# Evaluation de mi module.

Nom : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Prénom :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

42 questions, noté sur 40 points (2 points bonus).

Veuillez placer une croix devant les réponses choisies.

*Attention :*

* *Pour chaque question, il peut y avoir plusieurs réponses possibles.*
* *La réponse est considérée comme « Bonne » si toutes les bonnes réponses sont cochées*
* *Une croix entre 2 réponses est considérée comme non valide*

A vous de jouer.

1/ Que veut dire MVVM :

* Microsoft Visio Viewer for Macintosh
* Mother Viewing Vibes in Moon
* Model View ViewModel
* Model ViewModel View

2/ Qu’est-ce que le MVC :

* Un pattern d’architecture abstrait pour structurer son code
* Une marque de voiture à la mode
* Une méthodologie ou d'un motif de conception visant à faire le lien entre l'interface utilisateur et les modèles de données sous-jacents

3/ Dans le concept MVVM, La View sert à :

* Analyser et traiter les données venant de l’utilisateur
* Afficher les éléments venant du ViewModel
* Récupérer et transmettre les interactions venant de l’utilisateur au ViewModel
* Effectuer des opérations et des traitements

4/ Angular doit être compilé en JavaScript pour fonctionner dans le navigateur :

* Oui
* Non

5/ Qui effectue la compilation et la génération du Bundle Angular :

* NodeJs
* TypeScript
* WebPack
* Tsc

6/ Pourquoi typer ses variables :

* Pour obtenir un code plus lisible et compréhensible
* Pour faire classe dans les fichiers
* Pour connaitre plus facilement les propriétés à utiliser
* Pour améliorer la qualité générale du code
* Limiter les erreurs

7/ Parmi ces langages, lesquels permettent de typer les variables et fonctions :

* C#
* JavaScript
* TypeScript
* Angular

8/ Un composant Angular, généré par la CLI et sans les tests, se décompose en :

* 2 fichiers
* 3 fichiers
* 4 fichiers
* 5 fichiers

9/ Pour créer un nouveau composant, j’utilise la ligne suivante :

* ng g c <mon\_composant>
* ng generate component <mon\_composant>
* ng g c <mon\_composant> -m <mon\_module>
* ng g m <mon\_composant>

10/ Qui doit déclarer les Component pour permettre son utilisation :

* Le Module
* Le Router
* Le Component lui-même (il s’auto gère)

11/ Un module peut être utilisé en tant que :

* Module de Page (il gère une sous partie de l’application de façon autonome)
* Module de Composants (Il référence et exporte des composants réutilisables)
* Module de Routage (Il référence les routes utiles pour fonctionner)
* Module de Provider (Il référence et expose des Services réutilisables)

12/ A quoi sert la notion de Providers, dans un Module :

* A fournir les informations supplémentaires sur le comportement des Components
* A limiter le scope d’un Service au Module et ses enfants
* A trouver le meilleur Dealer du coin

13/ Le binding sert à :

* Déclencher des actions utilisateur
* Lier des éléments entre la partie TS et la partie HTML
* Affecter la valeur d’une variable

13/ Un attribut HTML entre [ ] est un binding :

* Mono-directionnel entre le HTML vers le TS
* Mono-directionnel entre le TS vers le HTML
* Bi-directionnel

14/ Un attribut HTML entre ( ) est un binding :

* Mono-directionnel entre le HTML vers le TS
* Mono-directionnel entre le TS vers le HTML
* Bi-directionnel

15/ Si un composant parent veut communiquer avec son enfant :

* Le parent doit disposer d’une variable décorée avec @Input
* Le parent doit disposer d’une variable décorée avec @Output
* L’enfant doit disposer d’une variable décorée avec @Input
* L’enfant doit disposer d’une variable décorée avec @Output

16/ Si un composant enfant veut communiquer avec son parent :

* Le parent doit disposer d’une variable décorée avec @Input
* Le parent doit disposer d’une variable décorée avec @Output
* L’enfant doit disposer d’une variable décorée avec @Input
* L’enfant doit disposer d’une variable décorée avec @Output

17/ Pour faire communiquer 2 composants entre eux, nous pouvons utiliser :

* Des décorateurs @Input et @Output
* Des Modules
* Des Services
* Des pigeons voyageurs

18/ Le Router peut servir à :

* Indiquer la route à un passant
* Faire la liaison entre l’Url et les composants à afficher
* Gérer le chargement de l’application au fur et à mesure de sa progression
* Effectuer un appel vers une API

19/ Quelle librairie gère, aujourd’hui par défaut, les Observables, dans Angular :

* NodeJs
* RxJs
* RgRx
* Angular lui même

20/ Tout le monde peut déclencher la notification d’un Observable :

* Vrai
* Faux

21/ L’objet qui s’abonne à un observable se nomme :

* L’Observeur
* L’Observable

22/ Lors de son abonnement, le Subject donnera :

* Les prochaines valeurs
* La valeur actuelle ainsi que les prochaines valeurs

23/ Lors de son abonnement, le BehaviorSubject donnera :

* Les prochaines valeurs
* La valeur actuelle ainsi que les prochaines valeurs

24/ Pour déclencher la notification d’un Subject ou BehaviorSubject, on utilise :

* monBehavior = maValeur ;
* monBehavior.emit(maValeur) ;
* monBehavior.next(maValeur) ;
* monBehavior.subscribe(maValeur) ;

25/ On se désabonne principalement d’un Observable :

* Pour ne pas surcharger l’Observable inutilement
* Pour éviter les fuites mémoires
* Parce que c’est bien de se désabonner

26/ Quelle est la différence principale entre une Promise et un Observable :

* La Promise donne le résultat 1 seule fois, l’Observable autant de fois, tant que l’on reste abonné
* La Promise donne le résultat autant de fois, tant que l’on reste abonné, l’Observable 1 seule fois
* Il n’y a pas de différence majeure entre les 2

27/ Avec quel opérateur pouvons-nous chainer des opérations asynchrones retournant des Observables :

* .map()
* .filter()
* .pipe()
* .switchMap()

28/ La synchronisation de plusieurs actions asynchrones peut se faire avec le ou les opérateurs :

* CombineLatest
* SwitchMap
* ForkJoin
* MergeMap

29/ L’abonnement à un Observable bloque le process d’exécution tant que l’on n’a pas eu de réponse de cet Observable :

* Vrai
* Faux

30/ Dans le cycle de vie de Angular, ngAfterViewInit se déclenche avant ngOnInit :

* Vrai
* Faux

31/ ngOnDestroy est déclenché :

* Lors de la création du composant
* Lors de la mise à jour d’une valeur d’un composant
* Avant la suppression du composant
* Après la suppression du composant

32/ Pour simplifier la gestion des formulaire, Angular propose une classe nommée :

* FormGroup
* FormBuilder
* FormInjector
* FormulaD

33/ Le binding automatique des formulaires doit se faire un objet de type :

* Le FormBuilder
* Le FormGroup
* Le FormFactory
* Le FormIdable

34/ Pour sécuriser un formulaire, on utilise des :

* Interceptors
* Validators
* Providers

35/ Où peut agir un Interceptor :

* Pendant le chargement du Module.
* Avant l’envoi de la requête http
* Pendant l’envoi de la requête http
* Après l’envoi de la requête http
* Pendant le chargement du Component

36/ Où agit le Resolver :

* Avant l’envoi de la requête http
* Avant le chargement du Module.
* Avant le chargement du Component
* Pendant le chargement du Component

37/ On peut bloquer l’accès à un composant grâce à :

* Un Resolver
* Un Formulaire
* Un Guard
* Plusieurs Guards
* Un Interceptor

38/ L’internationalisation oblige, par défaut, la génération de plusieurs bundles :

* Oui
* Non

39/ Un Bundle Angular peut être hébergé sur un serveur et être appelé via n’importe quelle url :

* Oui
* Non

40/ La balise <base href= contient la valeur :

* Du chemin d’accès du répertoire où sera stocké le bundle, sur le serveur d’hébergement
* L’url d’accès absolue de l’application Angular
* L’url d’accès relative de l’application Angular

41/ L’analyseur de Bundle est une librairie :

* Disponible nativement dans Angular
* Disponible nativement dans WebPack
* A installer de façon globale dans npm

42/ L’analyser de Bundle permet :

* De corriger automatiquement les erreurs de code
* D’analyser finement la structure du code de développement
* D’analyser finement la structure du code compilé