Alphorm.com

JavaScript, *avancé*

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Présentation du formateur
- Les autres formations sur Alphorm
- Le plan de formation
- Présentation des outils
- Les publics concernés

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Présentation du formateur

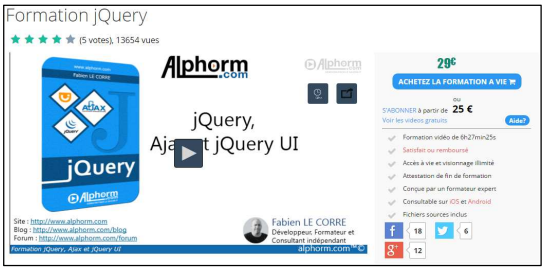
Frédéric GAURAT

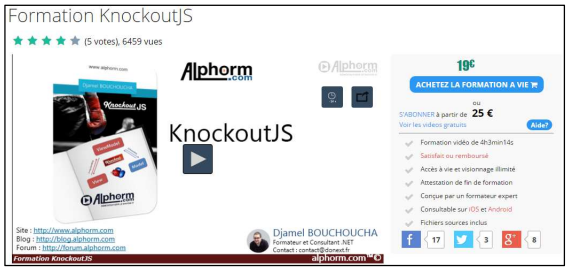
Développeur et formateur indépendant

- 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

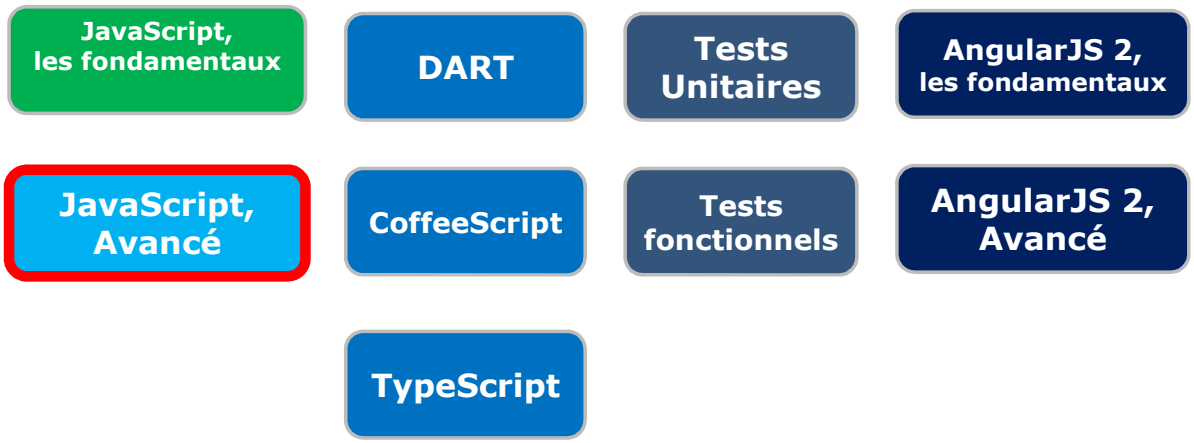
Les autres formations sur Alphorm





TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Cursus formations JavaScript



Le plan de formation

- Présentation de la formation
- Utilisation avancée des Fonctions
- Programmation Orienté Objet en JavaScript
- Programmation asynchrones
- Les tasks runner
- Programmation Modulaire
- Le futur de JavaScript

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Le plan de formation

- Présentation de la formation
- Utilisation avancée des Fonctions
- Programmation Orienté Objet en JavaScript
- Programmation asynchrones
- Les tasks runner
- Programmation Modulaire
- Le futur de JavaScript

Présentation des outils

Les éditeurs
- SublimeText
- Atom
- Microsoft Visual Studio Code
- NotePad++
- jsfiddle.net

Les navigateurs
- Chrome
- Firefox
- Internet Explorer

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les publics concernés

- Les développeurs et chefs de projets qui souhaitent approfondir leurs connaissances de JavaScript.

JavaScript Avancé

C'est parti !

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Utilisation avancée des Fonctions

Les différents types de fonction

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Rappel sur les fonctions
- Les fonctions anonymes
- Les fonctions callbacks
- Les fonctions immédiates
- Les fonctions internes
- Les scopes
- Les closures

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Rappel sur les fonctions

- Les fonctions permettent de factoriser du code pour permettre une réutilisation

```
function sum(a,b){
  return a+b;
}

var c = sum(1,2);
console.log(c);//3
```


Les fonctions anonymes

- Une fonction anonyme est une fonction qui n'a pas de nom

```
function(){
  alert('Hello');
}
```

- Très peu utilisée de cette façon, on préfère :

```
var a = function(){
  alert('Hello');
}
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les fonctions callbacks

- Les fonctions sont typées comme des variables
- On peut donc les passer en paramètre

```
var show = function(a){
  console.log('show : '+a);
}

function sum(a,b,affiche){
  var c = a+b;
  affiche(c);
  return c;
}

var result = sum (1,2,show);
```


Les fonctions immédiates (Self-invoking)

- Les fonctions immédiates sont une application des fonctions anonymes, mais appelées directement après leurs créations

```
(function(){
  var s="world"
  console.log("Hello "+s);
})();
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les fonctions internes (Inner)

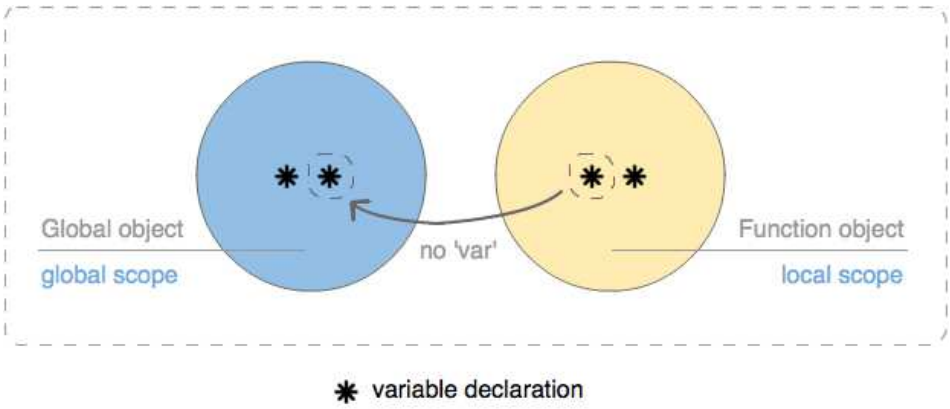
- Les fonctions sont typées comme des variables
- On peut donc les déclarer dans une fonction comme de simples variables

```
function sum(a,b,affiche){
  var c = a+b;
  var show = function(a){
    console.log('show : '+a);
  }

  affiche(c);
  return c;
}

var result = sum (1,2,show);
```


Les scopes



http://www.basing.com/books/javascript/how-variable-scope-works-in-javascript/2

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les closures

http://www.c-sharpcorner.com/UploadFile/dhananjaycoder/what-is-closure-in-javascript/

Ce qu'on a couvert

- Rappel sur les fonctions
- Les principales structures de fonctions
- Rapide présentation des scopes et closures

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Utilisation avancée des Fonctions

Les fonctions anonymes

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Les fonctions anonymes
- Exemple
- Mise en situation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les fonctions anonymes

- Structure fondamentale
- Elles permettent d'isoler des variables dans un contexte local
- Elles permettent de simuler un **namespace** ou **packages**

Examples

- Déclaration de base :

```
function(){
  alert('Hello');
}
```

- Mais peu pratique alors on stocke dans une variable :

```
var a = function(){
  alert('Hello');
}

a();
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Mise en situation

- Première approche pour la réalisation d'un namespace ou package

```
var namespace = function(){
  var a = "Hello";
  var b = "World";

  console.log(a+" "+b);
}

namespace();
```


Ce qu'on a couvert

- Les fonctions anonymes
- Exemple
- Mise en situation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Utilisation avancée des Fonctions

Les fonctions callbacks

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Les fonctions callbacks
- Exemple
- Mise en situation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les fonctions callbacks

- Egalement nommée fonction de rappel
- Elles sont exécutées au « bon moment »
- Elles sont souvent utilisées pour de l'évènementiel ou de l'Ajax

Exemple

- Fonction callback appelée lors du chargement de la page

```
var doLoad = function(){
  console.log("Window loaded");
}

window.onload = doLoad;
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Mise en situation

- CallBack et contexte d'exécution

```
<ul>
  <li>Li 1</li>
  <li>Li 2</li>
  <li>Li 3</li>
</ul>
```

Mauvaise méthode

```
var doLoad = function(){
  var elements = document.getElementsByTagName('li');;
  for(var i = 0; i < elements.length; i++) {

      elements[i].addEventListener( 'click', function() {
        console.log(i);
        elements[i].innerHTML = elements[i].innerHTML+' clicked';
      });
  }
}

window.onload = doLoad;
```

Bonne méthode

```
var doLoad = function(){
  var elements = document.getElementsByTagName('li');
  for(var i = 0; i < elements.length; i++) {
      elements[i].addEventListener( 'click', (function(i) {
        return function(){
          console.log(i);
          elements[i].innerHTML = elements[i].innerHTML+' clicked';
        };
      })(i));
  }
}
```


Ce qu'on a couvert

- Les fonctions callbacks
- Un exemple simple d'utilisation
- Mise en situation pratique qui intègre une fonction anonyme

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Utilisation avancée des Fonctions

Les fonctions immédiates

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Les fonctions immédiates
- Exemple
- Mise en situation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les fonctions immédiates (Self-invoking)

- Autre cas d'utilisation des fonctions anonymes
- Permet la limitation de la portée des variables tout en exécutant du code

Exemple

- Fonction anonyme normale :

```
function(){
  console.log('Do log');
}
```

- Fonction anonyme auto-appelée :

```
(
  function(){
    console.log('Do log');
  }
)();
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Mise en situation

- Fonction immédiate avec passage de paramètre

```
(
  function(w){
    console.log('Hello '+w);
  }
)('world');
```


Ce qu'on a couvert

- Les fonctions immédiates
- Exemple simple
- Mise en situation d'une fonction immédiate avec passage de paramètre

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Utilisation avancée des Fonctions

Les fonctions internes

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Les fonctions internes
- Exemple
- Mise en situation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les fonctions internes (Inner)

- Les fonctions sont des variables typées donc stockables dans une fonction !
- Elle peuvent servir pour déclarer des fonctions privées

Exemple

- Dans cet exemple MultPar2 n'est pas accessible.

```
function doMult(a) {
  function MultPar2(b) {
    return b * 2;
  };
  return 'résultat : ' + MultPar2(a);
};

var r = doMult(2);
console.log(r);
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Mise en situation

- Cas d'une fonction interne renvoyée par un return

```
function sayHello() {
  console.log('Hello');
  return function() {
    console.log('World');
  };
}
var func = sayHello();
func();
```


Ce qu'on a couvert

- Les fonctions internes
- Un exemple simple
- Cas d'un retour de fonction

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Utilisation avancée des Fonctions

Les Scopes

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Les scopes
- Exemple
- Mise en situation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les scopes

- En JavaScript, les portées sont définies par des fonctions et seulement par des fonctions
- Le « hoisting » vous assure qu'une variable déclarée tard dans le code sera remontée dans sa portée.

Exemple

- Différences de portée de variable

```
//global
var a = 3;
(
  function() {
    //local
    var b = 2;
  }
)();

console.log(a);
console.log(b);
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Mise en situation

- Le hoisting

```
"use strict";
var a = 1;
b = 2
console.log(a + b)
var b;
```

- Dans ce cas '**var b**' est remonté
- La syntaxe **use-strict** impose une rigueur de codage (utilisation de 'var' obligatoire par exemple)

Mise en situation (suite)

- Le hoisting

```
"use strict";
var a = 1;
console.log(a + b)
var b=2;
```

- Dans ce cas '**var b**' est remonté mais pas **b=2**, le résultat est **NaN**

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Mise en situation (suite)

- Le hoisting pour les fonctions

```
function start() {
  {
    doSomething(); // erreur : doSomething n'est pas défini sur firefox
    function doSomething() {
      console.log('doSomething');
    }
  }
}
start();
```

- Pas d'erreurs dans Chrome, en revanche ça ne fonctionne pas dans Firefox

Ce qu'on a couvert

- Les scopes, le hoisting
- Exemple
- Mise en situation du hoisting sur variables et fonctions

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Utilisation avancée des Fonctions

Les closures

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Les closures
- Exemple
- Mise en situation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les closures

- La notion de closure est associée à la notion de portée de variable
- Une variable est « enfermée » dans sa fonction (sa portée)
- Une fonction définie dans la même fonction (portée) va se « souvenir » de cette variable

https://developer.mozilla.org/fr/docs/Web/JavaScript/Closures

Les closures

- afficheNom() va se souvenir de la variable 'nom' déclarée dans la portée.

```
function créerFonction() {
  var nom = "Mozilla";

  function afficheNom() {
    alert(nom);
  }
  return afficheNom;
};

var maFonction = créerFonction();
maFonction();
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Mise en situation

- Ici, on stocke un tableau contenant le nom des jours.

```
function getDays() {
  var jours = ['Dimanche', 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi'];

  function getNomDuJours() {
    var numJour = new Date().getDay();
    console.log(jours[numJour]);
  }
  return getNomDuJours;
};

var maFonction = getDays();
maFonction();
```

- Dans ce cas le tableau n'est créé qu'une seule fois, pas de multiplication de ce tableau en mémoire

Ce qu'on a couvert

- Les closures
- Exemple d'utilisation
- Mise en situation illustrant une optimisation de l'utilisation mémoire

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

POO en JavaScript

Rappel sur les objets

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Les objets en JavaScript
- Des tableaux aux objets
- Accès aux propriétés

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les objets en JavaScript

- **Objet** : ensemble de propriétés.
- **Propriété** : association entre un nom (ou clé) et une valeur.
- **Cas particulier** :
  - Si la propriété est une fonction alors la propriété peut être appelée « méthode ».

Des tableaux aux objets

- Cas des tableaux

```
var jours = ['Dimanche', 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi'];

for(var i = 0; i<jours.length;i++){
  console.log(jours[i]);
}
```

- Pour accéder à un élément on utilise son index (sa clé numérique)

TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Des tableaux aux objets (suite)

- JavaScript ne supporte pas les tableaux associatifs
- Pour implémenter un tableau associatif, on passe par un objet :

```
var assoc_jours = {
  'jour_0':'Dimanche',
  'jour_1':'Lundi',
  'jour_2':'Mardi',
  'jour_3':'Mercredi',
  'jour_4':'Jeudi',
  'jour_5':'Vendredi',
  'jour_6':'Samedi'
};

//on utilise la notation tableau traditionnelle
console.log(assoc_jours['jour_1']);

//ou une boucle
for(key in assoc_jours){
  console.log(key+' : '+assoc_jours[key]);
}
```


Accès aux propriétés

- On utilise la notation « pointée »

```
var assoc_jours = {
  'jour_0':'Dimanche',
  'jour_1':'Lundi',
  'jour_2':'Mardi',
  'jour_3':'Mercredi',
  'jour_4':'Jeudi',
  'jour_5':'Vendredi',
  'jour_6':'Samedi'
};
console.dir(assoc_jours.jour_0);
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- Rappel sur les objets en JavaScript
- Comparaison des syntaxes des tableaux et des objets
- L'accès aux propriétés

Alphorm.com

POO en JavaScript

Rappel sur les prototypes

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

- Présentation de la notion de prototype
- Exemples

Présentation de la notion de prototype

- En JavaScript, chaque objet à un prototype
- Le prototype est une propriété de cet objet
- Le prototype lui-même est un objet
- Tous les objets JavaScript héritent leurs méthodes de leurs prototypes

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Notation litéral

- La propriété __proto__ donne un accès au prototype de l'objet

```
var client_1 = {
  nom:'DUPONT',
  prenom:'Robert'
};

var client_2 = {
  nom:'Martin',
  prenom:'Jean'
};

console.dir(client_1);
console.dir(client_2);
```

```
▼ Object
    nom: "DUPONT"
    prenom: "Robert"
  ▶ __proto__: Object
▼ Object
    nom: "Martin"
    prenom: "Jean"
  ▶ __proto__: Object
```


Notation avec constructeur

- Peu importe la façon de créer l'objet (littéral ou avec constructeur)

```
var client = function(n,p){
  this.nom = n;
  this.prenom = p;
}


var client_1 = new client('DUPONT','Robert');
var client_2 = new client('MARTIN','Jean');

console.dir(client_1);
console.dir(client_2);
```

```
▼ client
    nom: "DUPONT"
    prenom: "Robert"
  ▶ __proto__: Object
▼ client
    nom: "MARTIN"
    prenom: "Jean"
  ▶ __proto__: Object
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Ajout de propriétés et de méthodes

- JavaScript est dynamique, on peut ajouter des méthodes après déclaration et instanciation de l'objet.

```
var client = function(n,p){
  this.nom = n;
  this.prenom = p;
}


var client_1 = new client('DUPONT','Robert');
var client_2 = new client('MARTIN','Jean');

client_1.age = 40;
client_1.getAge = function(){
  return this.age;
}
console.log(client_1.getAge());

console.dir(client_1);
console.dir(client_2);
```

```
40
▼client
    age: 40
  ▶ getAge: function ()
    nom: "DUPONT"
    prenom: "Robert"
  ▶ __proto__: Object
▼client
    nom: "MARTIN"
    prenom: "Jean"
  ▶ __proto__: Object
```


Ajout de propriétés et de méthodes (suite)

- En ajoutant propriétés et méthodes sur le prototype c'est l'ensemble des objets qui en bénéficient

```
var client = function(n,p){
  this.nom = n;
  this.prenom = p;
}

client.prototype.age = 40;
client.prototype.getAge = function(){
  return this.age;
}

var client_1 = new client('DUPONT','Robert');
var client_2 = new client('MARTIN','Jean');

console.log(client_1.getAge());

console.dir(client_1);
console.dir(client_2);
```

```
40
▼client
    nom: "DUPONT"
    prenom: "Robert"
  ▼__proto__: Object
      age: 40
    ▶ constructor: function (n,p)
    ▶ getAge: function ()
    ▶ __proto__: Object
▼client
    nom: "MARTIN"
    prenom: "Jean"
  ▼__proto__: Object
      age: 40
    ▶ constructor: function (n,p)
    ▶ getAge: function ()
    ▶ __proto__: Object
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- Présentation de la notion de prototype
- Ajout de propriétés et méthodes sur un objet
- Ajout de propriétés et méthodes sur le prototype d'un objet

Alphorm.com

POO en JavaScript

Implémentation des constructeurs

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

- Présentation des fonctions comme constructeur d'objet
- La propriété **constructor**
- Exemples

Fonctions comme constructeur d'objet

- JavaScript n'a pas de notion de classe.
- Pour construire un objet on passe par une fonction constructeur
- Le constructeur est appelé au moment de l'instanciation et toutes les opérations qui y sont déclarées sont exécutées

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

La propriété constructor

- La propriété **constructor** renvoie la référence à la fonction utilisée pour créer le prototype de l'instance

Exemple

- Utilisation d'un constructeur

```
var client = function(){
  console.log('Construction de client');
};

var c = new client();
console.dir(c);
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Exemple

- Affichage de la fonction utilisée pour construire l'objet

```
var client = function(){
  console.log('Construction de client');
};

var c = new client();
console.dir(c.constructor);
```


Exemple

- Modifier un constructeur en utilisant la propriété **constructor**
- En réalité on ne change pas le constructeur mais seulement ce que renvoie la propriété **constructor** (il ne faut pas se fier à ce que renvoie **constructor**)

```
var client = function(){
  console.log('Construction de client');
};

function newConstructor(){
  console.log('Construction de client avec le nouveau constructeur');
}


client.prototype.constructor = newConstructor;
var c = new client();
console.dir(c.constructor);
```

```
Construction de client
▶ function newConstructor()
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- L'utilisation des fonctions constructeurs
- La propriété **constructor**

Alphorm.com

POO en JavaScript

Implémentation de l'encapsulation

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

- Encapsuler des propriétés et méthodes
- Simuler les propriétés et méthodes **publics**
- Simuler les propriétés et méthodes **privates**

Encapsuler des propriétés et méthodes

- JavaScript ne dispose pas de moyens pour déclarer des propriétés ou méthodes **public**, **private** ou **protected**
- Pour simuler ce comportement nous allons exploiter le comportement normal de la déclaration de variable

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Simuler les propriétés et méthodes publics

- Pour déclarer des méthodes ou propriétés **public** ou utilise « **this** »

```
var client = function(n,p){
  this.nom = n;
  this.prenom = p;

  this.affiche = function(){
    console.log(this.nom+" "+this.prenom);
  }
};

var c = new client("DUPONT","Robert");
console.log(c.nom);
console.log(c.prenom);
c.affiche();
```


Simuler les propriétés et méthodes privates

- Pour déclarer des méthodes ou propriétés **privates** ou utilise « **var** »

```
var client = function(n,p){
  var nom = n;
  var prenom = p;

  this.affiche = function(){
    console.log(nom+" "+prenom);
  }
};

var c = new client("DUPONT","Robert");
console.log(c.nom);     // undefined
console.log(c.prenom);  // undefined
c.affiche();
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- Les moyens de simuler les « privates » et « public » que l'on trouve dans d'autres langages.

Alphorm.com

POO en JavaScript

L'héritage en JavaScript

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

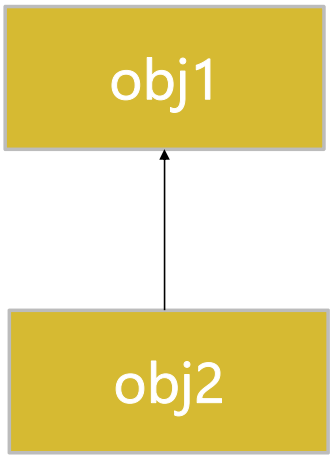
- Implémenter l'héritage en JavaScript
- Exemples

Implémenter l'héritage en JavaScript

- L'héritage permet de créer des objets spécialisés à partir d'un autre objet. JavaScript s'appuie sur l'héritage unique.
- Pour implémenter l'héritage, on assigne une instance de l'objet parent à la **propriété prototype de l'objet fils**.
- On parle donc d'héritage de prototype et de « **prototype chain** »
- Pour éviter que le constructeur soit celui du parent on réassigne le constructeur à l'enfant.

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Exemple d'héritage



Exemple d'héritage

```
var obj1 = function(){
  this.name='obj1';
  this.toString = function() {return this.name;};
}

var obj2 = function(){
  this.name='obj2';
}

obj2.prototype = new obj1();
obj2.prototype.constructor = obj2;


var o = new obj2();
console.log(o.toString());
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- La notion d'héritage de prototype en JavaScript

Alphorm.com

POO en JavaScript

Utilisation du « this »

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

- Définition et utilisation de « this »
- Exemples

Définition et utilisation de « this »

- Le « **this** » est une référence à un objet (une fonction en JavaScript)
- La valeur de this est déterminée par la façon dont on appelle la fonction.
- Dans le contexte global : this fait référence à l'objet global

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Dans un contexte de fonction

- En **mode non strict**, this n'étant pas défini il prend l'objet global

```
function func(){
  console.dir(this);
}

func();
```

```
▶ Window
```

- En **mode strict**, this n'étant pas défini il la valeur **undefined**

```
"use strict";
function func(){
  console.dir(this);
}

func();
```

```
▶ undefined
```


Dans un contexte de méthode

- this est l'objet à qui appartient la méthode

```
var obj = {
  name:'obj',
  getName:function(){
    console.log(this);
    return this.name;
  }
}
obj.getName();
```

```
▼ Object {name: "obj"}
  ▶ getName: function ()
    name: "obj"
  ▶ __proto__: Object
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Dans un contexte de constructeur

- this représente l'objet nouvellement créée

```
var obj = function(){
  this.name="obj";

  this.getName=function(){
    console.log(this);
    return this.name;
  }
}

  var o = new obj();
  o.getName();
```

```
▶ obj {name: "obj"}
```


Dans un context asynchrone

- this prend la valeur au moment de l'éxécution du code asynchrone

```
var obj = function() {
  this.name = "obj";

  this.getName = function() {

    setTimeout(function() {
      console.log("2000ms plus tard ...");
      console.dir(this);
    }, 2000);
    console.log(this);
    return this.name;
  }
}

var o = new obj();
o.getName();
```

```
▶ obj {name: "obj"}
2000ms plus tard ...
▶ Window
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- Le mot clé this dans des contextes d'utilisations différents

Alphorm.com

Programmation asynchrones

Le callback hell ou Pyramid of doom

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

- Définition du callback hell ou pyramid of doom
- Exemples

Définition du callback hell

- Le callback hell apparait lorsqu'un développeur tente de synchroniser un code asynchrone.

```
getTodo(1,function(resultToDo1){
  getInfoTodo(resultToDo1,function(resultInfoToDo2){
    getInfoTodo(resultInfoToDo2,function(resultInfoToDo3){
      // ...
    });
  });
});
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- La définition d'un callback hell

Alphorm.com

Programmation asynchrones

Les Promesses

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

- Définition d'une promesse
- Exemples

Définition d'une promesse

- Une promesse est utilisée pour réaliser des opérations asynchrone.
- Elle peut être dans un de ces états :
  - **en attente** : état initial, la promesse n'est ni remplie, ni rompue
  - **tenue** : l'opération a réussi
  - **rompue** : l'opération a échoué
  - **acquittée** : la promesse est tenue ou rompue mais elle n'est plus en attente.

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Promise

TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Exemple

- Nous souhaitons faire la somme de 2 entiers.
- Ces entiers seront calculés plus tard (on simule une requête serveur)

```
var a = undefined;
var b = undefined;

setTimeout(function() {
  a = 1;
}, 2000);

setTimeout(function() {
  b = 2;
}, 3000);

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

```
NaN
```


Exemple du callback hell

- Pour synchroniser les traitements nous exécuterons les traitements l'un après l'autre.

```
var a = undefined;
var b = undefined;

setTimeout(function() {
  a = 1;
  setTimeout(function() {
    b = 2;
    var c = a + b;
    console.log(c);

  }, 3000);
}, 2000);
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Exemple avec des promesses

- Etape 1 : on créée un objet Promise
- Etape 2 : on attend que la promesse soit tenue (resolve) avec la méthode then

```
var p1 = new Promise(function(resolve, reject) {
  setTimeout(function() {
    var a = 1;
    resolve(a);
  }, 2000);
});

p1.then(function(val_a){
  console.log(val_a);
});
```


Exemple avec des promesses

- Pour synchroniser les 2 traitements :
  - Lorsqu'une promesse est résolue on enchaîne avec l'autre

```
var p1 = new Promise(function(resolve, reject) {
  setTimeout(function() {
    var a = 1;
    resolve(a);
  }, 2000);
});

p1
  .then(function(result_a) {
    var p2 = new Promise(function(resolve, reject) {
      setTimeout(function() {
        var b = 2;
        resolve(b + result_a);
      }, 3000);
    });
    return p2;
  })
  .then(function(result_a_b) {
    console.log(result_a_b);
  });
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Exemple avec des promesses

- Pour synchroniser les traitements on peut passer par la méthode **all()** qui reçoit un tableau de promesses et qui sera résolue lorsque toutes les promesses du tableau le seront.

```
var p1 = new Promise(function(resolve, reject) {
  setTimeout(function() {
    var a = 1;
    resolve(a);
  }, 2000);
});

var p2 = new Promise(function(resolve, reject) {
  setTimeout(function() {
    var b = 2;
    resolve(b + result_a);
  }, 3000);
});

var arr = [p1,p2];

Promise.all(arr).then(function(results){
  console.dir(results);
});
```


Ce qu'on a couvert

- Le fonctionnement et l'utilisation des promesses
- Un exemple d'utilisation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les tasks runner

Présentation et tâches de base

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

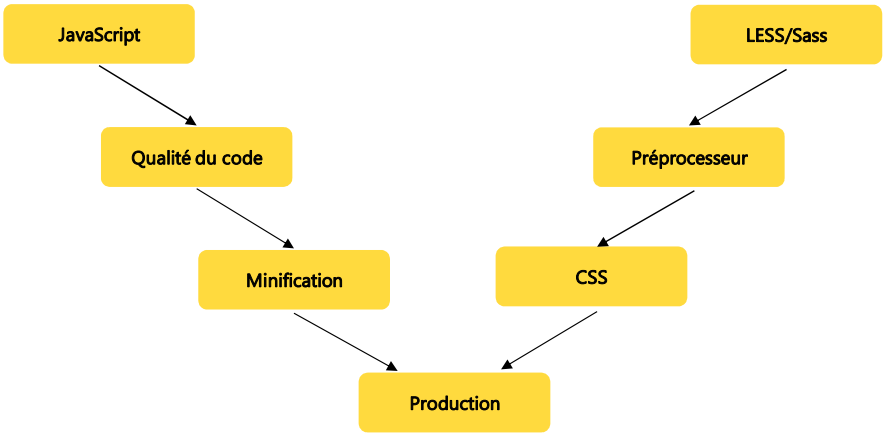
Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

Plan

- Les besoins des développeurs JavaScript
- Les outils

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Les besoins des développeurs JavaScript



Les outils

- Exécuteurs de tâches

TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- L'intérêts d'utiliser des exécuteurs de tâches
- Les principaux outils

Alphorm.com

Les tasks runner

GruntJS

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

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

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

- Présentation de GruntJS
- Utilisation

GruntJS

- GruntJS est un exécuteur de tâches
- Il s'appuie sur des **plugins**
- Ainsi que sur un fichier (**GruntFile**) pour configurer les plugins

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Installation

- L'installation se fait avec l'utilitaire npm
- L'interface grunt-cli

```
npm install -g grunt-cli
```

- L'outils est à installer directement dans votre projet

```
npm install grunt --save-dev
```


Installation d'un plugin

- Pour une tâche de minification le plugin est uglify
  - http://gruntjs.com/plugins pour la totalité de plugins
  - https://www.npmjs.com/package/grunt-contrib-uglify pour le plugin uglify
- Installation du plugin

```
npm install grunt-contrib-uglify --save-dev
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Le fichier Gruntfile.js

- Structure de base

```
module.exports = function(grunt) {

  grunt.initConfig({

  });

};
```


Configuration d'un plugin

- Chargement du plugin

```
// Load the plugin that provides the "uglify" task.
grunt.loadNpmTasks('grunt-contrib-uglify');
```

- Enregistrement de la tâche

```
// Default task(s).
grunt.registerTask('default', ['uglify']);
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Configuration d'un plugin (suite)

```
module.exports = function(grunt) {


  grunt.initConfig({
    uglify: {
      my_target: {
        files: {
          'dest/output.min.js': ['src/input.js']
        }
      }
    }
  });

  grunt.registerTask('default', ['uglify']);

  grunt.loadNpmTasks('grunt-contrib-uglify');

};
```


Ce qu'on a couvert

- L'exécuteur de tâche GruntJS
- L'installation
- L'installation et la configuration d'un plugin

TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Alphorm.com

Les tasks runner

GulpJS

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

Frédéric GAURAT
Formateur et Consultant indépendant
Ingénierie statistique et financière

Plan

- Présentation de GulpJS
- Utilisation de GulpJS

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

GulpJS

- GulpJS est un exécuteur de tâches
- Il s'appuie sur des plugins
- Ainsi que sur un fichier (**gulpfile.js**) pour configurer les plugins
- Il se diffère de **GruntJS** dans son mode de fonctionnement :
  - **Grunt** s'appuie sur des **fichiers** pour la réalisation des tâches
  - **GulpJS** s'appuie sur des **streams** (des fluxs entre les tâches)

Installation

- L'installation se fait avec l'utilitaire npm

```
$ npm install --global gulp-cli
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Installation d'un plugin

- Pour une tâche de minification le plugin est **uglify**
  - http://gulpjs.com/plugins/ pour la totalité des plugins
  - https://www.npmjs.com/package/gulp-uglify/ pour le plugin uglify
- Installation du plugin

```
npm install --save-dev gulp-uglify
```


Le fichier gulpfile.js

```
var gulp = require('gulp');

gulp.task('default', function() {
  // place code for your default task here
});
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Configuration d'un plugin

- Installation du plugin

```
npm install --save-dev gulp-uglify
```

- Chargement du plugin

```
var uglify = require('gulp-uglify');
```

- Enregistrement de la tâche

```
gulp.task('compress', function() {
  return gulp.src('lib/*.js')
    .pipe(uglify())
    .pipe(gulp.dest('dist'));
});
```


Configuration d'un plugin (suite)

```
var uglify = require('gulp-uglify');

gulp.task('compress', function() {
  return gulp.src('lib/*.js')
    .pipe(uglify())
    .pipe(gulp.dest('dist'));
});
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- L'exécuteur de tâche GulpJS
- L'installation
- L'installation et la configuration d'un plugin

Alphorm.com

Programmation modulaire

Organiser son code
et le rendre performant

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

Frédéric GAURAT
Formateur et Consultant indépendant
Ingénierie statistique et financière

TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Plan

- Introduction à la programmation modulaire
- Rappel sur le pattern Module
- Les problèmes et les solutions

La programmation modulaire

- En JavaScript il n'y a pas d'instruction pour créer des modules
- Il faut utiliser le pattern module
- Il n'y a pas de gestion de dépendance
- L'ensemble des scripts doivent être chargés au démarrage

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Le pattern module

- Le pattern module permet d'encapsuler du code comme un package ou namespace
- Il s'appuie sur une « **Anonymous Closure** »
  - Permet la déclaration de variable privée
  - Permet de maintenir l'état des variables

Le pattern module (suite)

- Anonymous Closure

```
(function () {
    var a;//private

}());
```

- Ajout d'une dépendance

```
var obj = {name:'MARTIN',firstname:'Jean'};

(function (d) {
  var dep = d;

}(obj));
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Le pattern module (suite)

- Module export

```
var obj = {name:'MARTIN',firstname:'Jean'};

var le_module = (function (d) {
  var dep = d;
  var innerObj = {};

  innerObj.sayHello=function(){
    return "Hello";
  };

  return innerObj;
}(obj));

var s =le_module.sayHello();
console.log(s);
```


Les problèmes

- Les modules sont des unités de code indépendant les uns des autres
- Ils sont stockés dans des fichiers distincts
- Difficile de gérer les dépendances
- Difficile de charger les fichiers au bon moment

TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Les solutions

- Asynchronous module definition (AMD)
  - Spécification JavaScript
  - Définit une API pour décrire des modules, gérer leurs dépendances et les charger dynamiquement (éventuellement)

```
define('le_module', ['module1'], function (module1) {

  var dep = module1;
  var innerObj = {};

  innerObj.sayHello=function(){
    return "Hello";
  };

  return innerObj;
});
```


Ce qu'on a couvert

- La programmation modulaire
- Le pattern module
- La spécification AMD

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Programmation Modulaire

Utilisation de RequireJS

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

Frédéric GAURAT
Formateur et Consultant indépendant
Ingénierie statistique et financière

Plan

- Présentation de RequireJS
- Utilisation de RequireJS

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Présentation de RequireJS

- **RequireJS** est un outil permettant le chargement de fichiers et de modules JavaScript
- http://requirejs.org/

Utilisation de RequireJS (initialisation)

- Chargement de RequireJS et de notre module principal

```
<script data-main="fr_06_01_00" src="require.js"></script>
```

- Chargement de notre application

```
requirejs(['fr_06_01_00_main']);
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Utilisation de RequireJS (utilisation)

- Définition du module

```
define(function () {
  return {
     print: function (msg) {
         console.log(msg);
     }
 };
});
```

- Utilisation du module

```
define(function (require) {

    var mod = require('fr_06_01_00_module_print');

    mod.print("OK");
});
```


Ce qu'on a couvert

- Le framework RequireJS
- Définition de module
- Chargement et utilisation de notre module

TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Alphorm.com

Le Futur de JavaScript

CoffeeScript

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

Frédéric GAURAT
Formateur et Consultant indépendant
Ingénierie statistique et financière

Plan

- Le langage CoffeeScript
- Installation et utilisation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Le langage CoffeeScript

- CoffeeScript est un langage permettant, après compilation de générer du JavaScript
- L'objectif est de permettre une utilisation simple de JavaScript

Installation

- Le compilateur s'installe via npm :

```
npm install -g coffee-script
```

- Il est possible de tester directement en ligne sur le site :
  - http://coffeescript.org/ ('Try Coffeescript')

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Utilisation

- Appel de fonction

```
alert "Hello CoffeeScript!"
```

```
alert("Hello CoffeeScript!");
```

- Déclaration de fonction

```
getHello = (name)-> "Hello "+name
```

```
var getHello;

getHello = function(name) {
  return "Hello " + name;
};
```


Utilisation

- Déclaration de fonction (suite)

```
getHello = (name)->
 console.log "Hello "+name
 "Hello "+name
```

```
var getHello;

getHello = function(name) {
  console.log("Hello " + name);
  return "Hello " + name;
};
```

- Déclaration de variable et portée lexicale

```
vGlobale = 1
maFonction = ->
  vLocale = -1
  vGlobale = 10

vLocale = maFonction()
```

```
var maFonction, vGlobale, vLocale;

vGlobale = 1;

maFonction = function() {
  var vLocale;
  vLocale = -1;
  return vGlobale = 10;
};

vLocale = maFonction();
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Utilisation

- Les boucles

```
affiche valeur for valeur in ['val1', 'val2', 'val3']
```

```
var i, len, ref, valeur;

ref = ['val1', 'val2', 'val3'];
for (i = 0, len = ref.length; i < len; i++) {
  valeur = ref[i];
  affiche(valeur);
}
```


Ce qu'on a couvert

- Une introduction à CoffeeScript
- Quelques exemples d'utilisation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Le Futur de JavaScript

Dart

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

Frédéric GAURAT
Formateur et Consultant indépendant
Ingénierie statistique et financière

Plan

- Présentation de Dart
- Utilisation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Présentation de Dart

- Langage de développement web (objet) proposé par Google
- Il a vocation à remplacer le JavaScript
- Pour le moment il permet d'obtenir du JavaScript après compilation
- Il peut tourner dans une VM côté serveur
- Il est testable en ligne : https://dartpad.dartlang.org
- Contrairement aux autres outils, il ne s'installe pas avec npm
- https://www.dartlang.org/downloads/

Exemple

- Exemple HelloWorld

```
sayhello(String name) {
  print('Hello $name.');
}

main() {
  var name = "Alphorm";
  sayhello(name);
}
```

```
Hello Alphorm.
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Déclaration de variable

- Il est possible d'annoter les variables avec un type

```
var name = 'Alphorm';
String name = 'Alphorm';
```


Déclaration de fonction

- Les type s'appliquent aussi aux fonctions et les déclarations sont soumises au compilateur

```
String getHello(String name) {
  return "Hello "+name;
}

main(){
  String h = getHello('Alphorm');
  print(h);
}
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Déclaration de classe

- La déclaration de classe se fait de la même façon qu'en Java ou C#

```
class Rectangle {
  num lng;
  num lrg;

  Rectangle(num a, num b) {
    this.lng = a;
    this.lrg = b;
  }
}

main(){
  Rectangle r = new Rectangle(2,3);
  print(r);
}
```


Ce qu'on a couvert

- Une introduction à Dart
- Quelques exemples d'utilisation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Alphorm.com

Le Futur de JavaScript

TypeScript

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

Frédéric GAURAT
Formateur et Consultant indépendant
Ingénierie statistique et financière

Plan

- Présentation de TypeScript
- Exemples d'utilisation

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Présentation de TypeScript

- TypeScript est un langage développé par Microsoft
- Il a pour but d'améliorer la production de code JavaScript
- Il intègre JavaScript (JavaScript est utilisable dans TypeScript)
- Il a été utilisé pour le développement d'Angular 2.
- Il est testable en ligne : http://www.typescriptlang.org/play/index.html

Installation et utilisation

INSTALL

```
npm install -g typescript
```

COMPILE

```
tsc helloworld.ts
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Exemple

- Les variables sont typées

```
function getHello(name: string) {
    return "Hello, " + name;
}

var s = "Alphorm";
var r = getHello(s);
console.log(r)
```


Déclaration de class

```
class Rectangle{
    longueur:number;
    largeur:number;

     constructor(a:number,b:number) {
         this.longueur = a;
         this.largeur = b;
    }
}


Rectangle r = new Rectangle(2,3);

console.log(r);
```


TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- Présentation de TypeScript
- Quelques exemples d'utilisation

Alphorm.com

Le Futur de JavaScript

ECMAScript6

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

Frédéric GAURAT
Formateur et Consultant indépendant
Ingénierie statistique et financière

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

- Présentation de ECMA
- Présentation de ECMAScript6
- Exemples d'utilisation

Présentation de ECMA

- ECMA (European Computer Manufacturers Association) est une organisation de standardisation.
- Elle est à l'origine des standards pour Dart (ECMA-408)
- ECMAScript (ECMA-262) mis en œuvre dans : ActionScript (Adobe), JavaScript, C++ (norme 2011).

TOUTE UTILISATION EN DEHORS
DE ALPHORM EST INTERDITE

Présentation de ECMAScript6

- Aussi nommée ECMSScript 2015 (sortie en Juin 2015)
- N'est pas encore supporté par les navigateurs du moment.
- Il est possible de passer par transcompilateur (compilateur de source à source) pour transformer l'ECMAScript6 en ECMAScript5 avec http://babeljs.io/

Présentation de ECMAScript6 (suite)

- Portée de variable avec let

```
1  var b = 3;
2  if(true){
3    let b = 3;
4    console.log(b);
5  }
6
7
```

```
1  "use strict";
2
3  var b = 3;
4  if (true) {
5    var _b = 3;
6    console.log(_b);
7  }
```

- Template String

```
1  var name = 'DUPONT';
2  var str = `Hello ${name}`
3
4
```

```
1  'use strict';
2
3  var name = 'DUPONT';
4  var str = 'Hello ' + name;
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Présentation de ECMAScript6 (suite)

- Paramètre de fonction par défaut

```
var f = function(i = 1,j=1){
  return i+j;
}

var r = f();
```

```
"use strict";

var f = function f() {
  var i = arguments.length <= 0 || arguments[0] === undefined ? 1 : arguments[0];
  var j = arguments.length <= 1 || arguments[1] === undefined ? 1 : arguments[1];

  return i + j;
};

var r = f();
```

- Les Lambdas (Arrow function)

```
var sayHello = (name) => "Hello "+name;
```

```
"use strict";

var sayHello = function sayHello(name) {
  return "Hello " + name;
};
```


Présentation de ECMAScript6 (suite)

- Les classes et héritage

```
class Rectangle {
    constructor(lng, lrg) {
        this.lng = lng;
        this.lrg = lrg;
    }
    toString() {
        return '(' + this.lng + ', ' + this.lrg + ')';
    }
}

class Carre extends Rectangle {
        constructor(cote) {
            super(cote, cote);
            this.cote = cote;
        }
        toString() {
            return super.toString() + ' Carre ' + this.cote;
        }
    }

    let cp = new Carre(8);
    console.log(cp.toString());

    console.log(cp instanceof Carre); // true
    console.log(cp instanceof Rectangle); // true
```


TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Ce qu'on a couvert

- L'organisme de standardisation ECMA
- L'ECMAScript6
- Les principales nouveautés de JavaScript ECMAScript 6

Alphorm.com

JavaScript, avancé

Conclusion

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

Frédéric GAURAT
Développeur et Formateur
Consultant indépendant

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Plan

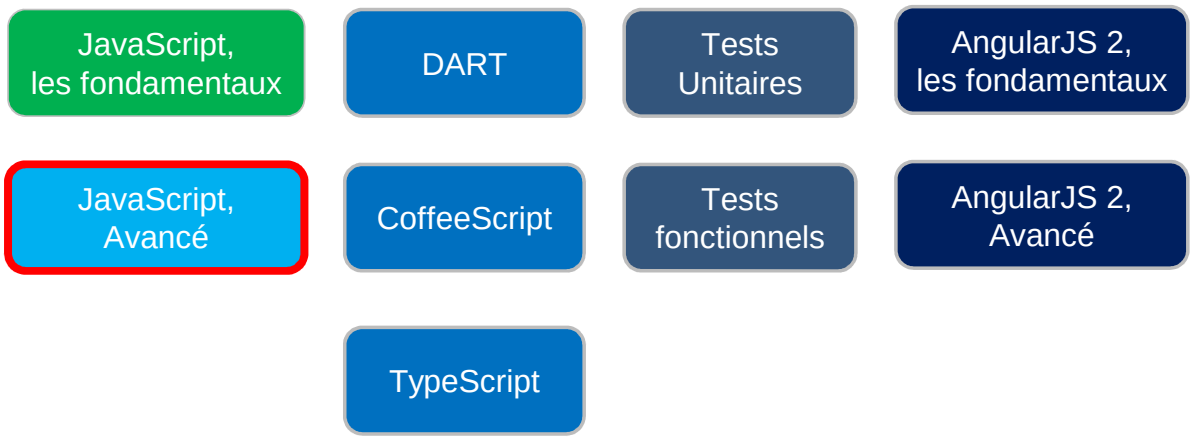
- Ce qui a été couvert
- Ce qui reste à aborder

Ce qu'on a couvert dans cette formation

- **Présentation de la formation**
  - Présentation de la formation
- **Utilisation avancée des Fonctions**
  - Les différents types de fonction (anonymes, callback, immédiates, internes)
  - Les scopes et les closures
- **Programmation Orienté Objet en JavaScript**
  - Rappel sur les objets et prototypes
  - Implémentation des constructeurs
  - Implémentation de l'encapsulation
  - L'héritage en JavaScript
  - Utilisation du « this »
- **Programmation asynchrones**
  - Le callback hell
  - Les promesses
- **Les tasks runner**
  - Les tâches de base (Qualité,Obfuscation,...)
  - Présentation de Grunt et Gulp
- **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

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE

Cursus formations JavaScript



A bientôt !

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

TOUTE UTILISATION EN DEHORS DE ALPHORM EST INTERDITE