S'INITIER AUX FRAMEWORKS JAVASCRIPT : BACKBONE.JS/ANGULAR/REACTJS/VUE.JS

8 jours (56 heures en présentiel ou 56 heures à distance en classe virtuelle)

Objectifs pédagogiques

Cette formation vous permet d'optimiser des développements en s'appuyant sur des frameworks et librairies actuels : Backbone.js, AngulatJS, ReactJS et Vue.js.

Population visée

Développeurs front et back-end.

Pré-requis

Une bonne maîtrise de la programmation orientée objets en JavaScript est indispensable.

Procédures de positionnement et d'évaluation des acquis à l'entrée de la prestation

Audit téléphonique d'un conseil-formation pour s'assurer des pré-requis et des besoins de l'apprenant, complété d'un audit de niveau via un formulaire à remplir, soumis à l'analyse du formateur-référent.

Méthodes pédagogiques

8 participants maximum, un poste par stagiaire et un support de cours est envoyé en fin de stage (vidéos tutorielles et/ou support spécifique). La formation est constituée d'apports théoriques, de démonstrations et de mises en pratique basées sur des exercices applicatifs et/ou ateliers.

Formateur

Formateur expérimenté, développeur de site Web, et spécialiste du Web.

Modalités de validation des acquis

Évaluation continue via des exercices applicatifs et/ou des ateliers de mise en pratique.
Évaluation en fin de stage par la complétion d'un questionnaire et/ou d'une certification officielle issue du Répertoire Spécifique.
Émargement quotidien d'une feuille de présence (en présentiel ou en ligne).
Complétion par le formateur/la formatrice d'un suivi d'acquisition des objectifs pédagogiques.
Remise d'une attestation individuelle de réalisation.

Contenu

Rappels

- Le couple HTML/CSS pour créer un document Web
- Le javascript pour ajouter des fonctionnalités
- Le javascript côté client et côté serveur
- Le navigateur et sa console
- Le DOM (Document Object Modèle) et les APIs
- Les IDE et éditeurs
- Les design patterns (singleton, observer, factory, MVC, …)

Les frameworks et les librairies

- Différence entre librairie et framework
- Les différents frameworks Javascript
- Pourquoi utiliser un framework ?
- Quand choisir d'utiliser un framework ?
- Quand choisir d'utiliser une librairie ?
- Compatibilité et frameworks multiples

BackboneJS

Backbone et le MVC

- Architecture d'une Single Page Application (SPA)
- Server-side vs Client-side
- MVC, MVP et MVVM
- Concurrents : Angular, Ember et Vue
- Backbone, Underscore et jQuery

Conception côté client

- Gestion du contexte
- Routage et navigation
- Authentification et autorisation

Modèle et collection

- Constructeurs et valeurs par défaut
- Structure interne des objets du modèle
- Getter et Setter tout en un
- Notification des changements

Vues et templates

- Propriétés des vues
- Evènements et réaffichage
- Templates underscore
- Mustache ou HandleBars

Router

- Définition des routes
- Gestion des URL et paramètres
- Evènements de navigation
- Bénéfices du routeur
- Instanciation et gestion des vues

Synchronisation des données

- Appels Ajax / JSON
- Backbone Sync
- Liens avec une API REST
- Stockage local

Extensions

- Intégration d'autres frameworks
- Backbone et jQuery
- Générateurs type Thorax
- Découverte de Marionnette

AngularJS

Présentation du framework

- La nouvelle version d'Angular
- Les nouveautés de TypeScript 4.0
- Le nouveau moteur d'Angular : Ivy
- Compilation AOT vs JIT

Architecture d'une application Angular

- Organiser son code avec les modules
- Les components et les templates
- Connecter components et templates avec le Data Binding
- Le rôle des directives
- Les services
- Notion d'injection de dépendance

Une première application Angular

- La structure d'un projet Angular
- Les modules et composants
- Démarrer "from scratch" avec Angular CLI
- Utilisation d'Angular CLI
- Création de projet
- Création de modules, components et services

Les templates

- Utiliser l'interpolation
- Property et event bindings
- Utiliser des variables locales
- Utilisation des pipes

Les formulaires

- Création de formulaires avec :
  - Le FormsModule
  - Le FormBuilder
- Validation et gestion des erreurs

La bibliothèque RxJS (Reactive extensions for JavaScript)

- La programmation réactive
- Observable et Observer
- Utilisation des opérateurs
- Communication entre components et les Subjects

Travail avec HTTP

- Le service HTTP
- Utilisation de RxJS
- Récupérer des données
- Rappel sur les Promises
- Utiliser les Observables

Le routage

- Les différentes versions du module de routage
- Fonctionnement du routage
- Configurer des routes et utiliser les directives

ReactJS

Présentation de ReactJS

- Positionnement de ReactJS
- Virtual DOM avec ReactJS
- Mise en place des outils de développement
- Tour d'horizon des outils de développement et d'intégration actuelle
- Création d'une application React avec le script "create-react-app"

Composants ReactJS

- Création d'un composant ReactJS
- Amélioration des fonctionnalités du composant développé
- Etats d'un composant et cycle de vie
- Gestion de l'état d'un composant
- Propriétés d'un composant
- Présentation de JSX et ES2015, que choisir ?
- Présentation approfondie du Virtual DOM

Communication inter-composants avec ReactJS

- Communication inter-composants
- Gestion des évènements
- Autobinding
- Composants de formulaire
- Manipulation du DOM
- Présentation de la propagation des données
- Flux des données
- Présentation des vues et contrôleurs dans ReactJS
- Création d'une application Single Page Application (SPA) avec ReactJS

Echanges avec le serveur

- Présentation de l'architecture REST
- Echanges entre l'application React et un serveur via REST

Les Hooks

- Présentation des Hooks
- Utiliser la state dans une fonction
- Les fonctions useState et useEffect

Améliorer une application ReactJS

- Gestion des erreurs avec les "Error Boundaries"
- Préserver la structure de l'arbre DOM avec les fragments
- Utiliser le contexte pour s'affranchir de la structure de l'arbre DOM
- Développer une application React avec TypeScript

Quelques patterns ReactJS

- Faire remonter l'état : Lifting State Up
- Le pattern Décorateur de ReactJS : Higher-Order Components

Redux

- Présentation du workflow
- Présentation de flux
- Eléments composants Redux
- Intégration de Redux dans React
- Les Hooks de Redux

Vue.js

Présentation de Vue.js

- Migration vers Vue.js
- MVVM selon Vue.js
- Les outils nécessaires liés à l'utilisation de Vue.js
- La gestion des interfaces graphiques par les données
- La liaison de données
- Les structures de contrôle
    - Répétitives
    - Alternatives
- Notion d'event

- Notion de component

Les essentiels de Vue.js

- Cycle de vie d'une requête dans Vue.js et son rôle
- Les modèles
- Notion d'interpolation
- Attributs, filtres et directives
- Les propriétés calculées
- Différence entre v-model et v-bind
- Le v-model
- Gérer des listes
- L'affichage conditionnel
- Event management
- Les composants dynamiques
- Echange de données entre composants

Aspects avancés de Vue.js

- Transitions et transitions CSS
- Transitions dynamiques
- Diverses animations
- Les fonctions "render"
- Les divers types de composants
    - Les composants fonctionnels
    - Les Single File Components
- Les différents types de directives
- Créer une directive adaptée

Les extensions

- Type mixin
- Type plug-in
- Les extensions (composants) monofichier

Le routing dans Vue.js

- Présentation des routes
- Les modes
- Les liens
- Vue initiale et liens