Cours 5
Javascript

HEI 2021 / 2022

Introduction

Qu’est-ce que Javascript

Introduction

- Javascript est un langage de script exécuté au sein des navigateurs.
  - Il peut également servir de langage serveur avec NodeJS.
- Il s'agit d'une implémentation du standard ECMAScript.
- La majorité des sites web modernes utilisent Javascript et les différents navigateurs ont chacun un moteur d'exécution du langage.

À quoi ça sert ?

- Javascript permet d'apporter une dynamisation des pages web directement dans le navigateur :
  - Manipulation des éléments de la page
  - Gestion des événements utilisateur
  - Mise à jour partielle de page
  - Validation, calcul exécutés sans passer par le serveur
  - …

Intégration

Lier du code Javascript à une page HTML

Script interne

- Le code Javascript peut directement être écrit dans une balise `script` de la page HTML.

```
<script type="application/javascript">
    alert("Le javascript s'exécute.");
</script>
```

Script externe

- Le code Javascript est la plupart du temps écrit dans un fichier à part avec l'extension `.js`. Le fichier est ensuite lié à la page HTML avec la balise `script`.

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

Emplacement de la balise script

- La balise `script` peut être ajoutée n'importe où dans la page HTML.

- Deux pratiques émergent :
  - À la fin du `body` pour que le Javascript soit chargé après le chargement du HTML
  - Dans la balise `head`

Syntaxe

À quoi ça ressemble ?

Généralités

- Le langage s'inspire de la syntaxe du C tout comme Java.

- On retrouve donc des similarités au niveau de l'écriture :
  - Les instructions sont séparées par des points-virgules ;
  - Les bloc d'instruction sont encadrés par des accolades { … }
  - Les commentaires s'écrivent avec // ou /* … */

Les opérateurs

- Les opérateurs sont similaires à ceux du C (et donc du Java) :

Type d'opération	**Opérateur**	**Exemple**
Affectation	=	*unNombre* = 0;
Incrémentation et décrémentation	++ et --	*unNombre*++;
Opérations mathématiques	+, -, *, / et %	*somme* = 4 + *unNombre*;
Comparaison	==, !=, <, <=, > et >=	*somme* < 10
Algèbre booléenne	&&, || et !	(*unNombre* % 2 == 0) && *somme* < 10;
Opérateur ternaire	? :	*unNombre* % 2 == 0 ? "pair" : "impair"
Concaténation	+	*uneChaine* = "Hello, " + "World!";

Variables

Les différents types manipulables

Variables

- Javascript est un langage à typage dynamique faible.
  - Les variables ne sont donc pas déclarées avec un type particulier. Le type de la variable est déduit de sa valeur.
- Les mots clés `var` et `let` permettent d'introduire une variable.

```
var unNombre = 42;

let uneChaine = "Bonjour !";
```

Portée des variables

- La portée des variables dépend du mot clé utilisé à leur déclaration.
  - `let` : la variable a une portée au niveau du bloc ;
  - `var` : la variable a une portée au niveau de la fonction ;
  - Aucun mot clé : la variable a une portée globale.

Types simples

- On retrouve les mêmes types simples que dans la plupart des langages :

```
let unNombreEntier = 42;
let unNombreFlottant = 3.14;
let unBooleen = true;
let uneChaine = "Bonjour !";
let uneAutreChaine = 'Hello!';
```

Tableaux

- En Javascript, les tableaux sont des objets (`Array`). On peut en déclarer un de 2 façons :

```
let monTableau       = new Array();
let monAutreTableau = [];
```

- La méthode `push` ajoute un élément au tableau :

```
monTableau.push("Une Chaine");
monTableau.push(37);
monTableau.push(false);
```

Tableaux

- Pour accéder à un élément du tableau, on peut utiliser son index :

```
let maValeur = monTableau[2];
monTableau[2] = 22;
```

Autres méthodes et attributs d'un tableau

Méthode	Description
length	Nombre d'éléments du tableau
push(el1[,…,elN])	Ajoute N éléments à la fin
pop() / shift()	Supprime le dernier / le premier élément
sort()	Classe les éléments par ordre naturel
splice(idxDebut, nbDelete)	Supprime nbDelete éléments depuis l'index idxDeb
indexOf(el) / lastIndexOf(el)	Retourne l'index du premier / dernier élément trouvé. Renvoie -1 si non trouvé
toString()	Retourne une chaîne de caractères avec tous les éléments séparés par une « , »

Objets

- Les objets Javascript ressemblent à des tableaux associatifs.
- Les différentes valeurs des champs d'un objet peuvent être accédées avec une notation pointée ou avec des crochets.

```
let utilisateur = {};
utilisateur.nom = "Dupont";
utilisateur["prenom"] = "Jean";
```

Objets

- Il est possible de déclarer directement les valeurs des attributs d'un objet à sa création.

```
let utilisateur = {
    nom: "Dupont",
    prenom: "Jean",
    age: getAge('1995-05-09'),
    categorie: categorieUtilisateur
};
```

- Cette façon de représenter des objets est l'inspiration du format JSON (JavaScript Object Notation).

Opérateurs de comparaison stricte

- Les opérateurs de comparaison en Javascript ne prennent pas en compte le type de la variable par défaut.

```
let a = 1;
let b = "1";
console.log(a == b); // affiche true
```

- Il existe deux opérateurs de comparaison stricte qui prennent en compte le type de la variable : `===` et `!==`

```
console.log(a === b); // affiche false
```

Comparaisons étranges

- Il va falloir être attentif aux types de variables manipulées car des comportements étranges peuvent apparaitre.

```
console.log(0 == "0");        // true
console.log(0 == false);      // true
console.log("0" == false);    // true
console.log(false == []);     // true
console.log(0 == []);         // true
console.log("0" == []);       // false
```

Variable non définie

- Une variable sans valeur vaut `undefined`.

```
let variableNonDefinie;
console.log(variableNonDefinie); // Ecrit undefined
let isTrue = (variableNonDefinie == undefined);
```

- La valeur `null` existe aussi mais est différente de `undefined`.

```
variableNonDefinie = null;
console.log(variableNonDefinie); // Ecrit null
console.log(variableNonDefinie == undefined); // true
console.log(variableNonDefinie === undefined); // false
```

Structures de contrôle

Instructions de base du langage

Introduction

- Les structures de contrôle regroupent les instructions de conditions et les boucles.

- En Javascript, elles sont similaires à celles du C (et donc du Java)

Condition (if … else …)

```
if (unNombre < 10) {
    alert("Le nombre vaut moins de 10");
} else if (unNombre <= 20) {
    alert("Le nombre est entre 10 et 20");
} else {
    alert("Le nombre est supérieur à 20");
}
```

Condition (switch case)

```
switch (couleur) {
    case "bleu":
        // la couleur est bleue
        break;
    case "rouge":
        // la couleur est rouge
        break;
    default:
        // la couleur n'est ni bleue, ni rouge
        break;
}
```

Boucles while et do…while

```
while (unNombre < 100) {
    console.log(unNombre + " n'est pas assez grand!");
    unNombre += 10;
}

do {
    console.log(unNombre + " est trop grand!");
    unNombre -= 10;
} while (unNombre > 10);
```

Boucle for simple

```
for(let i = 0; i < 10 ; i++){
    console.log("Itération n°" + i);
}
```

Boucle for…of

- La boucle `for…of` sert à itérer sur les valeurs d'un tableau.

```
let tableau = [10, 20, 30, 40];
for (let valeur of tableau) {
    console.log("Valeur = " + valeur);
}
```

Boucle for…in

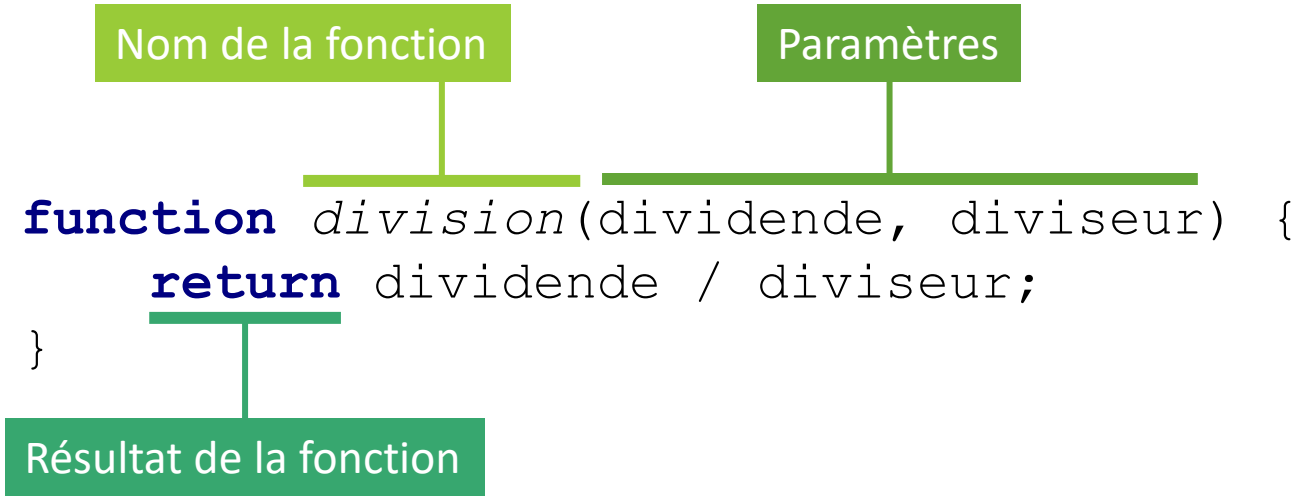
- La boucle `for…in` sert à itérer sur les propriétés d'un objet.

```
let objet = {a: 1, b: 2, c: 3};
for (let propriete in objet) {
    // propriete vaut "a", puis "b", puis "c"
    console.log(objet[propriete]);
}
```

Les fonctions

Introduction

- Les fonctions sont créées avec le mot clé `function`.
- Une fonction a un nom et peut déclarer de 0 à n paramètres



Appel de fonction

- Pour appeler une fonction, il suffit d'écrire son nom suivi des éventuels paramètres.

```
console.log(division(8, 2)); // 4
console.log(division(8, 3)); // 2.6666666666666665
```

- Il n'est pas obligatoire de préciser l'ensemble des paramètres. Les paramètres oubliés ont alors pour valeur `undefined`.

```
division(5);
division();
```

Fonctions comme variables

- En Javascript, une fonction peut être stockée dans une variable.

Nom de la variable utilisable pour appeler la fonction

```
let multiplication = function (a, b) {
    return a * b;
};

console.log(multiplication(3, 4)); // 12
```

Fonction comme paramètre d'une fonction

- Puisqu'une fonction peut être stockée dans une variable, elle peut donc être donnée en paramètre d'une autre fonction.

```
var fonctionAvecChoix = function (a, b, choix1, choix2) {
    if (a > b) {
        return choix1(a, b);
    } else {
        return choix2(a, b);
    }
};

console.log(fonctionAvecChoix(8, 2, division, multiplication)); // 4
console.log(fonctionAvecChoix(2, 8, division, multiplication)); // 16
```

Fonction comme une espèce de classe

- Il est possible d'utiliser les fonctions comme des espèces de classe comme on en utilise en Java.
- Le mot clé `this` permet de déclarer des propriétés ou méthodes.

```
let Utilisateur = function (nom, prenom) {
    this.nom = nom;
    this.prenom = prenom;

    this.getNomComplet = function () {
        return this.prenom + " " + this.nom;
    };
};
```

Instancier un objet

- Le mot clé new permet d'instancier un nouvel objet.

```
let jean = new Utilisateur("DUPONT", "Jean");

console.log(jean);                    // Objet Utilisateur
console.log(jean.nom);                // DUPONT
console.log(jean.getNomComplet());    // Jean DUPONT
```

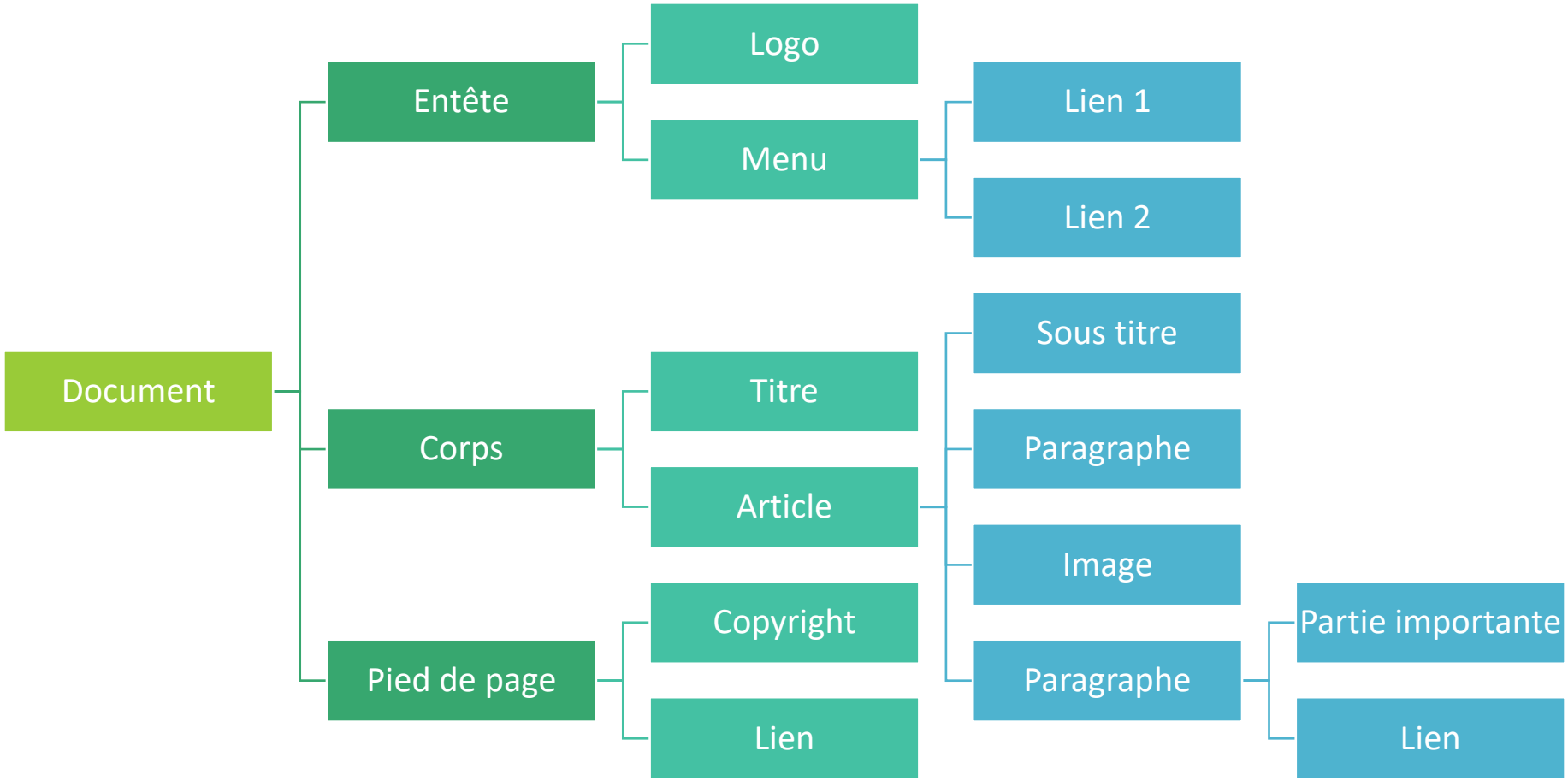
Document Object Model

Interagir avec la page HTML

DOM

- Le Document Object Model (DOM) est une représentation du document HTML ainsi qu'une API pour manipuler ce document.
- Le document HTML est structuré sous la forme d'un arbre avec pour chaque nœud de l'arbre un des éléments HTML.
- L'API DOM donne accès à des objets et fonction permettant de jouer avec les éléments HTML.

Arbre DOM


Document
Entête
Logo
Menu
Lien 1
Lien 2
Corps
Titre
Article
Sous titre
Paragraphe
Image
Paragraphe
Partie importante
Lien
Pied de page
Copyright
Lien

L’objet document

- L’objet `document` est la racine du document DOM. Il représente le document HTML complet.
- C’est une variable globale accessible dans le code Javascript.
- `document` possède plusieurs méthodes permettant de rechercher et récupérer des éléments HTML précis.

Recherche simple d'éléments

Méthode	Description
getElementById(string)	Récupère l'élément de la page avec l'id donné
getElementsByTagName(string)	Récupère un tableau des éléments avec la balise donnée
getElementsByClassName	Récupère un tableau des éléments avec la classe donnée

```
// élement de la page avec id="testId"
let element = document.getElementById("testId");
// tous les éléments p de la page
let paragraphes = document.getElementsByTagName("p");
// tous les éléments avec class="active"
let elementsActifs = document.getElementsByClassName("active");
```

Recherche avec des sélecteurs CSS

- Il est également possible de rechercher des éléments avec la syntaxe des sélecteurs CSS avec les méthodes `querySelector` et `querySelectorAll`.
  - `querySelector` renvoie le premier élément qui correspond au sélecteur
  - `querySelectorAll` renvoie tous les éléments qui correspondent au sélecteur

```
// Premier élément p avec class="important"
let paragrapheImportant = document.querySelector("p.important");
// Tous les éléments p avec class="important"
let paragraphesImportants = document.querySelectorAll("p.important");
```

L’objet Element

- Chaque élément HTML de la page est représenté par un objet `Element`. Toutes les méthodes de recherche précédentes retournent ce type d’objet.
- Les objets `Element` possèdent les mêmes méthodes de recherche que `document` pour rechercher dans ses éléments enfants.

```
// Récupère un paragraphe avec id="premier"
let premierP = document.querySelector("p#premier");
// Récupère tous les éléments strong contenu dans le paragraphe
let partiesImportantes = premierP.getElementsByTagName("strong");
```

Propriétés d'un élément

- Les propriétés d'un élément permettent de récupérer ses informations.

Nom	Description
id	Récupère l'identifiant de l'élément
attributes	Liste les attributs de l'élément
classList	Liste les classes de l'élément
className	Récupère la valeur de l'attribut class
innerHTML	Récupère le code HTML contenu dans l'élément
innerText ou textContent	Récupère le contenu textuel contenu dans l'élément
children	Liste les éléments enfants de l'élément
childElementCount	Récupère le nombre d'éléments enfants
…	

Méthodes d’un élément

- Un élément a également des méthodes facilitant sa manipulation.

Méthode	Description
getAttribute(nom)	Récupère la valeur d’un attribut particulier
setAttribute(nom, valeur)	Affecte une valeur à un attribut particulier
hasAttribute(nom)	Renvoie true ou false suivant si l’attribut existe
removeAttribute(nom)	Supprimer l’attribut
remove()	Supprime l’élément courant

Ajouter un nouvel élément

- L'objet document possède une méthode `createElement` permettant de créer un nouvel élément.
- Cet élément peut ensuite être ajouté en tant qu'enfant de n'importe quel élément avec la méthode `appendChild`.

```
// Création d'un élément li (non attaché à la page HTML pour l'instant)
let entreeListe = document.createElement("li");
// Ajout du contenu textuel du nouvel élément
entreeListe.innerText = "Nouvel élément";
// Récupère une liste (ul) de mon document HTML
let liste = document.getElementById("listePrincipale");
// Ajout de l'élément dans la page HTML comme nouvel enfant de la liste
liste.appendChild(entreeListe);
```

L’objet window

- L’objet `window` représente la fenêtre du navigateur.
- C’est une variable globale accessible dans le code Javascript.
- La propriété `onload` est intéressante quand on manipule le DOM. On peut lui donner une fonction qui sera exécutée une fois que l’ensemble du document DOM a été chargé.

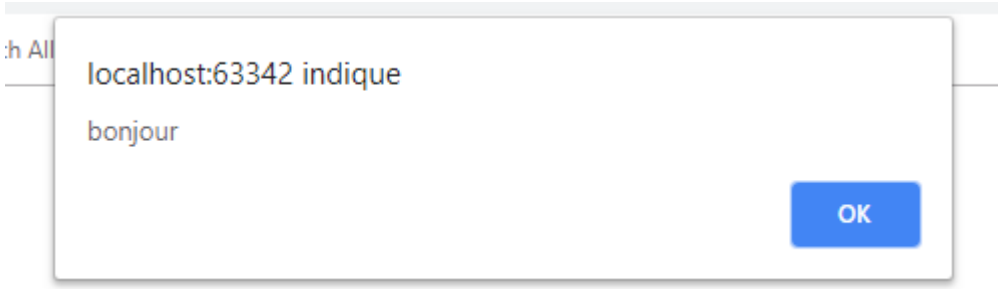
Exemple

```
window.onload = function () {
    let titreH1 = document.getElementById("test");
    console.log(titreH1.attributes);
    console.log(titreH1.getElementsByTagName("small"));
    document.getElementsByClassName("test");
};
```

Afficher des pop-up

- La méthode `alert` de l'objet `window` permet d'afficher une pop-up avec un message.

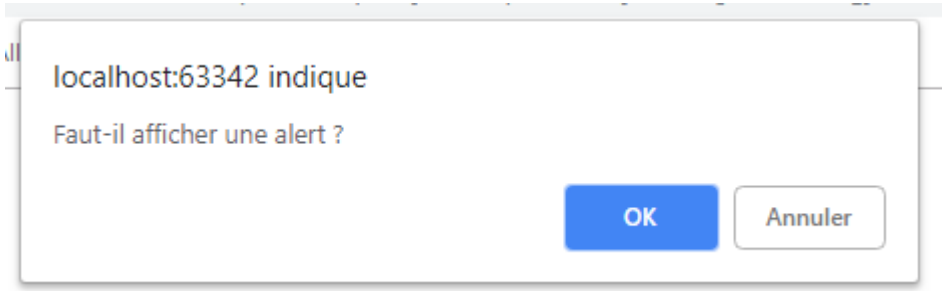
```
alert("bonjour");
window.alert("bonjour");
```



Fenêtre de confirmation

- La méthode `confirm` de l'objet `window` permet d'afficher une pop-up de confirmation. La méthode retourne `true` ou `false` suivant si l'utilisateur clique sur OK ou Annuler.

```
if(confirm("Faut-il afficher une alert ?")) {
    alert("Attention !");
}
```



Changer de page

- La propriété location de l'objet window permet de changer la page affichée.

```
window.location = "autrepage.html";
```

Événements

Répondre aux actions de l’utilisateur

Principe

- La gestion des événements va permettre d'associer des traitements Javascript à des actions faite par l'utilisateur sur le document HTML :
  - Clic sur un bouton
  - Soumission d'un formulaire
  - Changement de la valeur d'un champ texte
  - Appuis sur une touche du clavier
  - Survol de la souris sur un élément
  - …

Intercepter un événement

- Il y a deux façon d'attacher un traitement à un événement :
  - Dans le HTML avec un attribut onXXX :

```
<ul id="listePrincipale" onclick="alert('merci d\'avoir cliqué');">
```

  - Dans le Javascript avec une propriété onXXX :

```
let liste = document.getElementById("listePrincipale");
liste.onclick = function () {
    alert("merci d'avoir cliqué");
};
```

Événements principaux

Événement	Description
click	Lors d'un clic sur l'élément
change	Lorsque le contenu de l'élément change (valeur des éléments de formulaire par exemple)
focus	Lorsque l'élément gagne le focus
blur	Lorsque l'élément perd le focus
load	Lorsque l'élément est chargé (objet window)
submit	Lorsque que l'élément est soumis (pour un formulaire)
reset	Lorsque l'élément est réinitialisé (pour un formulaire)
…	

AJAX

Communiquer avec un serveur

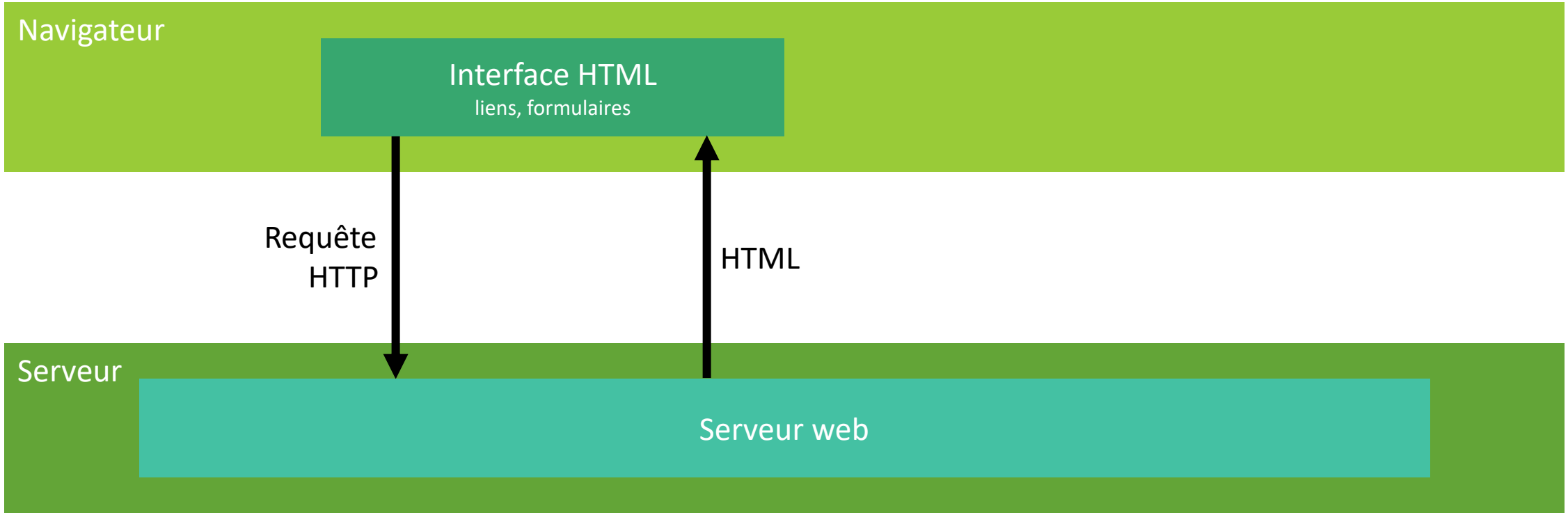
Généralités

- AJAX est l'acronyme de **A**synchronous **J**avascript **A**nd **X**ML.

- C'est un ensemble de technologies qui ont été assemblées pour pouvoir construire des sites web dynamiques interactifs :
  - Javascript pour sa manipulation du DOM
  - XMLHttpRequest pour communiquer avec le serveur
  - Un format de données d'échange : historiquement XML, remplacé souvent de nos jours par JSON

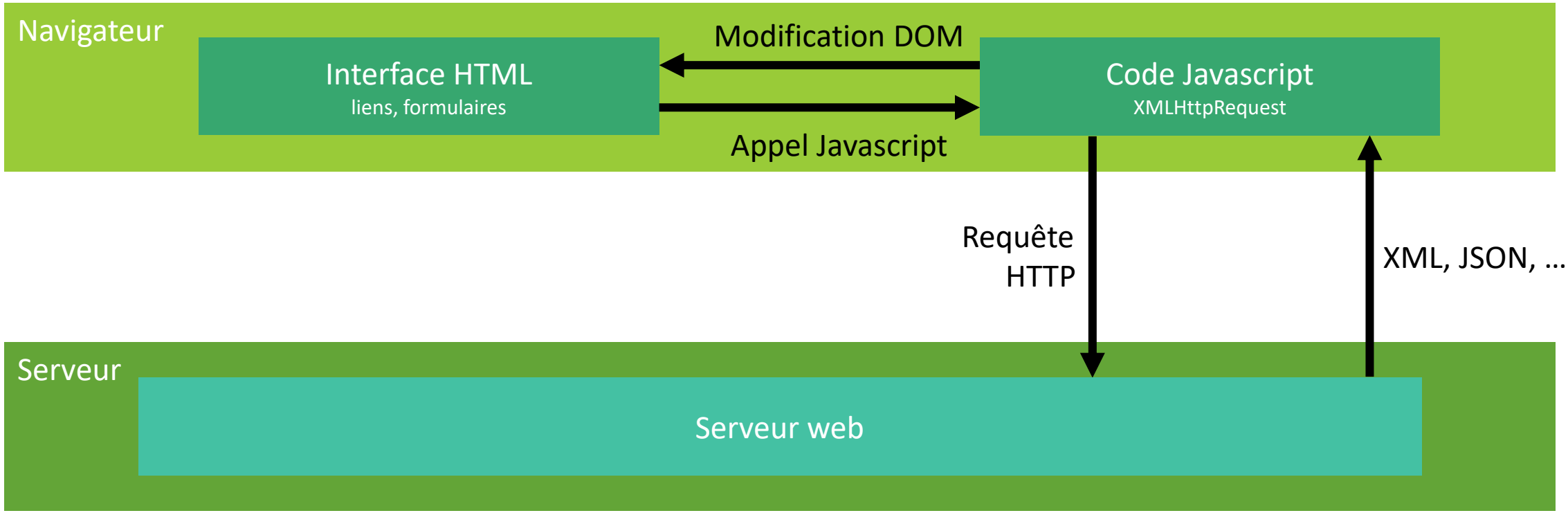
Application web classique

- Dans une page web classique, les communications avec le serveur via des requêtes HTTP permettent d'afficher une page web complète.



Application web AJAX

- Avec AJAX, le code Javascript va servir d'intermédiaire entre la page HTML et le serveur.



XMLHttpRequest

- Une requête AJAX est représentée par un objet `XMLHttpRequest`.

```
let requete = new XMLHttpRequest();
```

- La méthode `open` permet de décrire la requête que l'on veut exécuter.

```
requete.open("GET", "http://serveur.com/infos", true);
```

Types de réponses

- Il est possible de préciser le type de réponse attendu avec la propriété `responseType` de l'objet `XMLHttpRequest`.

```
let requete = new XMLHttpRequest();
requete.open("GET", "requete_qui_renvoie_json", true);
requete.responseType = "json";
```

- La réponse sera alors traitée en conséquence. Par exemple, pour la valeur `json`, la réponse sera parsée et transformée en objet Javascript.

Exécuter la requête

- Pour l'instant on n'a fait que décrire la requête. Pour l'exécuter, il faut appeler la méthode `send`.

```
requete.send();
```

- Dans le cas d'une requête POST, il est possible de préciser le body de la requête.

```
requete.send("nom=DUPONT&prenom=Jean");
```

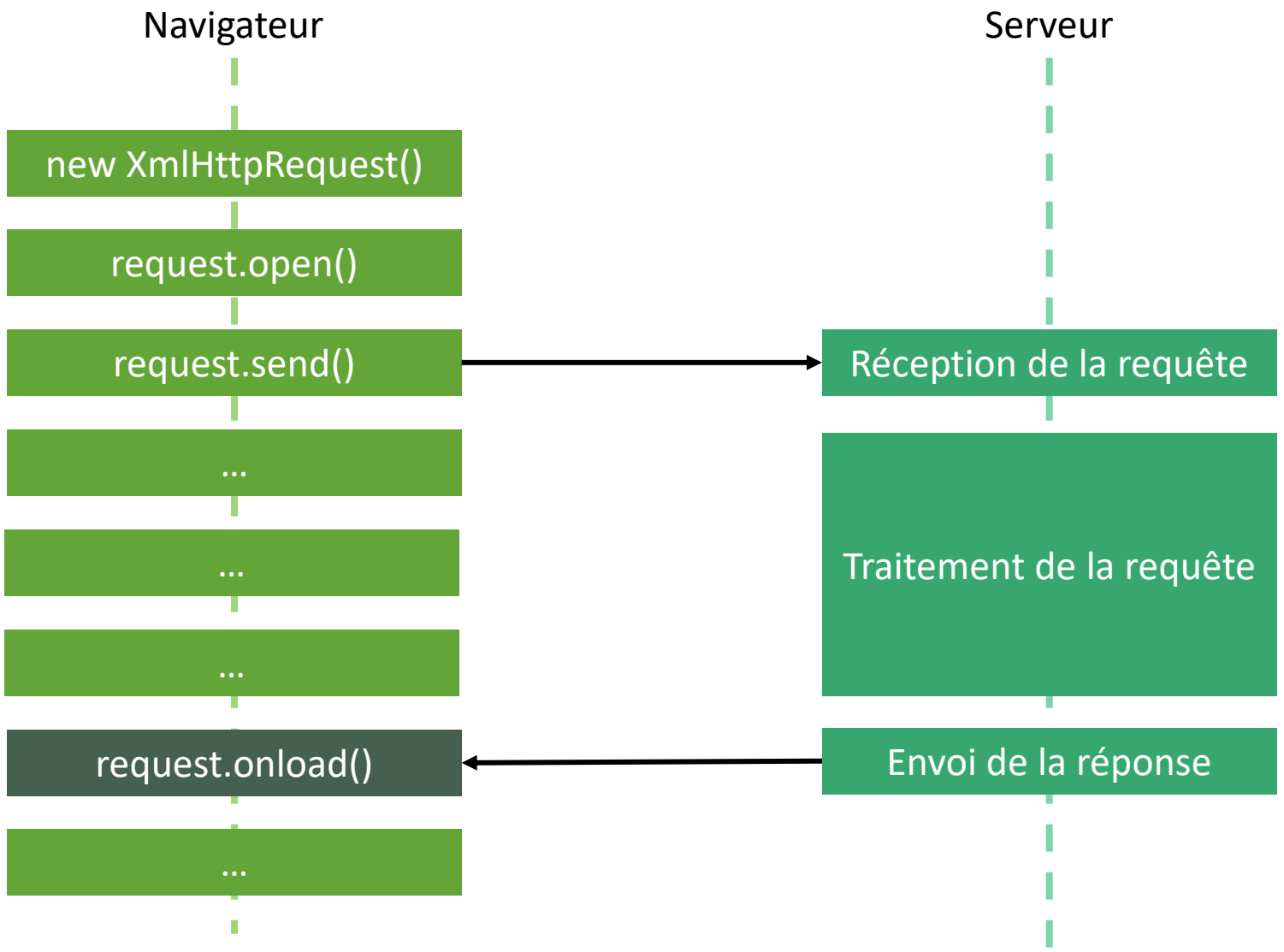
Traiter la réponse

- Pour traiter la réponse, il va falloir fournir une fonction à exécuter une fois la réponse reçue.
- Cela fonctionne avec des événements au niveau de l'objet `XMLHttpRequest`.

Événement	Description
onload	La requête est terminée (réponse reçue).
onreadystatechange	La requête a changé d'état.
onprogress	La réponse est en cours de réception. Surtout utile sur les réponses longues pour afficher une barre de chargement par exemple.
onerror	Le serveur n'a pas répondu.

Exemple de traitement avec onload

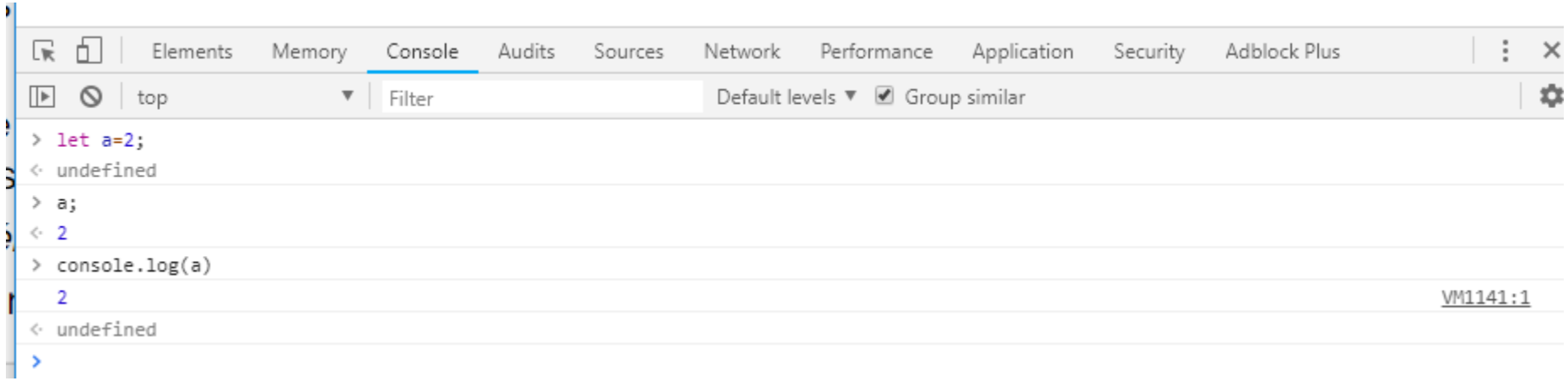
```
requete.onload = function () {
    if(this.status === 200) {
        let reponse = this.response;
        // faire quelque chose de la réponse
    } else {
        // traiter le cas d'erreur
    }
};
```


Navigateur
Serveur
new XmlHttpRequest()
request.open()
request.send()
...
...
...
request.onload()
...
Réception de la requête
Traitement de la requête
Envoi de la réponse

Quelques informations pratiques

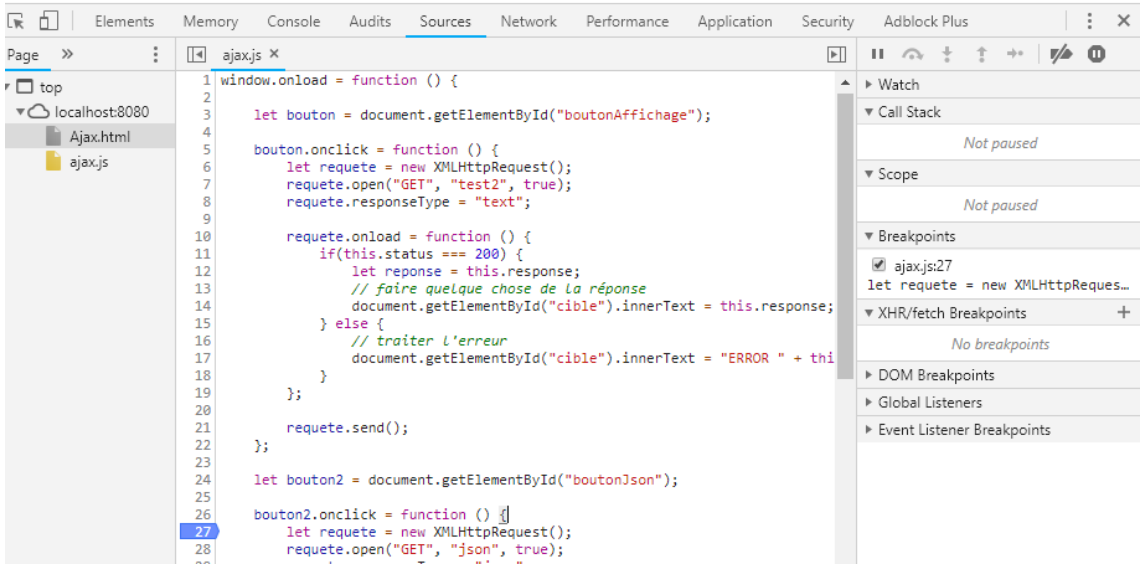
Log

- La méthode `console.log` permet d'afficher des informations dans la console.
- Cette console est accessible dans les outils de développement du navigateur.



Outils de debug

- Les navigateurs modernes proposent également un outil de debug comme celui utilisé en Java dans l'IDE.
- Dans chrome, il est accessible dans l'onglet Source des outils de développement.



Liens utiles

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference
- https://developer.mozilla.org/fr/docs/Web/API/Document_Object_Model