ANNEXE 5 – CURSUS FULLSTACK JAVASCRIPT

Préparation **O**pérationnelle à l'**E**mploi **C**ollective

Secteur du Numérique en région Provence Alpes Côte d'Azur

Table des matières

1 CURSUS DE FORMATION

Nous vous proposons le cursus ci-dessous. A votre demande, des variantes peuvent être apportées.

A ce cursus peut s'intégrer la certification SCRUM. Dans ce cas, le planning sera aménagé pour ne pas empiéter sur le temps de formation.

Module	Nb de jours	Nb d'heures
Ecrits professionnels	2	14
HTML5 / CSS3	3	21
JavaScript	5	35
Tests unitaires / d'intégration	3	21
Angular2	4	28
Agile / Scrum Master	2	14
Base de données relationnelles + NoSQL	4	28
Agile / Scrum Master suite	1	7
Node.js + Express	4	28
Certification Agile / Scrum (si pas de certification 1 journée d'Agilité en plus)	1	7
WebServices REST	2	14
TP	1	7
HTML5 JavaScript APIs	3	21
JavaScript Libraries	2	14
React	2	14
Sécurité des applications web	3	21
Framework Mobilité (ionic, angular native, react native)	3	21
Projet tutoré	10	70
TOTAL	**57**	**399**

2 PLANS DE COURS ET TP

Rédiger des écrits professionnels efficaces

A l'issue de ce stage les participants seront en mesure de : Utiliser efficacement les principales techniques de communication écrite - Rendre les documents clairs, attractifs pour faciliter leur lecture.

Référence : COME-RED

Durée : 2 jour(s) (14h)

Certification : Aucune

Appréciation : Evaluation qualitative de fin de stage

Modalités et moyens pédagogiques :
- Démonstrations – Cas pratiques – Synthèse et évaluation des acquis

Prérequis : Cette formation ne nécessite pas de prérequis.

Public concerné : Toute personne ayant à communiquer par écrit au sein de son entreprise.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

▪ POUR DES ECRITS EFFICACES

Avant de rédiger
Enrichir ses idées
Déterminer l'objectif de l'écrit
S'adapter au lecteur
Pour bien écrire
Le choix du plan
Paragraphes, introduction et conclusion
Titres, intertitres et chapeaux

▪ ECRIRE POUR ETRE LU ET COMPRIS

Les règles d'or de la lisibilité
Faire court et simple
Eviter le jargon
Eviter la langue de bois bureaucratique
Eviter le franglais
Comment écrire court et clair
Le style écrit fluide
Des écrits vivants et concrets
Savoir attirer l'attention
- Les techniques d'accroche
- La présentation des documents

▪ LES DIFFERENTS TYPES D'ECRITS PROFESSIONNELS

Les documents de l'entreprise et leur fonction
- Lettre
- Message
- Lettre d'envoi
- Note de service
- Compte-rendu
- Procès verbal
- Ecrits commerciaux

Prise de notes et compte-rendu

▪ RAPPELS SUR LES PIEGES DE LA LANGUE FRANCAISE

Les règles de l'orthographe grammaticale, les abréviations
Les verbes introducteurs dans les écrits professionnels
Néologismes et barbarismes
Les mots de liaison

▪ SYNTHESE DE LA SESSION

Réflexion sur les applications concrètes que chacun peut mettre en oeuvre dans son environnement
Conseils personnalisés donnés par l'animateur à chaque participant
Bilan oral et évaluation à chaud

HTML 5 et CSS 3 - Création de pages Web

A l'issue de ce stage les participants seront en mesure de : Comprendre la structuration d'une page HTML5 - Ajouter des styles CSS aux éléments d'une page - Utiliser les blocs et les tableaux - Créer des formulaires avec WebForms 2 - Tester les nouveautés HTML5 et CSS3.

Référence : HTM-FND

Durée : 3 jour(s) (21h)

Certification : Aucune

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Aucun.

Public concerné : Webmasters, concepteurs Web, développeurs, chefs de projets techniques.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

STRUCTURE D'UNE PAGE

Tags principaux
Simplification avec HTML5
Les éléments et leur placement

BALISES STRUCTURANTES

Blocs Div et Span
Listes
Tableaux
iFrames

WEBFORMS2

Champs de saisie
Listes déroulantes
Boutons radio
Champs date
PlaceHolder
Slider
Validation automatique

HTML5

Balises sémantiques
Audio et vidéo
Canvas et SVG
WebGL

STYLES CSS

Syntaxe des sélécteurs CSS
Pseudo sélecteurs
Principales propriétés CSS
Règles CSS
Frameworks CSS

STRUCTURE DES ELEMENTS

Le modèle de boîte
Eléments block et inline
La propriété display
Padding, marges et bordures

POSITIONNEMENT

Positionnement dans le flux
Positionnement absolu et relatif
Positionnement fixé
Positionnement flottant

CSS3

Nouveaux sélecteurs
Fonts, couleurs et bordures
Positionnement en colonnes
Animations
Media Queries

JavaScript

A l'issue de ce stage les participants seront en mesure de : Connaître les bases de JavaScript et du DOM - Juger de l'intérêt de jQuery pour la programmation cross-browser - Gérer les évènements et les manipulations dynamiques - Savoir les règles d'or de la programmation avec JavaScript - Réaliser des appels synchrones (Ajax).

Référence : JVS-IN

Durée : 5 jour(s) (35h)

Certification : Aucune

Appréciation : Evaluation qualitative de fin de stage

Modalités et moyens pédagogiques :
- Exposés
- cas pratiques
- synthèse

Prérequis : Connaissance de XHTML et CSS.

Public concerné : Développeurs, architectes, chefs de projets techniques.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

UN LANGAGE POUR LE WEB

Historique de JavaScript
JavaScript et Ajax
Librairies JavaScript
Avenir de JavaScript

BASES DU LANGAGE

Syntaxe de base
JSON
Pièges du typage dynamique
Programmation objet
Constructeurs et "this"
Fonctions et programmation fonctionnelle
Objet Window ou le contexte global
Contextes d'exécution

DOCUMENT OBJECT MODEL

Les objets du navigateur
Récupérer des éléments
Ajouter des balises
Modifier les contenus
Manipulation des CSS
Gestion des événements
Cross browsing
Amélioration avec jQuery

AJAX

Architecture des sites Web avec et sans Ajax
Asynchronisme dans le navigateur
XMLHttpRequest et HTML5
Gestion d'erreurs et timeout
Ajax avec jQuery

JQUERY

Les librairies jQuery, Dojo, Sencha ...
Fonctions indispensables
Effets visuels
jQuery et le DOM
Chaînage des appels
Fonctions avancées : proxy, merge et extend

Ingénierie logicielle Agile

A l'issue de ce stage les participants seront en mesure de : Appréhender le développement Agile basé sur des techniques modernes : TDD, intégration continue, automatisation des tests.

Référence : AGI-ING

Durée : 3 jour(s) (21h)

Certification : PSD - Professional Scrum Developer

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Avoir suivi les cours : "Méthode Agile - Comprendre la démarche" et "Leadership et Management Agile - Travail en équipe" ou avoir les connaissances équivalentes.

Public concerné : Architectes, chefs de projets et développeurs.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

LE TEST EN AGILE

Qualité et agilité
Approche globale et pilotage par les tests
TDD, BDD, ATDD, automatisation des tests, outils, reproductibilité
Types de tests, démos

LES LIBRAIRIES XUNIT

Déclarer un test
Les assertions
Préparation et nettoyage du test
Organisation des tests

PRINCIPES ET BONNES PRATIQUES DU TDD

Principes du TDD
Objectifs et avantages
Principe du Test First
Émergence du design, à partir des tests

STYLES DE TDD

Tests basés sur l'état ou le comportement
Utilisation de doublures (Mocks, Stubs, Spy, Fakes...)

REFACTORING ET CODE TESTABLE

SOLID
Inversion de dépendances et découplage
Polymorphisme
Nommage
Elimination de la duplication
Code smells et refactoring

TDD ET CODE LEGACY

Test et code intestable
Stratégies de refactoring du code legacy
L'intégration continue
Le pair programming

AngularJS

A l'issue de ce stage les participants seront en mesure de : Connaître les problématiques des Single Page Application (SPA) - Savoir y répondre avec AngularJS - Utiliser et définir des directives - Accéder au serveur depuis Angular - Gérer la navigation entre les vues et l'historique - Lier les composants et les modèles.

Référence : JVS-ANG

Durée : 4 jour(s) (28h)

Certification : Aucune

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Avoir une connaissance pratique de JavaScript et jQuery ou avoir suivi le cours JVS-IN "JavaScript".

Public concerné : Développeurs, architectes, chefs de projets techniques.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

ARCHITECTURE D'UNE SPA

Rôles du client et du serveur
Accès aux données par un service REST
Angular côté client
JSON au milieu
Gestion de l'état applicatif
Synchronisation des données
Navigation dans une application mono-page
Gestion des URL

PRINCIPES DE BASE D'ANGULAR

Model View Whatever : les types de MVC
Le MVC à la mode Angular
Bénéfices du Two Way Data Binding
Injection de dépendances

FIGURES IMPOSEES

Gestion des formulaires
Angular Templates et expressions
Navigation, hashbang et deeplinking
Accès serveur simplifié
Contrôleurs et modèles
Décoration par les CSS
Internationalisation
Routage et gestion de l'historique

PARTICULARITES D'ANGULAR

Processeur HTML
Filtres
Directives
Scopes
Modules
Providers
Services

LES DIRECTIVES EN DETAIL

Directives pré-définies
Directives Custom
Scope et cycle de vie
Accès au DOM, événements et templates
Transclusion

ANGULAR UN FRAMEWORK TESTABLE

Tests unitaires avec Angular
ngMock
End to End testing
Karma Test Runner
Protractor

ANGULAR PARTOUT ?

Angular comparé aux autres frameworks MVC
Modularité et applications multi-vues
Angular UI
Intégration d'Angular avec d'autres librairies
Modularité d'Angular
Les limites d'AngularJS

Méthodes Agiles - Comprendre la démarche

A l'issue de ce stage les participants seront en mesure de : Appréhender les principales méthodes Agile (Scrum, Extreme Programming, Kanban, Lean) - Développer une sensibilisation aux défis d'une transformation Agile.

Référence : AGI-MET

Durée : 2 jour(s) (14h)

Certification : Aucune

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Aucun.

Public concerné : Architectes, chefs de projets, directeurs de projets, développeurs, décideurs, responsables qualité, commerciaux...

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

▪ LES PRINCIPALES METHODES AGILE

Les raisons et origines de l'agilité : valeurs et principes
Filiation et comparaison des principales méthodes agiles : Lean, XP, Scrum, Kanban
Aperçu de Scrum
Les acteurs de Scrum
Développement itératif
Boîtes des temps, Timebox
Communication, interaction
Pratiques d'expression du besoin
Délivrer de la valeur
Les User Stories, Backlog
Personas
Notion de "fini"
Pratiques quotidiennes et pilotage
Visualisation et "radiateurs" d'information
Les burndown / up charts
Les standups
Pratiques de fin d'itération et de cycle
Les revues
Les rétrospectives
Extreme Programming
Les pratiques d'ingénieries
Dette technique
Feedback, partage du code
Tests automatisés, refactoring

▪ KANBAN

Mise en oeuvre de Kanban
Visualiser le flux
Gérer le flux
Limiter le travail
Classes de service

▪ TRANSFORMATION AGILE

Plan de passage à l'agilité
Conduite du changement
Scalabilité
Freins et contraintes
Contractualisation
Leadership et management Agile

Base de données relationnelles et NoSQL

A l'issue de ce stage les participants seront en mesure de : Maîtriser les caractéristiques techniques des bases de données NoSQL et les différentes solutions disponibles - Identifier les critères de choix.

Référence : NOSQL-TH

Durée : 4 jour(s) (28h)

Certification : Aucune

Appréciation : Evaluation qualitative de fin de stage

Modalités et moyens pédagogiques :
- Démonstrations – Cas pratiques – Synthèse et évaluation des acquis

Prérequis : Avoir des connaissances générales en informatique.

Public concerné : Toute personne amenée à travailler sur les bases de données SQL Server.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

▪ INTRODUCTION AUX BASES DE DONNEES

Introduction aux bases de données relationnelles
Autres types de bases de données
Analyse des données
Langage de bases de données
Origine des bases de données
Les notions de transaction
Les SGBD
La standardisation SQL
L'arrivée de nouveaux besoins
- Volumes importants liés aux technologies et aux nouveaux usages
- Traitements optimisés de flux de données au fil de l'eau

Développement des techniques sur différents aspects
- Stockage
- Indexation / recherche
- Calcul

Définition ETL (Extract Transform Load)

▪ MODELISATION DES DONNEES

Modélisation de données
Modèle de bases de données ANSI / SPARC
Modèle "Entité-Association"

▪ NORMALISATION

Pourquoi normaliser les données ?
Conditions de normalisation
Niveaux de normalisation
Dénormalisation

▪ RELATION

Mappage de schéma
Intégrité référentielle

▪ PERFORMANCE

Indexation
Performances des requêtes
Accès concurrentiel

▪ OBJETS DES BASES DE DONNEES

Tables
Vues
Procédures stockées
Autres objets de base de données

▪ CARACTERISTIQUES NOSQL

Structure de données proches des utilisateurs, développeurs
- Sérialisation
- Tables de hachage
- JSON

Priorité au traitement du côté client
Protocoles d'accès aux données
Interfaces depuis les langages classiques
Données structurées et non structurées
- Documents
- Images

Stockage réparti
- Réplication
- Sharding
- Protocole gossip
- Hachage

Parallélisation des traitements
- Implémentation de MapReduce

Cohérence des données et gestion des accès concurrents
- "Eventual consistency"
- MVCC (Multi-Version Concurrency Control)

▪ PRINCIPAUX ACTEURS

Les solutions NoSQL et leurs choix techniques
- CouchDB
- MongoDB
- Cassandra
- HBase (Hadoop)
- ElasticSearch...

Démonstrations avec Cassandra et CouchDB
Critères de choix

▪ MISE EN OEUVRE

Points à vérifier
- Méthode d'utilisation des données

Format de stockage
- JSON
- XML

Choix de la clé
Notion de clé composite
Aspects matériels
Besoins en mémoire
Disques
Répartition
Import des données

- Outils et méthodes selon les moteurs NoSQL

Node.js

A l'issue de ce stage les participants seront en mesure de : Penser et développer asynchrone dans un environnement multi-utilisateurs - Maîtriser les API fondamentales fournies par Node.js - Approfondir NPM et la modularité - Accéder aux données depuis Node.js - Utiliser les modules Express et Socket.IO - Déployer une application Node.js.

Référence : JVS-NOD

Durée : 4 jour(s) (28h)

Certification : Aucune

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Avoir une connaissance avancée de JavaScript ou avoir suivi le cours JVS-AV "JavaScript avancé".

Public concerné : Développeurs, architectes, chefs de projets techniques.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

IMPACTS ARCHITECTURAUX

Présentation de Node.js
Intérêts de JavaScript côté serveur
Asynchronisme
Stateless

ASYNCHRONISME AVEC NODE

Logique asynchrone
Des callbacks partout
Nested callbacks
Gestion d'erreurs
Patterns asynchrones

NODE API

Global objects
Node Event Loop
Event et EventEmitter
Process objects
Processus fils
Timers

MODULES

Inclusion de modules avec require
Résolution des noms, chemins et fichiers
Modules populaires

PROGRAMMATION SERVEUR

Modules http, net et dgram
Modules dns et url
Module fs (file system)
Connect
Ressources statiques
Moteurs de templates
Templates avec EJS
Templates avec Jade et Stylus

BACK END

Modes de connection
Accès MongoDB
Accès Redis
Accès MySQL
ORM pour Node.js

NODE PACKAGE MANAGER

Commandes NPM
Packager un module
Installation locale ou globale

EXPRESS WEB FRAMEWORK

Configuration
API Requêtes et Réponses
API Routes
Orientation Rest
Single et Multiple pages

SOCKET.IO

Web Sockets
Réception et envoi de messages
Socket.IO côté client

DEPLOIEMENT

Test avec mocha
Build avec Grunt
Liens avec Apache
Modes de déploiement
Forever

De Chef de projet à Manager Agile

A l'issue de ce stage les participants seront en mesure de : Appréhender le changement de paradigme du chef de projet au scrummaster, facilitateur - Appréhender le changement de paradigme du manager au leader et du "command & control" au "servant leadership" - Ce stage inclut le passage de la certification.

Référence : AGI-MAN

Durée : 1 jour(s) (7h)

Certification : PSM - Professional Scrum Master

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Avoir suivi les cours : "Méthodes Agile - Comprendre la démarche" et "Leadership et Management Agile - Travail en équipe" - Avoir des aptitudes relationnelles.

Public concerné : Coachs agiles en devenir et scrummasters.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

SAVOIR OPTIMISER LA VALEUR

Approche Lean
Eliminer les gaspillages
Voir le tout
Analyse des causes

SE PROJETER DANS LES NOUVELLES ORGANISATIONS COLLABORATIVES

La spirale dynamique de Clares
Les niveaux de Dilts
Les principes de délégation (Tannenbaum & Schmidt)
Les principes de constitution d'équipe (Tuckman)

EVOLUER DANS UN MONDE COMPLEXE

Guide de prise de décision (simple, compliqué, complexe, chaotique)
Modes de communication, dynamique de groupe
Rôle de scrummaster & coach agile
Posture et positionnement
Accompagnement adapté aux profils et aux capacités

OUTILS D'AUTO-EVALUATION

Préparation à la certification Scrummaster (cursus Scrum.org)

Java - Développer des services Web avec REST

A l'issue de ce stage les participants seront en mesure de : Créer et utiliser des Web services REST en Java dans le cadre d'une application de type SOA ou d'utilisation dans le cadre d'interface graphique ou service.

Référence : JAV-REST

Durée : 2 jour(s) (14h)

Certification : Aucune

Appréciation : Evaluation qualitative de fin de stage

Modalités et moyens pédagogiques :
- Démonstrations – Cas pratiques – Synthèse et évaluation des acquis

Prérequis : Bonnes connaissances en XML, langage et développement Java.

Public concerné : Concepteurs et développeurs amenés à conduire des projets utilisant des Web Services Rest.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

INTRODUCTION A REST EN JAVA

REST et http, même combat

Qu'est-ce qu'une architecture de type RESTful ? Ses principes

CONCEVOIR DES SERVICES RESTFUL

Le modèle objet

Le modèle des URIs

Présentation de JSON et XML

Les méthodes http GET, PUT, POST, DELETE...

PRESENTATION DE JAX-RS

Développer un service RESTful avec JAX-RS

Déployer le service JAX-RS

LES ANNOTATIONS JAX-RS

Utilité des annotations

- @Path
- @PathParam
- @MatrixParam
- @QueryParam
- @FormParam
- @HeaderParam
- @CookieParam

LA GESTION DES EXCEPTIONS

Réponse complexe et exceptions

Gérer les exceptions

DEPLOIEMENT ET INTEGRATION

Déploiement d'une application REST

Configuration des composants REST

Intégration avec EJB

Intégration avec Spring

LES DIFFERENTS TYPES DE CLIENT JAVA

Le client Browser

Le client Java

Apache HttpClient

LES PRINCIPALES IMPLEMENTATIONS DE REST (JAX-RS)

Jersey

Apache CXF

Les API de HTML 5 et CSS 3 - Pour les développeurs

A l'issue de ce stage les participants seront en mesure de : Connaître les API HTML5 et leur utilité - Valider des formulaires avec WebForms 2 - Dessiner avec Canvas et SVG - Gérer la déconnexion - Communiquer avec les WebSockets - Gérer les tâches longues avec les WebWorkers - Réaliser des animations avec CSS3 - Comprendre le responsive design et les liens entre HTML5 et la mobilité.

Référence : HT5-CSS3

Durée : 3 jour(s) (21h)

Certification : Aucune

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Pratique du langage JavaScript ou avoir suivi le cours JVS-IN "JavaScript".

Public concerné : Développeurs, architectes, chefs de projets techniques.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

LE WEB VERSION HTML5

Définition et limites de HTML5
Support des navigateurs
Modernizr et librairies pour la compatibilité
Impact sur les architectures Web
HTML5 pour les mobiles

STRUCTURE DES PAGES HTML5

Simplifications
Doctype
Balises sémantiques
Micro formats

WEBFORMS2

Nouveaux champs de saisie
Sliders, datalist et placeholder
Expressions régulières
Validation automatique
Validation dans le code

MULTIMEDIA ET GRAPHISME

Audi et vidéo
Canvas
SVG
WebGL

COMMUNICATIONS

XHR2
CORS
JSON
Messaging
WebSocket

WEBWORKERS

Modèle mono-thread
Worker API
Synchronisation
Shared Workers

FICHIERS ET RESSOURCES LOCALES

LocalStorage
SessionStorage
ApplicationCache
IndexDB
File API

DEVICE API

Géolocalisation
Orientation
Batterie
Caméra et micro
WebRTC

CSS3

Fonts
Sélecteurs CSS3
Bordures
Couleurs et opacité
Transitions et transformations
Animations

RESPONSIVE DESIGN

Vision OneWeb
Responsive Web Design
Progressive Enhancement
Media Query

JavaScript avancé

A l'issue de ce stage les participants seront en mesure de : Comprendre les contextes d'exécution - Structurer le code JavaScript en modules - Implémenter les concepts objets en JavaScript - Maîtriser les aspects fonctionnels et les "closures".

Référence : JVS-AV

Durée : 2 jour(s) (14h)

Certification : Aucune

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Avoir une connaissance pratique du langage JavaScript ou avoir suivi JVS-IN "JavaScript".

Public concerné : Développeurs, architectes et chefs de projets techniques.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

PIEGES DU LANGAGE

Typage faible, "hoisting"...
Contexte et variables globales

PROGRAMMATION OBJET

Constructeurs
"this"
Prototype et __prototype
Héritage
Visibilité

PROGRAMMATION FONCTIONNELLE

Fonctions anonymes
Fonctions immédiates
Fonctions internes
Redéfinition
Surcharge et mot clé (arguments)
Closure

STRUCTURATION ET QUALITE DU CODE

Séparation en multiple fichiers
Définition de modules
Asynchronous Module Definition (AMD)
AMD avec Require.js
Impacts des "closures" sur la lisibilité
Qualité avec JSHint et JSLint

EVOLUTIONS RECENTES

Présentation des langages dérivés de JavaScript : TypeScript, Dart, CoffeeScript
ECMAScript 6 et ses nouveautés : classes, modules, fonctions =>, promise, nouvelles méthodes de Object...

ReactJS - Maîtriser le framework

A l'issue de ce stage les participants seront en mesure de : Connaître ReactJS et son fonctionnement - Ecrire des composants en ES2015 - Développer une application à l'aide d'un environnement de développement.

Référence : JVS-REA

Durée : 2 jour(s) (14h)

Certification : Aucune

Appréciation : Evaluation qualitative de fin de stage

Modalités et moyens pédagogiques :
- Démonstrations – Cas pratiques – Synthèse et évaluation des acquis

Prérequis : Avoir une très bonne connaissance de JavaScript et de ses aspects avancés.

Public concerné : Développeurs ayant à réaliser des applications Web responsives.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

INTRODUCTION

Rappels sur les aspects avancés de JavaScript
Rappels sur HTML5, CSS3, le DOM
Rappels du pattern MVC
Positionnement de ReactJS
Virtual DOM avec ReactJS
Mise en place des outils de développement
Plug-ins nécessaires aux outils
Synthèse des apports de ES2015
Compatibilité actuelle des browsers
Tour d'horizon des outils de développement et d'intégration actuelle
Compilateurs disponibles

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
Auto-binding
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

APPLICATIONS ISOMORPHIQUES AVEC REACTJS

Faire fonctionner l'application ReactJS aussi bien du côté serveur que client
JavaScript du côté serveur
Présentation de Node.js

REACTJS ET MOBILITE

Construction d'applications natives IOS et Android avec React Native

Sécurité des applications Web

A l'issue de ce stage les participants seront en mesure de : Repérer une attaque ciblée contre une application Web - Comprendre le déroulement d'une attaque - Savoir reconnaître les principales attaques des pirates et leurs cibles - Maîtriser les différents paramètres et les différents composants de sécurité - Mettre en place des mesures de sécurisation simples pour les applications Web Active X et Flash.

Référence : SEC-SAW

Durée : 3 jour(s) (21h)

Certification : Aucune

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Connaissances des réseaux et des principes de la SSI.

Public concerné : Responsables sécurité du SI, chefs de projets informatique, ingénieurs, développeurs, webmasters et administrateurs systèmes et réseaux.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

▪ INTRODUCTION

Histoire et évolution du Web
Les enjeux du Web
Les protocoles du Web
Menaces sur le Web et sécurité
L'OWASP

▪ ATTAQUES CONTRE LE WEB

Panorama du top ten de l'OWASP
Principales attaques par injections
Attaques des navigateurs Web
Attaques sur HTTPs
Le vol de session
Le vol de cookies
Le vol de données

▪ NOUVEAUX TYPES D'ATTAQUES

Attaques inter-domaines
- Utilité des interactions entre domaines
- Causalité des attaques

Codes malveillants sous JavaScript et Ajax
- Codes JavaScript
- Codes Ajax
- Vers célèbres

L'alternative .NET
- Attaques XML
- Attaques SQL
- Attaques XSS

▪ CLIENTS LOURDS

ActiveX
- Présentation
- Failles et contre-mesures
- Protection contre les objets ActiveX

Applications Flash
- Présentation
- Hacking pour Flash
- XSS et XSF

▪ SECURISATION ET BONNES PRATIQUES

Analyse et méthodologie
Sécuriser le code
Sécuriser l'environnement
Sécuriser le navigateur
Audit et pentest
Etude des failles avec Web Coat

Framework Mobilité (ionic, angular native, react native)

A l'issue de ce stage les participants seront en mesure de : Maitriser la Framework et le SDK Ionic, basé sur AngularJS et Cordova. Vous développerez des applications mobiles hybrides pour iPhone et Android proches des applications natives. Outre la mise en oeuvre des fonctionnalités, vous découvrirez les outils de productivité basés sur Node.js.

Référence :

Durée : 3 jour(s) (21h)

Certification : Aucune

Appréciation : Exercices de validation - Attestation de stages

Modalités et moyens pédagogiques :
- Exposés
- Cas pratiques
- Synthèse

Prérequis : Bonnes connaissances des langages HTML, CSS et JavaScript. La maîtrise du Framework AngularJS constitue un atout supplémentaire.

Public concerné : Développeurs Web et chefs de projets mobilité.

Cette formation :

- est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par M2i Formation
- bénéficie d'un suivi de son exécution par une feuille de présence émargée par demi-journée par les stagiaires et le formateur.

PROGRAMME

CONFIGURER UN ENVIRONNEMENT DE DÉVELOPPEMENT MODERNE

Choix de l'éditeur, les Plug-ins indispensables.
Socle des bonnes pratiques : mobile et "Web-platform".
Node.js utilitaire de développement. Synchronisation "multi-device".
Choisir et paramétrer un "workflow" mobile.
Yeoman. Utiliser un "scaffoldeur" de projet.

IONIC SDK : PRÉSENTATION ET MISE EN OEUVRE

Framework de développement "hybride", positionnement.
Composantes : Utilitaires, CSS, JavaScript, Services.
Technologies : services cloud, Cordova, le choix d'AngularJS.
Démarrer, émuler et débugger une application.
Apport du Framework CSS.

IONIC + ANGULARJS : INITIATION AU FRAMEWORK DE GOOGLE

Structure d'une application AngularJS.
Identifier les ressources : Services, Contrôleur, Directives.
Ajax : consommation de services Web.
SPA (Single Page Application) routes et navigation.
Les directives apportées par Ionic.
Utilisation indépendante du Framework CSS Ionic.

LE FRAMEWORK IONIC EN PROFONDEUR

Personnalisation ciblée de la plateforme (IOS/Android).
Gestion du contenu : listes, chargement, "scroll", "pull to refresh".
Gestion de la navigation : menu, route et persistance.
Composant interactifs : "modal, action sheet, popover"
Créer des formulaires efficaces.
Gestuelle utilisateur : "swipe, slide, tap...".

GÉRER LA PERSISTANCE DES DONNÉES UTILISATEUR

Adopter une stratégie "offline first".
Les API pour la sauvegarde locale : IndexDB localStorage.
Centraliser les données : redéfinir le cycle UX.
Quelles fonctionnalités attendre d'un Backend Mobile ?
Les services d'authentification et de "push notification".

PRÉPARER LE "BUILD" ET LE DÉPLOIEMENT

Automatiser la création des icônes et écrans de démarrage.
Présentation des services de la "Ionic Platform".
"Build" service de compilation.
Cycle de déploiement continu.

Exemple de TP pour ce cursus

Les participants doivent mettre en avant les points techniques suivants :

- L'application a été développée avec les technologies suivantes : TypeScript / Angular 2 / Angular Material 2 / NodeJS / MongoDB / Git / Karma / Jasmine / Webpack / Protractor / Sass.

- La séparation du code frontend et backend grâce à des APIs RESTful permet entre autres, de réutiliser le même backend pour le développement d'applications mobiles.

- Le cloisonnement et la séparation de la logique visuelle et de la "business logic" dans le code permet la réutilisation du code Angular pour le développement d'applications mobiles avec les frameworks NativeScript ou Ionic.

- Le développement a été effectué en suivant la méthode Scrum et à l'aide de l'outil Pivotal Tracker.

- L'approche agile et le calcul de vélocité a permis de reprioriser les fonctionnalités afin d'obtenir un MVP (Minimum Viable Product) à la fin du Workshop.

- L'implémentation de l'application en suivant l'approche TDD (Test Driven Development) (tests unitaires / intégration / end-to-end) et l'utilisation d'une plateforme d'intégration continue (CodeShip + Heroku) a permis de réduire au maximum le temps perdu en debug et en administration système. Les tests unitaires couvrent plus de 90% du code.

- Afin de garantir une "scalability" horizontale, le service de stockage de fichiers est prêt à être surchargé pour utiliser une plateforme d'hébergement de fichiers statiques telle que celle fournie par Amazon S3.

3 EXTRAIT DE SUPPORT DE COURS

Chapitre 7
Promesses

1. Introduction

Ce chapitre est dédié aux promesses. Mais avant d'explorer ce concept, il est pertinent de comprendre pourquoi l'utiliser.

S'ensuivra une analyse détaillée d'une promesse : ses états, l'approche semblabe à du code synchrone et la gestion des erreurs. Puis, nous verrons comment créer une promesse de différentes façons, ce qui permettera de bien appréhender le concept. Enfin, nous explorerons l'intégration des promesses dans Node et dans les générateurs.

2. Callbacks vs promesses

L'approche traditionnelle avec Node est d'utiliser des callbacks. Ainsi, on ne bloque pas le serveur en attendant d'exécuter une tâche : celle-ci est lancée, et une fois terminée, elle appelle une fonction.

Cependant, les callbacks souffrent de plusieurs maux. Et le premier est le problème de l'imbrication : effectivement, en cas d'appels successifs à de nombreuses callbacks, le code devient rapidement illisible, et pire, dur à maintenir.

De surcroît, il faut gérer les erreurs à la main : d'ailleurs, si l'on oublie d'en traiter une, on fait face à de gros problèmes (une erreur non traitée est difficile à identifier pendant l'exécution du programme). Ce n'est pas tout : les exceptions donnent aussi quelques sueurs froides. Par exemple, il suffit d'en lancer une au sein d'une callback pour obtenir un système instable car personne ne peut la rattraper (`try {} catch {}`). En outre, Node ne supporte pas les exceptions non gérées. Sans oublier le fait qu'il est impossible de récupérer la pile d'erreurs complète ! Vraiment gênant pour le débogage.

Autre subtilité, il est nécessaire de faire très attention à la manière dont on appelle une callback. Synchrone ou asynchrone, il faut choisir, car si l'on mélange les paradigmes, il est de coutume de dire qu'on libère « Zalgo » (monstre de l'imaginaire populaire, sorte d'abomination qui rend fou). Et ce genre de comportement rend effectivement fou : impossible de prédire le comportement de la callback ! Et au final, de quoi s'arracher les cheveux pour trouver l'origine du problème.

Enfin, pour clore cette longue liste de difficultés, une callback doit par définition être appelée une fois au maximum. Si ce point peut paraître étrange, sachez qu'un appel multiple par erreur peut arriver dans une fonction un peu longue ou complexe, avec par exemple un abus de copier/coller. Toujours est-il que rien ne protège de cette erreur.

Mais heureusement, les promesses sont là pour remédier à ces problèmes. Et c'est exactement pour cela qu'elles bénéficient d'un chapitre complet.

Un exemple est bien entendu beaucoup plus parlant. Voici le code de la section Programmation asynchrone du chapitre Concepts, réécrit en utilisant les promesses et l'API de Bluebird (utilisée tout au long du chapitre).

```
var Bluebird = require('bluebird');

var got = Bluebird.promisify(require('got'));
var writeFile = Bluebird.promisify(require('fs').writeFile);

var files = [
    {
        name: 'foo.html',
        url: 'http://example.org/foo.html',
    },
    {
```

```
        name: 'bar.html',
        url: 'http://example.org/bar.html',
    },
];

Bluebird

    // Télécharge en parallèle tous les fichiers de `files`.
    .map(files, function (file) {
        return got(file.url);
    })

    // Puis les enregistre sur le disque (toujours en parallèle).
    .map(function (result, i) {
        // got() retourne deux résultats que Bluebird.promisify()
        // transforme en tableau.
        var content = result[0];

        return writeFile(files[i].name, content);
    })

    .then(function () {
        console.log('Tout s\'est bien passé');
    })
    .catch(function (error) {
        console.error(error);
    })
;
```

En premier, il ne faut pas oublier d'inclure la bibliothèque `bluebird` via `require()` (il sera expliqué ce qu'est Bluebird plus loin). La différence entre le code du chapitre Concepts et celui-ci commence juste après : les deux fonctions requises (`got` et `writeFile`) sont « promessifiées » (voir la section Création d'une promesse - À partir d'une fonction Node) avec la fonction `promesify()` que vous allez observer tout au long de ce chapitre.

Les deux fichiers sont toujours foo.html et bar.html, dans la variable `files`.

Ensuite, le premier appel à `Bluebird.map()` applique la fonction `got` à tous les éléments du tableau, autrement dit : on télécharge en parallèle tous les fichiers présents dans `files`. L'enregistrement sur le disque se passe aussi via un appel à `map()` qui met les résultats de la lecture des fichiers dans un tableau. Ces résultats sont écrits via `writeFile()`, et tout à la fin, à la dernière étape, au niveau de `then()`, est affiché un message indiquant si tout s'est bien déroulé (ou pas).

La gestion des erreurs se situe à un seul endroit, au niveau de `catch()` ! Comme on peut le constater, l'écriture d'un tel code est fortement simplifiée comparativement à la version callbacks. Et ce n'est que le début : il est temps de comprendre maintenant le concept de promesse.

3. Notion de promesse

La promesse est un paradigme de programmation asynchrone qui résout avec style beaucoup de problèmes que l'on trouve en utilisant le passage de continuations (la convention dans Node).

Et concrètement ? Une promesse est un objet représentant une valeur qui sera disponible dans le futur.

Par exemple, avec `readFile()` qui est asynchrone, le contenu du fichier foo.txt ne sera disponible que plus tard :

```
var promise = readFile('foo.txt');
```

Il est possible d'accéder à la valeur d'une promesse en enregistrant une fonction callback avec la méthode `.then(callback)` qui sera appelée quand cette valeur sera disponible. Voici ce que cela donne pour afficher le résultat de `promise` :

```
promise.then(function (content) {
console.log('le contenu de foo.txt est', content);
});
```

Une promesse peut être rejetée, par exemple si la lecture du fichier a échoué. L'erreur associée à ce rejet peut être récupérée de la même façon qu'est récupérée la valeur, mais avec la méthode `.catch(callback)` :

```
promise.catch(function (error) {
  console.error('la lecture a échoué car', error);
});
```

Les promesses sont tellement pratiques qu'elles seront incluses par défaut dans la prochaine version de JavaScript (ECMAScript 6) avec l'objet `Promise` et qu'elles sont utilisées dans la plupart des nouvelles API de HTML5.

Outre l'implémentation officielle, il existe un grand nombre de bibliothèques implémentant les promesses tout en respectant la spécification standard A+, ce qui leur permet d'être complètement intercompatibles.

Remarque

A+ est un standard open source pour les promesses en JavaScript. Il explicite la terminologie commune et le cahier des charges pour quiconque souhaiterait faire une bibliothèque de promesses. Pour plus de détails sur ce qu'est A+, nous vous invitons à consulter cette URL : https://promisesaplus.com/.

Node ne fournissant pas pour le moment d'implémentation native, il est nécessaire d'utiliser l'une de ces bibliothèques. Historiquement la plus utilisée fut Q (disponible à cette adresse : https://github.com/kriskowal/q), cependant elle est de plus en plus remplacée par Bluebird. Le dépôt officiel et la documentation sont accessibles via cette URL : https://github.com/petkaantonov/bluebird. Non contente d'être l'une des plus performantes (sinon la plus performante), elle propose aussi un nombre impressionnant de méthodes très pratiques permettant de gagner considérablement en efficacité lors du développement avec des promesses.

Dans les exemples de ce chapitre, il est fait référence à `Promise` lorsque seules des méthodes standards et compatibles avec l'implémentation officielle sont utilisées (compatibles avec ES6 et la majorité des implémentations), et à Bluebird lorsque nous utilisons certaines de ses méthodes spécifiques.

3.1 États d'une promesse

Une promesse est dans un état, et un seul. Les états possibles sont :

– en attente (*pending*), quand l'opération asynchrone n'est pas terminée.
– remplie (*fulfilled*), quand l'opération est terminée avec succès.
– rejetée (*rejected*), quand l'opération a échoué.

Une fois que la promesse est remplie ou rejetée, elle ne change plus jamais.

3.2 Similarité avec du code synchrone

Un des objectifs des promesses est de rendre le code asynchrone le plus facile à écrire pour un développeur. L'idée est donc qu'il se rapproche le plus possible du code synchrone.

Voici un code synchrone qui lit un fichier JSON, décode son contenu puis, après l'avoir modifié, le réécrit :

```
try {
    var config = JSON.parse(fs.readFileSync('config.json'));

    config.foo = 'bar';

    fs.writeFile('config.json', JSON.stringify(config));
} catch (error) {
    console.error('Erreur :', error);
}
```

Et voici une version asynchrone basée sur les promesses :

```
var readFile = Bluebird.promisify(fs.readFile);
var writeFile = Bluebird.promisify(fs.writeFile);

readFile('config.json').then(JSON.parse).then(function (config) {
    config.foo = 'bar';

    return writeFile('config.json', JSON.stringify(config));
}).catch(function (error) {
    console.error('Erreur :', error);
});
```