe Javascript

+ Et si on modifiait les fonctions dans le DOM...

```
<!DOCTYPE html>
<html lang="fr-fr">
  <head>
    <title>Document HTML</title>
    <meta charset="UTF-8"/>
    <script type="text/javascript" src="test.js" ></script>
  </head>
  <body>
    <p id="montexte">Document HTML</p>
    <p id="declencheur">je change tout</p>
    <script>
      <!--
      onloadHandler();
      -->
    </script>
  </body>
</html>
```

```
function changeTexte(valeur) {
  var element = document.getElementById("montexte");
  if (valeur == 1) {
    element.textContent = "Je change";
  } else {
    element.textContent = "Je reviens";
  }
}

function declencheurOver(event) {
  changeTexte(1);
}
function declencheurOut(event) {
  changeTexte(2);
}

function onloadHandler() {
  var pDeclenche = document.getElementById("declencheur");
  pDeclenche.onmouseover = declencheurOver;
  pDeclenche.onmouseout = declencheurOut;
}
```

Syntaxe Javascript

- + On pousse un peu les modifications du DOM
 - + Créer un nouvel élément:
`var elt = document.createElement(nomBalise);`
 - + Créer un nouvel élément texte :
`var textContent = document.createTextNode(text);`
 - + Ajouter un élément à la liste des fils d'un nœud :
`unNoeud.appendChild(nouveauFils);`
 - + Ajouter un nœud avant l'un des nœuds d'un élément
`unNoeud.insertBefore(nouveauFils, avantFils);`
 - + Retirer un nœud de la liste des fils
`unNoeud.removeChild(ancienFils);`

JSON



JSON

- + JavaScript Object Notation
Format de données spécifique à Javascript

Utilisé pour mémoriser des informations et permettre à javascript de les manipuler plus facilement.

JSON

- + Quelques éléments de syntaxe :

- + { ... } : objet
- + Champ:valeur
- + [...] : tableau

- + Exemples :

- + { nom : "valeur", prenom : "valeur" }
- + { personne: { nom : "valeur", prenom : "valeur" } }
- + ["Bleu", "Rouge", "Vert"]

JSON

+ Quelques fonctions à connaître

```
// Crédation
myObj = {name: "John", age: 31, city: "New York"};
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

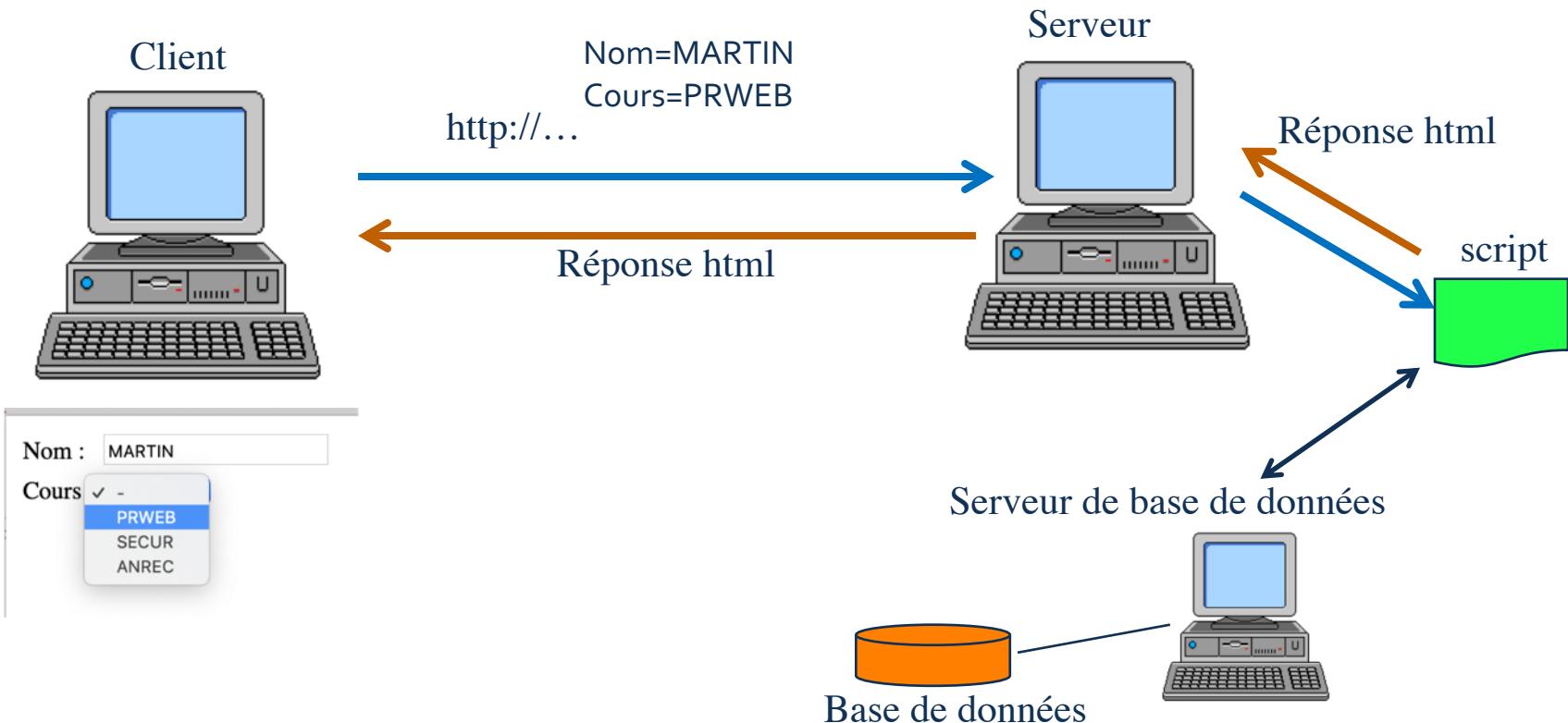
// Restauration
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
var myName = obj.name;
```

AJAX



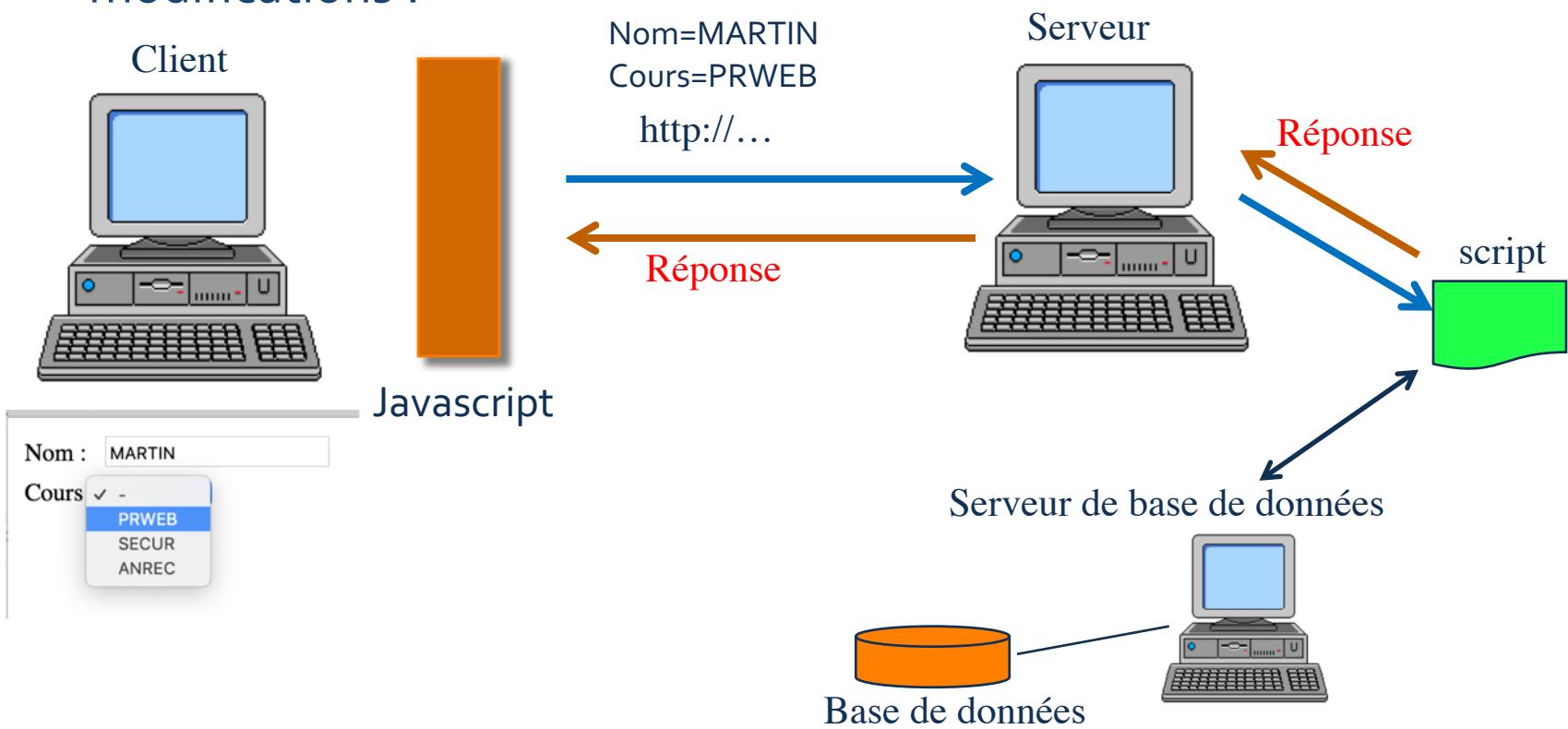
AJAX

+ Fonctionnement d'un formulaire



AJAX

- + Est-il possible de ne pas recharger la page mais de faire des modifications ?



AJAX

- + AJAX = Asynchronous JavaScript and XML
 - + La page appelle une fonction javascript
 - + Javascript lance un appel au serveur avec des paramètres
 - + Le serveur traite la demande et retourne des informations
 - + Javascript récupère les informations et modifie la page en conséquence

AJAX

- + Fonctionnement
 - + Création d'un objet XMLHttpRequest
 - + Ajout des paramètres dans l'objet XMLHttpRequest
 - + Communication avec le serveur pour exécuter un script avec les informations de l'objet XMLHttpRequest
 - + En mode GET ou POST
 - + En mode
 - + Synchrone
Attente de la réponse pour poursuivre le script
 - + ou Asynchrone
Utilisation d'une fonction de callback pour traiter la réponse
Le script se poursuit.
 - + Réception du résultat
 - + Interprétation du résultat
 - + Modification de la page en conséquence

AJAX

- + La récupération des informations
 - + En mode texte
 - + Au format XML
 - Utilisation du DOM pour traiter la réponse
 - + JSON = JavaScript Object Notation
 - Format des données Javascript
 - Analyse du texte via des fonctions Javascript pour reconstituer les variables

AJAX

+ Utilisation avec JQUERY

+ Jquery ?

Bibliothèque Javascript

<http://jquery.com/download/>

+ Exemple d'appel AJAX

```
function refresh() {
    $.ajax({
        url: "/prweb/ajaxItemCategory.php",
        data: {
            code: "pandemonium"
        },
        method: "POST",
        success: function( result ) {
            console.log("ok");
        },
        error: function (resultat, statut, erreur) {
            console.log("error");
        }
    });
}
```

