



Alphorm.com

JavaScript, *Les fondamentaux*

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Présentation du formateur
- Les autres formations sur Alphorm
- Qu'est-ce que JavaScript
- Le plan de formation
- Présentation des outils
- Les publics concernés



Formation JavaScript, les fondamentaux

alphorm.com™©



Présentation du formateur

Frédéric GAURAT

Développeur et formateur indépendant



Compétences

- Web Front :** HTML5/CSS3, JavaScript, Angular
- Web Back :** PHP, Symfony, CakePHP, JEE
- Mobile :** Android, Cordova/PhoneGap/Ionic

Mes références

- Site :** www.eolem.com
- Profil **Alphorm** : <http://www.alphorm.com/formateur/frederic-gaurat>

Formation JavaScript, les fondamentaux

alphorm.com™©



Les autres formations sur Alphorm

Formation jQuery

★★★★★ (5 votes), 13654 vues

jQuery, Ajax et jQuery UI

Fabien LE CORRE
Développeur Formateur et Consultant indépendant
alphorm.com

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Formation jQuery, Ajax et jQuery UI

20€
ACHETEZ LA FORMATION A VIE

OU S'ABONNER à partir de 25 €
Voir les vidéos gratuites

Aidez!

jQuery, Ajax et jQuery UI

Formation KnockoutJS

★★★★★ (5 votes), 6459 vues

KnockoutJS

Djamel BOUCHOUCHA
Formateur et Consultant .NET
Contact : conf@djamel.net

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>
Forum : <http://www.alphorm.com/forum>

Formation KnockoutJS

19€
ACHETEZ LA FORMATION A VIE

OU S'ABONNER à partir de 25 €
Voir les vidéos gratuites

Aidez!

KnockoutJS

Formation KnockoutJS

★★★★★ (5 votes), 6459 vues

KnockoutJS

Djamel BOUCHOUCHA
Formateur et Consultant .NET
Contact : conf@djamel.net

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>
Forum : <http://www.alphorm.com/forum>

Formation KnockoutJS

19€
ACHETEZ LA FORMATION A VIE

OU S'ABONNER à partir de 25 €
Voir les vidéos gratuites

Aidez!

KnockoutJS

Formation NodeJS, les fondamentaux

Apprendre les bases de NodeJS, votre chemin moderne vers des applications web avec une montée en charge importante

★★★★★ (5 votes), 921 vues

Formation
Les fondamentaux de
NodeJS

ÉDOUARD FERRARI
Formation NodeJS, les fondamentaux

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>
Formation NodeJS, les fondamentaux

39€
ACHETEZ LA FORMATION A VIE

OU S'ABONNER à partir de 25 €
Voir les vidéos gratuites

Aidez!

Formation NodeJS, les fondamentaux

Formation JavaScript, les fondamentaux

alphorm.com™©



JavaScript ?

- Langage créé en **1995**
 - Issu d'un langage de développement coté serveur (LiveScript)
 - Un partenariat Netscape/Sun donnera son nom définitif : **JavaScript**
 - Soumis à L'**Ecma** (organisme de standardisation) en **1997**
 - JavaScript devient un langage à part entière **1997**
- **Concrètement :**
- JavaScript va nous permettre de manipuler une page HTML directement dans le navigateur Web de l'utilisateur

JS



Cursus formations JavaScript

JavaScript,
les fondamentaux

DART

Tests
Unitaires

AngularJS 2,
les fondamentaux

JavaScript,
Avancé

CoffeeScript

Tests
fonctionnels

AngularJS 2,
Avancé

TypeScript



Le plan de formation

1. Introduction au développement Web
2. Les bases du langage
3. Les pièges du langage
4. Introduction à la programmation orienté objet
5. Javascript dans le navigateur
6. Javascript et la communication avec le serveur



Présentation des outils

Les éditeurs

- SublimeText
- Atom
- Microsoft Visual Studio Code
- NotePad++

Les navigateurs

- Chrome
- Firefox
- Internet Explorer



Les publics concernés

Les développeurs et chefs de projets qui souhaite connaître le fonctionnement de JavaScript.



JavaScript

C'est parti !



Alphorm.com

Introduction au développement Web

Etat de l'art des architectures Web

Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Le Web et son évolution et l'impact de JavaScript
- Le protocole HTTP et son utilisation
- L'architecture REST

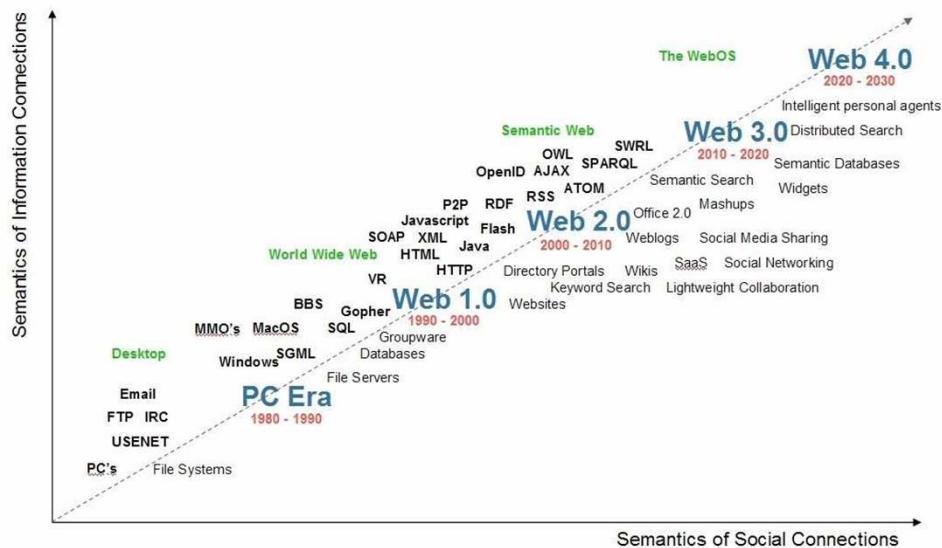


Formation JavaScript, les fondamentaux

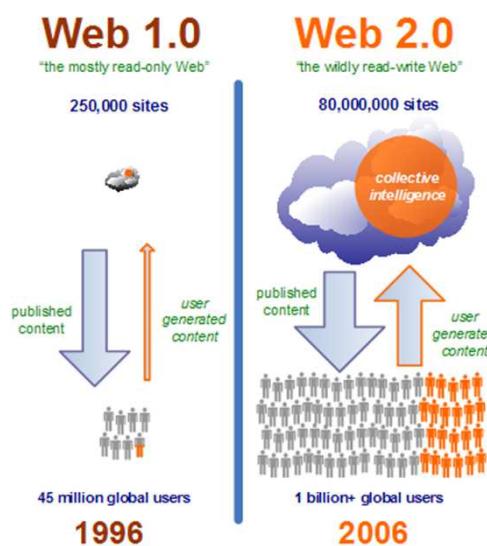
alphorm.com™©



Le Web 1.0 -> Le Web 2.0 -> ...

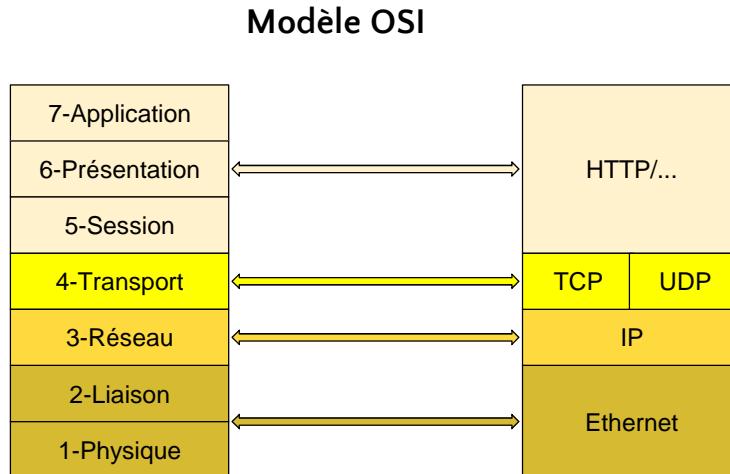


Le Web 1.0 -> Le Web 2.0

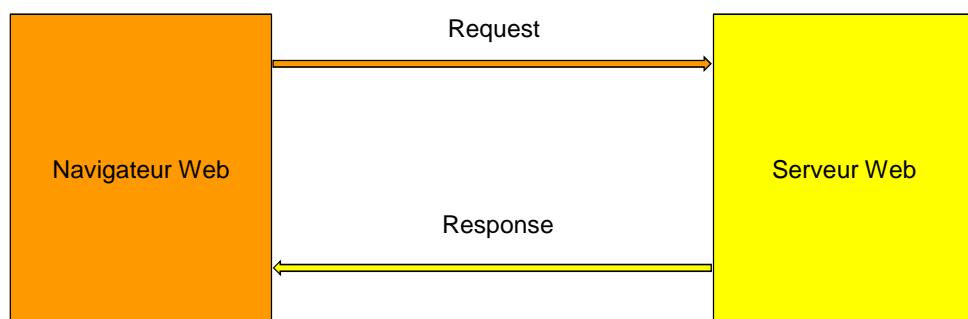




Le protocole HTTP

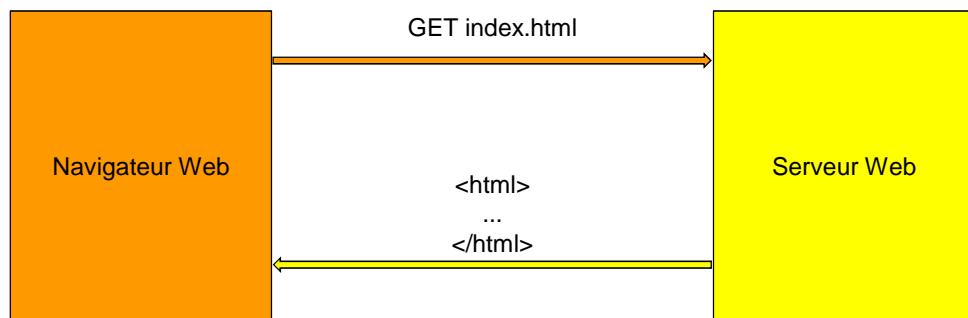


Le protocole HTTP : Un échange HTTP 1/2





Le protocole HTTP : Un échange HTTP 2/2



Le protocole HTTP : Les méthodes

GET	Demander une ressource
POST	Transmettre des données à une ressource
PUT	Remplacer ou ajouter une ressource
DELETE	Supprimer une ressource
HEAD	Demander des informations sur la ressource
OPTIONS	Connaître les options de communications d'une ressource
CONNECT	utiliser un proxy comme tunnel de communication
TRACE	permet d'effectuer des tests
PATCH	modification partielle d'une ressource



REST : Representational State Transfer

- L'architecture REST permet d'exploiter les méthodes HTTP associées à une URL...
- Lorsqu'un navigateur récupère (GET) une page (la ressource) et l'affiche on peut dire qu'il va récupérer la **représentation** courante de **l'état** de la ressource.
- **Exemple** : considérons cette URL : www.bibliotheque.fr/livres
 - GET sur www.bibliotheque.fr/livres/1 va **retourner** le livre dont l'id est 1.
 - DELETE sur www.bibliotheque.fr/livres/1 va **effacer** le livre dont l'id est 1.
 - POST sur www.bibliotheque.fr/livres/ permet de **créer** un nouveau livre
 - PUT sur www.bibliotheque.fr/livres/1 permet de **modifier** le livre dont l'id est 1.



Ce qu'on a couvert

- L'importance de JavaScript dans les nouvelles pratiques du développement web.
- Le fonctionnement du protocole HTTP
- La notion de web service avec l'architecture REST





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux

Alphorm.com

Introduction au développement Web

Installation de l'environnement
de développement



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Les outils nécessaires au développement JavaScript
- Node package manager (**npm**)
- L'éditeur Atom
- Les extensions d'Atom
 - Emmet
 - JSHint
- L'exécution
 - browser-sync



Formation JavaScript, les fondamentaux

alphorm.com™©



Les Outils pour coder en JavaScript

- Télécharger des outils et bibliothèques
- Coder
- Améliorer la productivité
- Tester son code



Node Package Manager

- **Node Package Manager** est un outil intégré dans **Node.js®** qui permet le téléchargement de bibliothèques et d'outils nécessaires au développement JavaScript.
- Pour information, **Node.js®** est une plate-forme de développement logiciel JavaScript.



Formation NodeJS sur Alphorm

- <http://www.alphorm.com/tutoriel/formation-en-ligne-nodejs-les-fondamentaux>

Formation NodeJS, les fondamentaux

Apprendre les bases de NodeJS, votre chemin moderne vers des applications web avec une montée en charge importante

★★★★★ (5 votes), 1027 vues

39€
ACHETEZ LA FORMATION A VIE !
ou
S'ABONNER à partir de 25 €
Voir les vidéos gratuits
Aide ?
✓ Formation vidéo de 9h35min44s
✓ Satisfait ou remboursé
✓ Accès à vie et visionnage illimité
✓ Attestation de fin de formation
✓ Conçue par un formateur expert
✓ Consultable sur iOS et Android
✓ Fichiers sources inclus
f 6 t 3 g+ 0

Formation JavaScript, les fondamentaux

alphorm.com™©



L'éditeur

- Il existe différents outils permettant de coder en JavaScript.
- Un simple **NotePad** suffirait mais ne nous rendrait pas suffisamment productif.
- Il existe plusieurs outils appropriés :
 - SublimeText**
 - Visual Studio Code**
 - NotePad++**
 - ...
- Nous utiliserons **Atom** (<https://atom.io/>) pour cette formation.
- Il est riche d'une grande communauté et il est possible de l'enrichir avec de nombreuses extensions.

Formation JavaScript, les fondamentaux

alphorm.com™©



Tester son code

- Nous avons besoins d'un serveur Web pour tester nos TPs
- C'est le rôle de **BrowserSync** (<https://www.browsersync.io/>)



Ce qu'on a couvert

- Les éditeurs pour coder
- Les modules supplémentaires pour faciliter le développement
- Les outils pour démarrer très rapidement un serveur web minimum





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Introduction au développement Web

Exemple HelloWorld



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Tester et valider notre environnement de développement



Formation JavaScript, les fondamentaux

alphorm.com™©



HelloWorld

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    alert("HelloWorld");
    console.log("HelloWorld");
  </script>
</head>
<body>

</body>
</html>
```



Ce qu'on a couvert

- Nous avons testé et validé notre environnement de développement





Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Les bases du langage

Les variables et leurs types



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Notion de variable
- Déclarer une variable
- Initialiser une variable
- Les contraintes de nommage
- Les opérateurs
- Les types primitifs et l'opérateur **typeof**



Formation JavaScript, les fondamentaux

alphorm.com™©



Notion de variable

- Utilisées pour stocker des valeurs
- Deux étapes pour les utiliser :
 - Déclarer la variable
 - Initialiser la variable



Déclarer une variable

- L'instruction **var**

```
var a; // déclaration
```



Initialiser une variable

- Il y a 2 possibilités :
 - Déclarer la variable et l'initialiser en 2 étapes :

```
var a; // déclaration  
a=1; // initialisation
```

- Déclarer la variable et l'initialiser en même temps :

```
var a = 1; // déclaration et initialisation
```



Les contraintes de nommages

- Les variables sont “**case sensitive**”
- C'est à dire :

```
var a = "minuscule";  
var A = "MAJUSCULE";
```

 - a et A sont deux variables différentes !**
- Il est recommandé d'utiliser le plus possible la touche de “complétion” de votre éditeur (touche TAB).
- Elle doivent commencer par une lettre



Les opérateurs

- Les opérateurs arithmétiques

+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo
++	Incrément
--	Décrément



Les types de variables et l'opérateur typeof

- En JavaScript les variables sont **typées**
- Il y a **5** types primitifs
 - **Number** : entier ou flottant
 - **String** : chaîne de caractères
 - **Boolean** : (true ou false)
 - **Undefined** : quand la variable que vous souhaitez utiliser n'existe pas
 - **Null** : type particulier n'ayant qu'une seule valeur : **null**
- Comment connaître le type d'une variable : l'opérateur **typeof**.



Ce qu'on a couvert

- Les variables et les pièges de la déclaration
- Les opérateurs et leurs fonctionnements
- Les types 5 primitifs l'utilisation de l'opérateur typeof



Alphorm.com

Les bases du langage

Les tableaux

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Frédéric GAURAT
Développeur et Formateur
Consultant indépendant
alphorm.com™©



Plan

- Notion de tableau
- Déclarer et initialiser un tableau
- Accéder aux éléments d'un tableau
- Les opérations sur les tableaux



Notion de tableau

- Un tableau est **une variable** qui contient une **liste de valeur**
- Représentation d'un tableau :

0	Une valeur
1	Une autre valeur
2	Encore une autre valeur



Déclarer et initialiser un tableau

- Déclaration d'un tableau vide

```
var a = [];
```

- Déclaration d'un tableau avec des valeurs

```
var a = [1,2,3];
```



Accéder aux éléments d'un tableau

- Les éléments d'un tableau sont accessibles par un index.
- Cet index est un nombre entier avec pour valeur initiale **0**.
- En reprenant cet exemple :

```
var a = [1,2,3];
```
- On en déduit la structure du tableau et la façon d'accéder aux éléments :

0	1
1	2
2	3

```
console.log(a[0]);
console.log(a[1]);
console.log(a[2]);
```



Modifier ou Ajouter une valeur

- Modification de la valeur située à l'index **2**.

```
a[2] = "la valeur 3";
```

- Ajout d'une valeur à l'index **3**.

```
| a[3] = "la valeur 4";
```

- Ajout d'une valeur à l'index **6** (!)

```
a[6] = "la valeur 7";
```



Effacer une valeur

```
var a = [1,2,3];
```

```
delete a[1];
```



Contenu d'un tableau

- Un tableau peut contenir différents types de données...

```
var a =[1,"deux",false,null,undefined];
```

- ... même d'autres tableaux

```
var a =[1,"deux",false,null,[1,2,3]];
```

- Pour accéder au tableau dans le tableau :

```
console.log(a[4][1]);
```



Ce qu'on a couvert

- Les tableaux sont des variables qui stockent des données
- Les éléments d'un tableau sont indexés
- Les index commencent à 0 et s'incrémentent lors de l'ajout d'éléments dans le tableau
- On accède à un élément en utilisant la notation avec crochet et son index: **a[0]**
- Un tableau peut contenir différents types de données, même d'autres tableaux





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Les bases du langage

Les conditions



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Notion de bloc de code
- Structure d'une condition if
- Structure d'une condition if - else
- Structure d'une condition if - else if - else
- La notation ternaire
- Le Switch



Formation JavaScript, les fondamentaux

alphorm.com™©



Notion de bloc de code

- Un bloc de code est un ensemble d'instructions encadré par des accolades

```
{  
    var a = 1;  
    var b = 2;  
}
```

- Tout le code du bloc est exécuté au moment où le bloc est sollicité



Notion de comparaison

- Une comparaison renvoie toujours une valeur **booléenne**
- Elle est constituée :
 - d'un opérateur de comparaison
 - d'un **opérateur logique**
 - de n'importe quelle **valeur** pouvant être convertie en booléen



Les opérateurs

• Logiques :

- !(NOT)
- &&(AND)
- || (OR)

• Comparaisons :

- == (EQUAL)
- === (STRICT EQUALITY)
- < (LESS THAN)
- > (GREATER THAN),...



Structure d'une condition if

```
var a = 2;
var result = '';
if(a>1){
    result = "a est supérieur à 1";
}
```



Structure d'une condition if - else

```
var a = 1;
var result = '';
if (a > 2) {
    result = "a est supérieur à 2";
} else {
    result = "a n'est pas supérieur à 2";
}
```



Structure d'une condition if - else if - else

```
var a = 1;
var result = '';

if(a==1){
    result = "a==1";
}
else if(a==2){
    result = "a==2";
}
else if(a==3){
    result = "a==3";
}
else {
    result = " a est différent de 1,2,3";
}
```



Exemple d'utilisation

- TP : Comment tester l'existence d'une variable



La notation ternaire

- Ce code peut être simplifié :

```
var a = 1;
var result = '';
if (a > 2) {
    result = "a est supérieur à 2";
} else {
    result = "a n'est pas supérieur à 2";
}
```

- Par :

```
//notation ternaire
var a = 1;
var result = a<2?"a est supérieur à 2":"a n'est pas supérieur à 2";
```



Le switch

```
var a = 1;
var result = '';
switch (a) {
  case 1:
    result = 'a = 1';
    break;
  case 2:
    result = 'a = 2';
    break;
  default:
    result = 'a n\'est différent de 1 et 2 ';
    break;
}
```



Ce qu'on a couvert

- Les blocs de code
- La structure d'une condition
- Les différents tests
- La notation ternaire pour simplifier le code
- La simplification des tests complexes avec le switch





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux

Alphorm.com

Les bases du langage

Les boucles



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- La boucle while
- La boucle do...while
- La boucle for
- La boucle for...in
- La boucle for...of



Formation JavaScript, les fondamentaux

alphorm.com™©



La boucle while

```
var a = 0;  
while (a < 10) {  
    a++;  
}
```



La boucle do...while

```
var a = 0;  
do {  
    a++;  
} while (a < 10);
```



La boucle for

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



La boucle for...in

```
var a = ['un','deux','trois','quatre'];  
for (var i in a) {  
    console.log(i+ ' '+a[i]);  
}
```



La boucle for...of

- Nouveauté ECMAScript6 !

```
var a = ['un', 'deux', 'trois', 'quatre'];
for (var b of a) {
    console.log(b);
}
```



Ce qu'on a couvert

- La structure des boucles en JavaScript
- Une introduction à l'ECMAScript 6





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Les bases du langage

Les fonctions



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Notion de fonction
- Appeler des fonctions et passer des paramètres
- Quelques fonctions du langage
- Portée des variables
- Considérations importantes sur les fonctions



Formation JavaScript, les fondamentaux

alphorm.com™©



Notion de fonction

- Les fonctions sont des blocs de code réutilisables

```
function somme(a,b){  
    var c = a+b;  
    return c;  
}
```

- Une fonction renvoie toujours une valeur (**return**) !



Appeler des fonctions et passer des paramètres 1/2

```
function somme(a,b){  
    var c = a+b;  
    return c;  
}  
  
var resultat = somme(2,3);  
console.log(resultat); //5
```



2/2

- Tous les paramètres sont disponibles dans une variable spéciale : **arguments**

```
function une_function(){
    console.log(arguments);
}
une_function(1,"un");
```



Quelques fonctions du langage

- **parseInt()** : renvoie un entier à partir d'une chaîne de caractères
- **parseFloat()** : renvoie un flottant à partir d'une chaîne de caractères
- **isNaN()** : teste si le paramètre n'est pas un nombre
- **isFinite()** : teste si le paramètre n'est pas Infinity



Portée des variables

- La portée d'une variable n'est pas définie par un bloc de code !
- La portée d'une variable **est définie** par une fonction
- Une variable doit être déclarée avec **var**

```
var global = 'variable globale';

function une_function(){
    var local = 'variable locale';
}
```

```
var global = 'variable globale';

function une_function(){
    local = 'variable n\'est pas locale';
}
```



Considérations importantes sur les fonctions 1/2

- Les fonctions sont des données, elles sont typées !

```
function une_function(){
    console.log('une_function');

    console.log(typeof une_function);
}
```

- On peut en déduire cette notation

```
var une_function = function(){
    console.log('une_function');

    console.log(typeof une_function);
};
```



Considérations importantes sur les fonctions 2/2

- Les fonctions ont **3 caractéristiques** :

1. Elles contiennent du **code**
2. Elles sont **exécutables**
3. Elles se comportent comme des **variables**



Ce qu'on a couvert

- Les fonctions et leurs implémentations
- Découverte et utilisation de quelques fonctions du langage
- Lien entre fonction et portée de variable
- Caractéristiques des fonctions





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux

Alphorm.com

Les bases du langage

Gérer les erreurs



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Les Exceptions
- La structure **try...catch**
- La structure **try...finally**
- La structure **try...catch...finally**
- Lancer une exception avec **throw**



Formation JavaScript, les fondamentaux

alphorm.com™©



Les Exceptions

- Les exceptions sont la représentation d'un comportement exceptionnel
- Elles interrompent le flot d'exécution normal de votre application
- Elles évitent la scrutation d'une erreur



La console

- L'API console fournit par les navigateurs offre plus de possibilités que le simple `console.log("...");`
 - `console.info()` : affiche un message d'information

```
> console.info('un message d\'information')
un message d'information
```
 - `console.error()` : affiche une erreur

```
> console.error('un message d\'erreur');
un message d'erreur
```
 - `console.warn()` : affiche un message de warning

```
> console.warn('un message de warning');
un message de warning
```
 - ...



La structure try...catch

- Provoquons une erreur, ici la variable **ma_variable** n'existe pas.

```
console.log(ma_variable);
```

- Interceptons cette erreur

```
try{  
    console.log(ma_variable);  
}  
catch(e){  
    console.error("Il y a une erreur");  
}
```

- Le bloc **try** exécute du code susceptible de provoquer une erreur
- Le bloc **catch** intercepte cette erreur si elle a lieu



La structure try...finally

- Le bloc **finally** est exécuté quelque soit le résultat de l'exécution du bloc **try** (erreur ou pas)

```
try{  
    console.log(ma_variable);  
}  
finally{  
    console.info("après le bloc try");  
}
```

- Utile pour initialiser la valeur d'une variable pour la suite de l'exécution



La structure try...catch...finally

- C'est la forme la plus complète pour gérer des exceptions

```
try{
    console.log(ma_variable);
}
catch(e){
    console.error("Il y a une erreur");
}
finally{
    console.info("après le bloc try");
}
```



Lancer une exception avec throw

- Il est possible de lancer ses propres exceptions

```
function division(a,b){
    if(b==0){
        throw "Erreur : Division par zéro";
    }
    return a/b;
}

try{
    var c = division(2,0);
    console.log(c);
}
catch(erreur){
    console.log(erreur);
}
```



Ce qu'on a couvert

- Le mécanisme de gestion des erreurs en JavaScript
- Un aperçu de l'API console proposée par les navigateurs



Formation JavaScript, les fondamentaux

alphorm.com™©



Alphorm.com

Les pièges du langage

Notion de « Hoisting »

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Frédéric GAURAT
Développeur et Formateur
Consultant indépendant
alphorm.com™©



Plan

- Qu'est ce que le hoisting (traduction : une remontée)
- Le hoisting et la déclaration de variable
- Le hoisting et l'initialisation de variable



Qu'est ce que le hoisting

- En JavaScript, les déclarations de variables sont traitées avant tout !

```
a=2;  
var a;  
  
//est strictement équivalent à  
  
var a;  
a=2;
```



Le hoisting et la déclaration de variable

- **Rappel** : 2 étapes pour utiliser une variable :
 1. Déclaration
 2. Initialisation
- Nous le savons maintenant : les déclarations sont remontées (hoisted)



Le hoisting et l'initialisation de variable

- En revanche les initialisations, elles, ne sont pas remontées :

```
var a = 2; // Déclaration et initialisation de a
console.log(a+b);
var b = 3; // Déclaration et initialisation de b
```

- NaN ! b est bien déclarée (par hoisting) mais pas initialisé -> Undefined



Ce qu'on a couvert

- Les subtilités des déclarations et d'initialisation de variable en JavaScript



Alphorm.com

Introduction à la POO

Notions d'objet (les classes, l'héritage)

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Frédéric GAURAT
Développeur et Formateur
Consultant indépendant
alphorm.com™©



Plan

- Les objets ?
- Créer et utiliser des objets
- Les constructeurs
- L'opérateur instanceof



Les objets ?

- D'après wikipedia :
 - « ... un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre. »
- Bref...
 - Les **types primitifs** sont des éléments purement informatiques : String, Number
 - Les **objets** des représentations du monde réel : une facture, un client, un utilisateur.



Les objets en JavaScript

• Proche de la notion de tableau :

```
var a = ['un', 'deux', 'trois'];
```

Key	Value
0	un
1	deux
2	trois



Les objets en JavaScript

```
var personne = {  
    nom: "DURAND",  
    prenom: "Robert"  
};
```

Key	Value
nom	DURAND
prenom	Robert



Tableau indexé et tableau associatif

- Deux sortes de tableaux :
 1. **Indexé** -> les clefs sont des entiers
 2. **Associatif** -> les clefs sont des chaînes de caractères
- JavaScript utilise les objets pour implémenter les tableaux associatifs



Structure d'un objet

- Les propriétés et les méthodes

```
var personne = {
    nom:"DURAND",
    prenom:"Robert",
    direBonjour : function(){
        console.log('Bonjour');
    }
};
```



Utilisation d'un objet

```
var personne = {  
    nom:"DURAND",  
    prenom:"Robert",  
    direBonjour : function(){  
        console.log('Bonjour');  
    }  
};  
  
personne.direBonjour();
```



Création différée d'un objet

- JavaScript est un langage dynamique
- Il permet la modification d'un objet après sa création !

```
var personne={};  
personne.nom = "DURAND";  
personne.prenom = "Robert";  
personne.direBonjour = function(){  
    console.log('Bonjour');  
};
```



La valeur this

- **this** : « cet objet » ou l'objet courant

```
var personne = {
    nom:"DURAND",
    prenom:"Robert",
    afficheNom : function(){
        console.log('nom : '+this.nom);
    }
};
```

- Ici on affiche la propriété **nom** de l'**objet courant**



Notion de constructeur 1/2

- Utiliser une **function** pour créer un objet

```
function Personne(){
    this.nom = "DURAND";
    this.prenom = "Robert";
}

var personne = new Personne();
console.log(personne.nom);
```



Notion de constructeur 2/2

- Utiliser une **function** et passer des paramètres pour créer un objet

```
function Personne(p_nom,p_prenom){  
    this.nom = p_nom;  
    this.prenom = p_prenom;  
}  
  
var nom_personne = "DURAND";  
var prenom_personne = "Robert";  
var personne = new Personne(nom_personne,prenom_personne);  
console.log(personne.nom);
```



L'opérateur instanceof

- Type primitif -> typeof
- Objet -> instanceof
- L'opérateur **instanceof** teste le constructeur de l'objet

```
function Personne(p_nom,p_prenom){  
    this.nom = p_nom;  
    this.prenom = p_prenom;  
}  
  
var nom_personne = "DURAND";  
var prenom_personne = "Robert";  
var personne = new Personne(nom_personne,prenom_personne);  
console.log(personne.nom);  
  
console.log(personne instanceof Personne ? "oui":"non");
```



L'objet Object

- **Object** est l'objet de base implémenté par l'ensemble des objets

JavaScript.

```
var obj = {};  
var obj = new Object();
```

- Autrement dit tous les objets JavaScript **héritent** de Object
- Ils bénéficient des méthodes de Object :
 - **toString()**
 - **valueOf()**



Ce qu'on a couvert

- Les objets en JavaScript
- La construction d'objet
- L'utilisation d'objet
- Tester l'instance d'un objet





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Introduction à la POO

Notions de prototype



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Qu'est ce qu'un prototype?
- Ajouter des propriétés à un prototype
- Distinguer les propriétés d'un objet et celles de son prototype
- Améliorer les objets JavaScript (Array,...)



Formation JavaScript, les fondamentaux

alphorm.com™©



Qu'est ce qu'un prototype

· Rappel de **Wikipedia** :

- Javascript est un langage **orienté objet à prototype**, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une **propriété de prototypage** qui permet d'en créer des objets héritiers personnalisés.



Qu'est ce qu'un prototype

- On le sait, les fonctions sont des objets
- Elles ont des propriétés
- Une de ces propriétés est : **prototype**
- Il est possible d'ajouter des propriétés et des méthodes à partir du prototype



Ajouter des propriétés à un prototype

```
function Personne(p_nom,p_prenom){  
    this.nom = p_nom;  
    this.prenom = p_prenom;  
}  
  
Personne.prototype.age = 30;  
Personne.prototype.getAge = function(){  
    return this.age;  
}  
  
var personne = new Personne("DURAND","Robert");  
console.log(personne.getAge());
```



Ajouter des propriétés à un prototype

- En Javascript les objets sont passés par **référence**.
- C'est-à-dire qu'il y a **un** prototype pour tous les objets du même constructeur.

```
var personne = new Personne("DURAND","Robert");  
console.log(personne.getAge());  
var personne2 = new Personne("DURAND","Robert");  
console.log(personne2.getAge());  
Personne.prototype.age = 130;  
console.log(personne.getAge());  
console.log(personne2.getAge());
```



prototype

Dans notre exemple

```
function Personne(p_nom,p_prenom){  
    this.nom = p_nom;  
    this.prenom = p_prenom;  
}  
Personne.prototype.age = 30;  
var personne = new Personne("DURAND","Robert");  
console.log(personne.age);  
console.log(personne.constructor.prototype.age);
```



Améliorer les objets JavaScript (Array,...)

```
Array.prototype.showAll = function(){  
    for(var i=0;i<this.length;i++){  
        console.log(this[i]);  
    }  
};  
  
var arr = ['un','deux','trois'];  
arr.showAll();
```



Ce qu'on a couvert

- La notion de prototype et le fonctionnement précis des objets
- La modification de prototype
- La modification d'objet du langage



Alphorm.com

Introduction à la POO

Notions de
design patterns

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Frédéric GAURAT
Développeur et Formateur
Consultant indépendant
alphorm.com™©



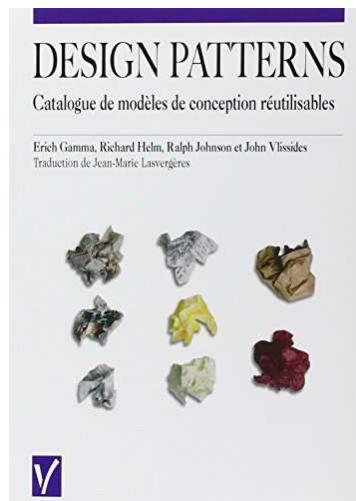
Plan

- Les Design Patterns : un problème, une solution
- Le pattern **Module**
- Le pattern **Singleton**



solution

- Le Gof book (Gang of four)





Le pattern Module

- Le pattern Module a été conçu à l'origine pour permettre **l'encapsulation** (private et public)
- En JavaScript ce pattern est utilisé pour simuler le comportement d'une classe.



Le pattern Module – fonction auto-appelante

```
(function sayHello(){  
    console.log("Hello");  
})();
```



Le pattern Module – Implementation

```
var monModule = (function () {
    var compteur = 0;
    return {
        incrementCompteur: function () {
            return compteur++;
        }
    };
})();

var i = monModule.incrementCompteur();
console.log(i);
i = monModule.incrementCompteur();
console.log(i);
```



Le pattern Singleton

- Le pattern Singleton permet de récupérer seulement **un** objet à partir d'un constructeur.



Le pattern Singleton

```
var monSingleton = (function () {
    var instance;
    function init() {
        // Singleton
        var hello = "Hello "+Math.random();
        return {
            sayHello: function() {
                console.log(hello);
            }
        };
    }

    return {
        getInstance: function () {

            if ( !instance ) {
                instance = init();
            }
            return instance;
        }
    };
})();
```

```
var s = monSingleton.getInstance();
var s2 = monSingleton.getInstance();
s.sayHello();
s2.sayHello();
```



Ce qu'on a couvert

- Une brève introduction au design patterns
- Le pattern Module
- Le pattern Singleton





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux

Alphorm.com

Javascript dans le navigateur

Les objets disponibles



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Les objets existants dans le navigateur (*The Browser Object Model*)
- Utilisation des principaux objets

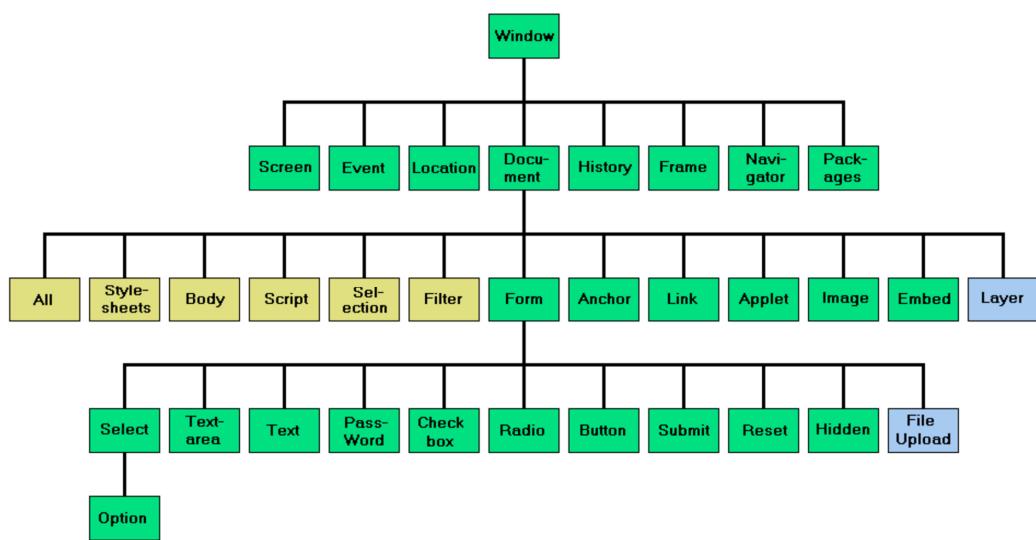


Formation JavaScript, les fondamentaux

alphorm.com™©



Model)



L'objet Window

- L'objet **Window** représente la fenêtre du navigateur

```
console.log(window.innerWidth);
console.log(window.innerHeight);
```



L'objet Form et ses entants

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        window.document.forms['le_form'].le_champ_text.value="Nouvelle valeur";
    </script>
</head>
<body>
    <form name="le_form" action="">
        <input type="text" name="le_champ_text">
    </form>
</body>
</html>
```



L'objet Document

• Document Object Model

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1 id="le_titre"></h1>
    <script>
        var le_titre = window.document.getElementById('le_titre');
        le_titre.innerHTML = "Un titre";
    </script>

</body>
</html>
```



Ce qu'on a couvert

- Les 2 modèles d'objets présents dans un navigateur le **BOM** et le **DOM**



Formation JavaScript, les fondamentaux

alphorm.com™©



Alphorm.com

Javascript dans le navigateur

Le cycle de vie
d'une page web

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Frédéric GAURAT
Développeur et Formateur
Consultant indépendant
alphorm.com™©



Plan

- Comprendre le fonctionnement de JavaScript dans une page Web.
- Exécuter du code au chargement de la page



Exécution du code JavaScript

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        alert('Bonjour');
    </script>
</head>
<body>
    <h1>Hello</h1>
</body>
</html>
```



Exécution du code JavaScript

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>

</head>
<body>
    <h1>Hello</h1>
    <script>
        alert('Bonjour');
    </script>
</body>
</html>
```



Analyses des 2 exemples

- Dans le premier exemple : l'alert bloque l'affichage du titre
- Dans le second exemple : le titre est affiché et l'alert s'affiche ensuite
- **Conclusion** le code d'une page Web est exécuté de façon séquentielle
- Or, nous souhaitons exécuter notre code quand la page est prête, c'est-à-dire quand elle est chargée.



Exécution au chargement de la page

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    function doload(){
      alert('hello');
    }
  </script>
</head>
<body onload='doload()'>
  <h1>Hello</h1>
</body>
</html>
```



Ce qu'on a couvert

- Le fonctionnement du navigateur
- Exécuté du code au moment du chargement





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Javascript dans le navigateur

Notion et utilisation des évènements



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Notion d'évènement
- Les principaux évènements en JavaScript



Formation JavaScript, les fondamentaux

alphorm.com™©



Notion d'évènement

- Les évènements vont permettre de réagir à des actions faites par l'utilisateur ou des évolutions de la page.



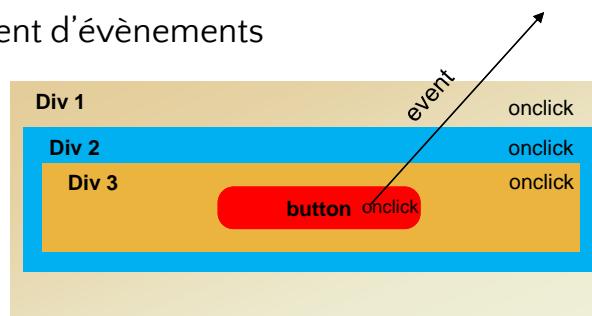
Le principaux événements

Event	Description
onchange	Un élément HTML a changé
onclick	Un utilisateur clique sur un élément HTML
onmouseover	Un utilisateur survol un élément HTML avec la souris
onmouseout	Un utilisateur quitte un élément HTML avec la souris
onkeydown	Un utilisateur appui sur une touche du clavier
onload	Le navigateur a fini le chargement de la page



Principes de fonctionnement

- Un évènement est un objet
 - Il a des propriétés et des méthodes
- Le bouillonnement d'évènements



onchange

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        function doChange(e){
            console.dir(e);
            console.log('value change');

        }
    </script>
</head>
<body>
    <input type="text" onchange="doChange(event)">
</body>
</html>
```



onclick

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        function doClick(e){
            console.dir(e);
            console.log('button click');
        }
    </script>
</head>
<body>
    <input type="button" onclick="doClick(event)" value="click">
</body>
</html>
```



JavaScript non-intrusif

- Il ne faut pas faire référence à du JavaScript dans une page web
- WebDesigner et Développeur ne sont pas forcément les mêmes personnes !



onchange non-intrusif

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        var doLoad = function(){
            var the_input = document.getElementById('the_input');
            the_input.onchange = function(e){
                console.dir(e);
                console.log('button click');
            };
        }
        window.onload = doLoad;
    </script>
</head>
<body>
    <input type="text" id='the_input'>
</body>
</html>
```



onclick non-intrusif

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        var doLoad = function(){
            var the_button = document.getElementById('the_button');
            the_button.onclick = function(e){
                console.dir(e);
                console.log('button click');
            };
        }
        window.onload = doLoad;
    </script>
</head>
<body>
    <input id="the_button" type="button" value="click">
</body>
</html>
```



Ce qu'on a couvert

- Les évènements JavaScript
- Le bouillonnement d'évènements
- Coder les principaux évènements
- Implémenter les bonnes pratiques



Formation JavaScript, les fondamentaux

alphorm.com™©



Alphorm.com

Javascript dans le navigateur

Manipuler
les formulaires

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Frédéric GAURAT
Développeur et Formateur
Consultant indépendant
alphorm.com™©



Plan

- Les formulaires
- Les éléments de formulaire
- La validation de formulaire



Les formulaires

- Les formulaires sont des objets :
 - Propriétés : action, method,....
 - Méthodes : submit(), reset()



Les éléments de formulaires

Il y en a peu :

- Input (text,file,date,...)
- Checkbox
- Radio
- Select
- Textarea
- Button



Les éléments de formulaires

Ils sont accessibles de différentes façons :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    window.onload = function(){
      var elm = document.forms[0].elements[2];
      var elm = document.forms.monForm.elements.monElement_3;
      var elm = document.forms.monForm.elements['monElement_3'];
    }
  </script>
</head>
<body>
  <form action="" name="monForm">
    <input type="text" name="monElement_1">
    <input type="text" name="monElement_2">
    <input type="text" name="monElement_3">
  </form>
</body>
</html>
```



Validation de formulaire

- Valider un formulaire c'est empêcher son envoi s'il n'est pas correct
- On implémente la méthode **onsubmit** qui doit renvoyer un booléen

```
<script>
  window.onload = function(){
    document.forms.monForm.onsubmit = function(e){
      var result = false;
      var elem1 = document.forms.monForm.monElement_1;

      if(elem1=='value1'){
        result = true;
      }
      else result = false;

      return result;
    }
  }
</script>
```

```
<form action="" name="monForm">
  <input type="text" name="monElement_1">
  <input type="submit" value="envoyer">
</form>
```



Ce qu'on a couvert

- Les formulaires et leurs fonctionnement
- Les éléments de formulaire
- La validation





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Javascript dans le navigateur

Manipuler une page web avec le DOM



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Approfondissement de la notion de DOM
- Récupération des éléments par différentes méthodes



Formation JavaScript, les fondamentaux

alphorm.com™©



Le DOM

- Le DOM est une **API** de manipulation de document (balisé)
- Normalisé par le W3C
- Implémenter dans tous les langages



Récupération d'éléments

- Considérons cet extrait de page

```
<p id="p1">
  p1
  Lorem ipsum dolor sit amet, consectetur adipisicing elit.
</p>

<p id="p2" class="la_class">
  p2
  Lorem ipsum dolor sit amet, consectetur adipisicing elit.
</p>

<p id="p3" class="la_class">
  p3
  Lorem ipsum dolor sit amet, consectetur adipisicing elit.
</p>
```



Récupération d'éléments par id

```
window.onload = function(){
    var p1 = document.getElementById('p1');
}
```



Récupération d'éléments par tag

```
window.onload = function(){
    var p = document.getElementsByTagName('p');
}
```



class

```
window.onload = function(){
    var p = document.getElementsByClassName('la_class');
}
```



Récupération d'éléments par notation CSS

```
window.onload = function(){
    var p = document.querySelector('#p1');
}
```



Récupération d'éléments par notation CSS

```
window.onload = function(){
    var p = document.querySelectorAll('.la_class');
}
```



Ce qu'on a couvert

- Le Document Object Model
- Les différents moyens de récupérer des éléments dans une page





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux

Alphorm.com

Javascript dans le navigateur

Manipuler les styles CSS



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Manipuler les styles CSS en JavaScript
- Afficher/Cacher des éléments



Formation JavaScript, les fondamentaux

alphorm.com™©



Rappels sur les notations

- **Camel case** : backgroundColor
- **Train case** : background-color



Modifier un style CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        window.onload = function(){
            var p = document.getElementById('p1');
            p.style.backgroundColor = 'yellow';
        }
    </script>
</head>
<body>
    <p id="p1">
        p1
        Lorem ipsum dolor sit amet, consectetur adipisicing elit.
    </p>
</body>
</html>
```



Afficher/Cacher des éléments

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        window.onload = function(){
            var btn = document.getElementById('btn');
            btn.onclick = function(){
                var p1 = document.getElementById('p1');
                var visible = p1.style.visibility;
                if(visible == 'visible') p1.style.visibility = 'hidden';
                else p1.style.visibility = 'visible';
            }
        }
    </script>
</head>
<body>
    <input type="button" id="btn" value="Afficher/Cacher"><br/>
    <p id="p1">
        p1
        Lorem ipsum dolor sit amet, consectetur adipisicing elit.
    </p>
</body>
</html>
```



Ce qu'on a couvert

- Les notations CSS -> JavaScript
- Modifier un style CSS
- Afficher et Masquer un élément en passant par un évènement





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Javascript et la communication
avec le serveur

Présentation d'Ajax et JSON



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Découvrir la technologie Ajax
- Découvrir JSON



Formation JavaScript, les fondamentaux

alphorm.com™©



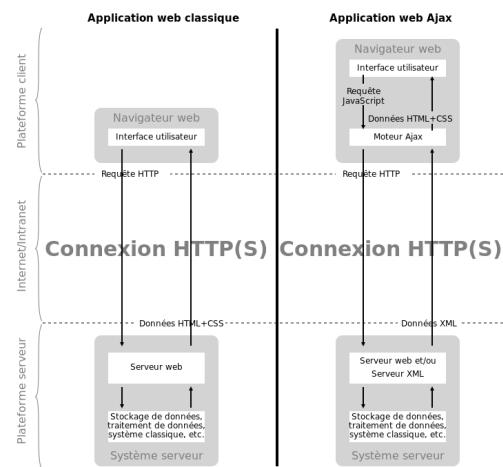
Découvrir la technologie Ajax

- Était l'acronyme de Asynchronous JavaScript and XML
- Maintenant on parle de technologie Ajax



Découvrir la technologie Ajax

- Le navigateur ne charge plus complètement une page
- Il récupère les données en JavaScript
- Il modifie la page en conséquence





Utiliser Ajax 1/3

- Faire une requête Ajax

```
var xhr = new XMLHttpRequest();
xhr.open("GET", "data.json", true);
xhr.send(null);
```

- La requête est de type GET
- Vers la ressource « data.json »
- Elle est asynchrone



Utiliser Ajax 2/3

- Attendre la réponse du serveur

```
var xhr = new XMLHttpRequest();
xhr.open('GET','lorem.txt',true);
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4 && xhr.status == 200){
        console.log(xhr.responseText);
    }
}
xhr.send();
```



Utiliser Ajax 3/3

- Les états de l'objet XHR

Valeur	Description
0	Requête non initialisée
1	Connexion établie
2	Requête reçue
3	Requête en cours de traitement
4	Requête finie et la réponse est prête



Découvrir JSON

- Le format des données renvoyées par le serveur n'est plus XML
- C'est le **JSON : JavaScript Object Notation**
- Un format naturellement compris par JavaScript

```
var reponseDuServeur = "{ 'nom': 'DURAND', 'prenom': 'Robert'}";
```



Utiliser JSON

- Il suffit d'évaluer une chaîne contenant du JSON pour obtenir l'objet JS

```
var reponseDuServeur = "{ 'nom': 'DURAND', 'prenom': 'Robert'}";  
var obj = eval('('+reponseDuServeur+')');
```



Ce qu'on a couvert

- L'architecture Ajax
- L'utilisation d'Ajax en Javascript
- Le format JSON





Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Javascript et la communication
avec le serveur

Soumettre un formulaire en Ajax



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Utiliser Ajax pour envoyer un formulaire au serveur



Formation JavaScript, les fondamentaux

alphorm.com™©



Envoyer un formulaire en Ajax 1/2

- Récupérer les données et les transformer en JSON

```
<script>
    window.onload=function(){
        document.forms.le_form.onsubmit = function(){
            var le_form = document.forms.le_form;
            var data={};

            for(var i = 0; i<document.forms.le_form.length;i++ ){
                if(le_form[i].name)
                    data[le_form[i].name] = le_form[i].value;
            }
            return false;
        }
    }
</script>
```



Envoyer un formulaire en Ajax 2/2

- Les envoyer au serveur

```
<script>
    window.onload=function(){
        document.forms.le_form.onsubmit = function(){
            var le_form = document.forms.le_form;
            var data={};

            for(var i = 0; i<document.forms.le_form.length;i++ ){
                if(le_form[i].name)
                    data[le_form[i].name] = le_form[i].value;
            }

            var str_data = JSON.stringify(data);
            var xhr = new XMLHttpRequest();
            xhr.open("POST", "gens", true);
            xhr.setRequestHeader('Content-Type', 'application/json; charset=UTF-8');
            xhr.onreadystatechange = function(data) {
                if (xhr.readyState == 4 && (xhr.status == 200 )) {
                    console.log(data);
                }
            };
            xhr.send(str_data);

            return false;
        }
    }
</script>
```



Ce qu'on a couvert

- L'envoi d'un formulaire en Ajax



Alphorm.com

Javascript et la communication
avec le serveur

Présentation des
services web Restful

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Frédéric GAURAT
Développeur et Formateur
Consultant indépendant
alphorm.com™©



Plan

- Rappel de la définition d'un service Web
- L'architecture REST



Qu'est-ce qu'un service Web

- D'après Wikipedia :
 - Un service web est un programme informatique de la famille des technologies web
 - Il permet la communication et l'échange de données



Le protocole HTTP : Les méthodes

GET	Demander une ressource
POST	Transmettre des données à une ressource
PUT	Remplacer ou ajouter une ressource
DELETE	Supprimer une ressource
HEAD	Demander des informations sur la ressource
OPTIONS	Connaître les options de communications d'une ressource
CONNECT	utiliser un proxy comme tunnel de communication
TRACE	permet d'effectuer des tests
PATCH	modification partielle d'une ressource



REST : Representational State Transfer

- L'architecture REST permet d'exploiter les méthodes HTTP associées à une URL...
- Lorsqu'un navigateur récupère (GET) une page (la ressource) et l'affiche on peut dire qu'il va récupérer la **représentation** courante de **l'état** de la ressource.
- **Exemple** : considérons cette URL : www.bibliotheque.fr/livres
 - **GET** sur www.bibliotheque.fr/livres/1 va **retourner** le livre dont l'id est 1.
 - **DELETE** sur www.bibliotheque.fr/livres/1 va **effacer** le livre dont l'id est 1.
 - **POST** sur www.bibliotheque.fr/livres/ permet de **créer** un nouveau livre
 - **PUT** sur www.bibliotheque.fr/livres/1 permet de **modifier** le livre dont l'id est 1.



Ce qu'on a couvert

- La définition d'un service Web
- L'architecture REST



Formation JavaScript, les fondamentaux

alphorm.com™©



Alphorm.com

Javascript et la communication
avec le serveur

Consommer des services web
Restful avec JavaScript : **CREATE**

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Frédéric GAURAT
Développeur et Formateur
Consultant indépendant
alphorm.com™©



Plan

- Développer une application CRUD
- Simuler une API REST



Développer une application CRUD

- Créer une application simple pour gérer une liste de client
 - 1. Create : Un formulaire
 - 2. Read : Une table qui affichera les données
 - 3. Update : un formulaire avec les données pré-rempli accessible à partir de la table
 - 4. Delete : un bouton de suppression sera disponible dans la table



Simuler une API REST

- Présentation de Deployd
- Installation de Deployd : <http://deployd.com/>
- Mise en place de l'API



Create : Un formulaire

- Construire le formulaire
- Coder la création du client



Ce qu'on a couvert

- La structure d'une application simple
- L'installation de deployd pour simuler une API REST
- Le développement du formulaire de création d'un client



Formation JavaScript, les fondamentaux

alphorm.com™©



Alphorm.com

Javascript et la communication
avec le serveur

Consommer des services web
Restful avec JavaScript : **READ**

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©



Plan

- Développer la partie d'affichage de nos clients



Développer une application CRUD

- Create : Une formulaire
- Read : Une table qui affichera les données
- Update : un formulaire avec les données pré-rempli accessible à partir de la table
- Delete : un bouton de suppression sera disponible dans la table



Ce qu'on a couvert

- Le développement de la table qui affichera les clients stockés dans le serveur



Formation JavaScript, les fondamentaux

alphorm.com™©



Alphorm.com

Javascript et la communication
avec le serveur

Consommer des services web
Restful avec JavaScript : **UPDATE**

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>

Formation JavaScript, les fondamentaux



Frédéric GAURAT
Développeur et Formateur
Consultant indépendant
alphorm.com™©



Plan

- Gérer la mise à jours de nos clients



Développer une application CRUD

- Create : Une formulaire
- Read : Une table qui affichera les données
- Update : un formulaire avec les données pré-rempli accessible à partir de la table
- Delete : un bouton de suppression sera disponible dans la table



Ce qu'on a couvert

- La mise à jours d'un client dans notre application



Alphorm.com

Javascript et la communication
avec le serveur

Consommer des services web
Restful avec JavaScript : **DELETE**

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant



Plan

- Gérer la suppression d'un client



Développer une application CRUD

- Create : Une formulaire
- Read : Une table qui affichera les données
- Update : un formulaire avec les données pré-rempli accessible à partir de la table
- Delete : un bouton de suppression sera disponible dans la table



Ce qu'on a couvert

- La suppression d'un client



Alphorm.com

JavaScript Les Fondamentaux

Conclusion



Frédéric GAURAT

Développeur et Formateur
Consultant indépendant

alphorm.com™©

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Plan

- Ce qui a été couvert
- Ce qui reste à aborder



Ce qu'on a couvert dans cette formation

- Introduction au développement Web
 - Etat de l'art des architectures web
 - Installation de l'environnement de développement
 - Les bases du langage
 - Les variables et leurs types, les tableaux, les conditions, les boucles, les fonctions, les erreurs
- Les pièges du langage
 - Notion de « Hoisting »
- Introduction à la programmation orienté objet
 - Notions d'objet (les classes, l'héritage)
 - Notions de prototype
 - Notions de design patterns
- Javascript dans le navigateur
 - Les objets disponibles
 - Le cycle de vie d'une page web
- Notion et utilisation des événements
 - Manipuler les formulaires, les pages web avec le DOM, les styles
- Javascript et la communication avec le serveur
 - Présentation d'Ajax et JSON
 - Soumettre un formulaire en Ajax
 - Présentation des services web Restful
 - Consommer des services web Restful avec Javascript : CREATE, READ, UPDATE, DELETE



Cursus formations JavaScript

JavaScript,
les fondamentaux

DART

Tests
Unitaires

AngularJS 2,
les fondamentaux

JavaScript,
Avancé

CoffeeScript

Tests
fonctionnels

AngularJS 2,
Avancé

TypeScript

Formation JavaScript, les fondamentaux

alphorm.com™©



La formation Javascript avancé

• Utilisation avancée des Fonctions

- Les différents types de fonction (anonymes, immédiates, internes)
- Les scopes et les closures

• Programmation Orienté Objet en JavaScript

- Rappel sur les prototypes
- Implémentation des constructeurs
- Implémentation de l'encapsulation
- L'héritage en JavaScript
- Utilisation du « this »
- Implémentation de Design Patterns

• Programmation asynchrones

- Le callback hell

▪ Les promesses

• Les tasks runner

- Les tâches de base (Qualité,Obfuscation,...)
- Présentation de Grunt
- Présentation de Gulp
- Présentation de Brunch

• Programmation Modulaire

- Organiser son code et le rendre performant : Asynchronous Module Definition (AMD)
- Utilisation de RequireJS

• Le futur de JavaScript

- CoffeeScript, Dart,TypeScript
- La spécification ECMAScript 6

Formation JavaScript, les fondamentaux

alphorm.com™©



A bientôt !



- . Site : www.eolem.com
- . Profil Alphorm : <http://www.alphorm.com/formateur/frederic-gaurat>

