Master 1 informatique – Université Claude Bernard Lyon 1 - mercredi 10/06/2026

M1IF 13 – Web avancé, web mobile – Examen

Durée : 1 heure 00 – Documents autorisés (4 pages max) – Ordinateurs, calculatrices, tablettes, téléphones portables, montres connectées, drones… interdits

Toutes les questions ont au moins une bonne et une mauvaise réponse.
Il faut cocher toutes les bonnes propositions d'une question pour avoir un point.
Si vous cochez une mauvaise proposition, vous annulez la question.
Remplissez au stylo noir ou bleu la case de la ou des bonne(s) réponse(s) ; une croix ne suffit pas.
Ne barrez pas une mauvaise réponse, mettez du blanc.
Ne redessinez pas une case que vous avez effacée, laissez blanc.
La première question n'est pas prise en compte dans la notation, mais elle est indispensable pour la notation des autres questions.

Questions:

1. **Un event loop facilite**
   a. ❌ la programmation côté client
   b. ❌ la programmation côté serveur
   c. ✅ la programmation fonctionnelle
   d. ❌ la programmation déclarative
   e. ✅ les entrées-sorties
2. **Dans un projet Node.js, NPM permet de**
   a. ✅ gérer les dépendances
   b. ❌ spécifier l'arborescence des fichiers
   c. ✅ définir un script pour lancer le code du projet
   d. ✅ définir un script pour lancer les tests
   e. ✅ choisir un type de gestion de la modularité
3. **Parmi les propositions suivantes quelles sont celles qui sont des standards pour gérer la modularité en JS ?**
   a. ❌ *Bun*
   b. ✅ *ES2015*
   c. ✅ *UMD*
   d. ❌ *WebMod*
   e. ✅ *CommonJS*
4. **Si la commande `npm install` détecte une vulnérabilité, cela peut signifier**
   a. ❌ que le certificat HTTPS installé sur votre machine n'est pas valide
   b. ❌ que votre code contient des failles connues qu'un attaquant pourrait exploiter
   c. ✅ que l'une des dépendances de votre projet contient des failles connues qu'un attaquant pourrait exploiter
   d. ❌ qu'une correction de bug dans votre code peut supprimer cette vulnérabilité
   e. ❌ rien du tout, les vulnérabilités sont détectées directement par les frameworks de test
5. **Parmi les propositions suivantes, quelle(s) est/sont la(les) fonctionnalité(s) d'un bundler JavaScript ?**
   a. ❌ Compiler le JavaScript en code machine natif pour le navigateur
   b. ✅ Regrouper dans un même *bundle* différents types de fichiers (JS, CSS, JPG...)
   c. ✅ Rassembler les fichiers JavaScript, résoudre leurs dépendances et les fusionner en un ou plusieurs fichiers optimisés
   d. ❌ Remplacer l'utilisation du langage JavaScript par du TypeScript

    e. ✅ Lancer le code JavaScript côté serveur pour tester le client

6. **Parmi les affirmations suivantes concernant les outils de bundling, laquelle/lesquelles est/sont correcte(s) ?**
    a. ✅ *Webpack* est connu pour sa grande configurabilité et son vaste écosystème de loaders, ce qui le rend adapté aux applications complexes
    b. ❌ *Vite* est l'outil le plus rapide, mais est de moins en moins utilisé car il ne supporte pas le langage TypeScript
    c. ✅ *Parcel* se distingue par son approche "zéro configuration" par défaut, permettant un démarrage très rapide
    d. ✅ *Rollup* est souvent préféré pour la publication de bibliothèques JavaScript grâce à son efficacité pour produire des bundles compacts et supportant différents formats
    e. ❌ *Esbuild* est l'outil le plus simple et le plus utilisé actuellement, car il est proposé avec la majorité des grands frameworks
7. **Parmi les propositions suivantes, quel(s) concept(s) est/sont directement lié(s) à l'utilisation des bundlers ?**
    a. ✅ Le *code Splitting*, qui permet de diviser le bundle principal en plusieurs fichiers chargés à la demande
    b. ✅ Le *Hot Module Replacement* (HMR), qui permet de mettre à jour les modules dans le navigateur sans recharger toute la page
    c. ❌ La compilation du code en *Web Assembly* (WASM) de manière automatique pour tous les fichiers JS
    d. ✅ Le *tree shaking*, qui élimine le code mort (non utilisé) du bundle final pour réduire sa taille
    e. ❌ L'automatisation des tests unitaires sans configuration supplémentaire
8. **Parmi les affirmations suivantes, laquelle/lesquelles décri(ven)t correctement le langage TypeScript ?**
    a. ❌ TypeScript est un langage exécuté directement par les navigateurs modernes sans aucune étape de compilation
    b. ✅ TypeScript est un superset de JavaScript, ce qui signifie que tout code JavaScript valide est également du TypeScript valide
    c. ❌ Le compilateur TypeScript (`tsc`) peut transformer le code directement en WebAssembly pour des performances optimales côté serveur
    d. ❌ Les annotations de types (par exemple `let x: number`) sont conservées dans le code JavaScript généré pour permettre le contrôle de type à l'exécution
    e. ❌ TypeScript ne permet d'utiliser que les fonctionnalités JavaScript les plus anciennes pour assurer une compatibilité maximale
9. **Concernant le langage TypeScript, quelles propositions sont exactes ?**
    a. ❌ L'annotation de type est obligatoire pour chaque variable, sinon le compilateur lève une erreur immédiate
    b. ✅ Les interfaces permettent de définir la structure d'un objet, vérifiant que les propriétés et leurs types sont respectés à la compilation
    c. ✅ Le type `any` désactive les vérifications de type pour une variable, ce qui est utile pour intégrer du code non typé mais à utiliser avec précaution
    d. ✅ Les décorateurs permettent d'ajouter des métadonnées ou modifier le comportement des classes et de leurs membres (souvent utilisé avec des frameworks comme *Angular*)
    e. ❌ En l'absence d'annotation explicite, TypeScript impose l'immutabilité stricte par défaut de toutes les propriétés d'objet
10. **Quel est le principe fondamental d'un *Mashup* tel qu'il a été présenté en cours ?**
    a. ❌ Il s'agit d'une application web qui peut fonctionner en local sans connexion Internet
    b. ✅ Un *mashup* combine des données, des services ou des fonctionnalités provenant de plusieurs sources externes (APIs) pour créer un nouveau service ou une nouvelle application unifiée
    c. ❌ C'est une application web conçue avec un *bundler* pour remplacer le serveur web par des fichiers HTML et JS statiques contenant tout le code nécessaire

d. ❌ Un *mashup* est une application mobile native utilisant les APIs de l'appareil client (GPS, caméra) et qu'on peut installer sur cet appareil

e. ❌ Il s'agit d'un outil qui "pré-mâche" (pré-compile) le code JavaScript en bytecode pour accélérer le transfert au client

11. **Quels sont les principaux défis de sécurité et les mécanismes de protection à considérer lors de la création d'un *mashup* ?**
    a. ✅ La *Same-Origin Policy* empêche par défaut un script d'un domaine d'accéder aux données d'un autre domaine pour protéger l'utilisateur
    b. ✅ L'attaque *Cross-Site Request Forgery* (CSRF) est un risque majeur lié au fait qu'un mashup peut utiliser la même authentification sur plusieurs APIs
    c. ❌ Le navigateur n'exécute que le code du serveur et refuse le code ne respectant pas la *Same-Origin Policy*, ce qui rend les APIs invulnérables aux attaques
    d. ✅ L'utilisation de CORS permet à des serveurs tiers d'autoriser sélectivement certains domaines à accéder à ses ressources
    e. ❌ La gestion des clés API et *tokens* JWT est nativement intégrée aux navigateurs, pour simplifier les vérifications de sécurité côté client
12. **Comment fonctionne le mécanisme CORS pour permettre l'accès aux ressources cross-domaines ?**
    a. ✅ Dans certains cas, le client doit envoyer une requête préliminaire (*preflight*) avant d'envoyer la requête réelle ; pour cela, il utilise la méthode `OPTIONS` pour envoyer les métadonnées de la requête
    b. ❌ La configuration CORS se fait principalement du côté du client, en modifiant en JavaScript les en-têtes de la requête
    c. ✅ Le serveur distant doit envoyer des en-têtes de réponse spécifiques (comme `Access-Control-Allow-Origin`) pour autoriser le navigateur à exposer les données au script
    d. ❌ CORS fonctionne uniquement avec la méthode HTTP `GET`, contrairement à JSONP qui supporte `POST`
    e. ❌ Une requête *cross-origin* utilisant la méthode `POST` et dont le contenu est un document JSON peut être envoyée en CORS simple (sans *preflight*)
13. **Quelle(s) affirmation(s) décri(ven)t le mieux l'objectif principal et le fonctionnement des frameworks JavaScript modernes ?**
    a. ❌ Ils remplacent totalement le langage JavaScript en fournissant un environnement d'exécution natif pour les scripts
    b. ✅ Ils visent à simplifier la gestion de l'état et du rendu de l'application en utilisant une architecture déclarative et réactive
    c. ❌ Ils sont conçus principalement pour exécuter du code côté serveur et ne gèrent pas l'interface utilisateur
    d. ❌ Ils supposent l'utilisation exclusive de la programmation impérative pour manipuler les éléments du DOM
    e. ✅ Ils s'appuient sur la notion de composant afin de gérer l'application de façon modulaire
14. **Concernant l'architecture réactive de type "flux de données" implémentée dans les *stores* des frameworks (*Vue.js*, *Vuex*, *Pinia*) :**
    a. ✅ Le modèle de données (*State*) est la source de vérité unique ; la Vue (*View*) est une fonction déclarative de cet état
    b. ❌ Les changements d'état se propagent directement de manière bidirectionnelle grâce à des événements entre la vue et le modèle
    c. ✅ La notion de *store* est destinée à centraliser la gestion des états et à faciliter le débogage et les tests
    d. ✅ Le DOM Virtuel (*virtual DOM*) calcule efficacement les différences entre l'état précédent et l'état actuel pour minimiser les mises à jour du DOM réel
    e. ❌ La réactivité implique de remplacer complètement le HTML par du code JavaScript capable de re-générer le DOM à chaque interaction

15. **Concernant l'évolution des *stores* dans l'écosystème *Vue.js*, quelle(s) affirmation(s) est/sont correcte(s) ?**
    a. ✅ *Pinia* remplace *Vuex* comme solution recommandée pour Vue 3, offrant une API plus simple
    b. ✅ *Vuex* nécessite l'utilisation de *mutations* obligatoires pour modifier l'état, tandis que *Pinia* permet de modifier l'état directement dans les *actions*
    c. ❌ *Vuex* est la seule solution de gestion d'état compatible avec Vue 3 et les projets utilisant la *Composition API*
    d. ❌ *Pinia* est conçu pour fonctionner uniquement avec la *Options API* et non avec la *Composition API*
    e. ✅ Les *actions* permettent de modifier l'état de l'application, déclenchant une mise à jour automatique de la vue
16. **Quelle(s) affirmation(s) décri(ven)t correctement l'approche *Mobile First* ?**
    a. ❌ *Mobile First* consiste à adapter la version bureau (*desktop*) d'un site pour mobile via des *media queries* inversées
    b. ✅ *Mobile First* recommande de concevoir et coder des applications en respectant les contraintes du mobile avant d'enrichir l'expérience pour les autres types de clients
    c. ❌ *Mobile First* s'appuie sur l'approche *Responsive Design* pour garantir que la mise en page reste identique sur tous les appareils, en réduisant le modèle de grille en fonction de la taille de l'écran
    d. ❌ *Mobile First* implique de charger toutes les ressources dès le début, quel que soit l'appareil, pour simplifier le code CSS
    e. ❌ *Mobile First* peut recommander de créer une version distincte du site (URL différente) pour mobile, et d'aiguiller le client en détectant le header `User Agent`
17. **Concernant la gestion des interactions tactiles via les *Touch Events* en JavaScript, quelle(s) proposition(s) est/sont exacte(s) ?**
    a. ✅ L'événement `touchstart` se déclenche lorsqu'un ou plusieurs doigts entrent en contact avec l'écran
    b. ✅ Les événements tactiles fournissent un tableau `touches` qui liste tous les points de contact actifs, contrairement aux événements souris qui ne gèrent qu'un seul point
    c. ❌ Pour gérer un "tap" (tapotement), il suffit d'écouter l'événement `touchend` sans vérifier la durée ou le déplacement du doigt
    d. ✅ La propriété `e.touches.clientX` permet de récupérer les coordonnées X du premier point de contact dans l'objet événement `e`
    e. ❌ Les interfaces `TouchEvent` et `MouseEvent` héritent de `PointerEvent`, qui définit des événements plus génériques
18. **Parmi les propositions suivantes, quelle(s) fonctionnalité(s) est/sont accessible(s) via des API des navigateurs implémentant des spécifications du W3C ?**
    a. ✅ L'API de géolocalisation `navigator.geolocation` permet d'obtenir la position GPS précise de l'appareil, mais nécessite l'autorisation explicite de l'utilisateur
    b. ✅ L'API `DeviceOrientation` fournit les données de l'accéléromètre et du gyroscope, permettant de détecter l'orientation de l'appareil dans l'espace (*alpha*, *beta*, *gamma*)
    c. ❌ L'API de géolocalisation `navigator.geolocation` permet d'obtenir l'adresse approximative à laquelle se trouve l'appareil, sans nécessiter d'autorisation explicite de l'utilisateur
    d. ✅ La *Vibration API* permet de commander le moteur de vibration du téléphone via `navigator.vibrate()`, sous réserve de support du navigateur
    e. ❌ L'API `ExtTemperature` s'appuie sur l'utilisation de capteurs externes connectés à l'appareil, si ceux-ci sont disponibles
19. **Concernant les techniques de rendu côté serveur (SSR) et de génération de site statique (SSG) pour améliorer les performances, quelle(s) affirmation(s) est/sont correcte(s) ?**
    a. ✅ Le SSR génère le HTML complet de la page côté serveur à la demande de chaque utilisateur, ce qui améliore le *First Contentful Paint* (FCP), mais augmente la charge CPU du serveur

b. ✅ Le SSG pré-génère les pages HTML au moment de la construction (*build*) du projet, permettant un chargement quasi-instantané et une mise en cache facile via un CDN

c. ✅ L'*hydratation* est le processus par lequel le navigateur prend le HTML généré par SSR/SSG et "attache" les écouteurs d'événements JavaScript pour rendre la page interactive, ce qui peut parfois causer un délai avant l'interactivité (*Time to Interactive*) si le *bundle* JS est lourd

d. ❌ L'hydratation est une étape nécessaire dans les applications SSG, mais facultative dans les applications SSR : si elle échoue (en SSR), le site reste statique mais parfaitement fonctionnel pour toutes les interactions utilisateur

e. ❌ Le SSG est la seule méthode permettant d'éviter totalement l'envoi de JavaScript au navigateur, contrairement au SSR qui nécessite toujours un *bundle* JS complet pour la réactivité

20. **Parmi les propositions suivantes, laquelle/lesquelles décri(ven)t correctement les différentes API de communication inter-contextes (fenêtres, *iframes*, *workers*) ?**

a. ✅ `postMessage` est une méthode générique qui permet d'envoyer un message à n'importe quelle fenêtre ou *worker* accessible, nécessitant une cible explicite

b. ❌ Un `MessageChannel` est conçu pour la communication un-à-plusieurs (*one-to-many*) depuis un *browsing context* source, pour permettre le broadcast vers plusieurs onglets

c. ✅ L'interface `MessagePort` permet de standardiser chaque extrêmité d'un canal de communication utilisé dans la spécification `postMessage`

d. ✅ `postMessage` permet le transfert d'objets plutôt que leur recopie

e. ✅ Un `MessageChannel` crée un lien bidirectionnel privé entre deux ports (`MessagePort`) qui ne peut être utilisé que par les deux extrémités connectées, offrant ainsi un canal sécurisé et isolé

21. **Parmi les affirmations suivantes concernant les différents types de *Web Workers* (*Dedicated Workers*, *Shared Workers*, *Service Workers*), laquelle/lesquelles est/sont correcte(s) ?**

a. ✅ Un *Dedicated Worker* s'exécute dans un thread séparé du thread principal, ne peut communiquer avec le DOM que via `postMessage`, et est idéal pour les tâches CPU intensives sans bloquer l'interface utilisateur

b. ✅ Un *Shared Worker* peut être partagé par plusieurs documents (onglets ou iframes) de la même origine, permettant une communication centralisée entre plusieurs *browsing contexts*, contrairement au *Dedicated Worker* qui est lié à un seul document

c. ❌ Le *Service Worker* est le seul type de *Web Worker* qui s'exécute dans le thread principal du navigateur, car il ne sert pas à exécuter du code applicatif

d. ✅ Un "Service Worker" agit comme un proxy réseau, interceptant les requêtes de la page pour mettre en cache les ressources et permettre le fonctionnement hors ligne

e. ❌ Pour optimiser les performances, il est recommandé de ne faire exécuter aux *Dedicated Workers* que des tâches simples et unitaires, et de les réinstancier à chaque nouvel appel plutôt que de les laisser inactifs

22. **Quelle(s) affirmation(s) décri(ven)t correctement le fonctionnement des stratégies de cache présentées dans le cours ?**

a. ✅ Dans la stratégie *Cache First*, si la ressource est trouvée dans le cache du navigateur, elle est retournée immédiatement ; si elle est absente, une requête réseau est effectuée et la réponse est renvoyer à l'application puis mise en cache pour les futures requêtes

b. ✅ La stratégie *Network First* (Réseau d'abord) commence par tenter de récupérer la ressource via le réseau ; si la requête échoue et si une version en cache est disponible, la version du cache est retournée comme fallback pour garantir l'accès aux données

c. ❌ La stratégie *Cache First* est préférable pour les données métier dynamiques (par exemple provenant d'appels externes à des APIs), car elle garantit toujours au moins une version disponible et ne bloque pas l'application

d. ✅ Après une requête, il est recommandé d'utiliser `response.clone()` avant de mettre la réponse en cache, car l'objet `Response` ne peut être lu qu'une seule fois

e. ❌ La stratégie *Cache First* ne permet pas de mettre à jour le cache une fois qu'une ressource y est stockée, c'est pourquoi il est préférable d'utiliser l'API *Web Storage* depuis le *service worker* pour stocker les mises à jour des fichiers statiques

23. **Parmi les affirmations suivantes, laquelle/lesquelles décri(ven)t correctement les prérequis techniques et l'architecture d'une *Progressive Web App* (PWA) ?**
    a. ✅ Pour qu'une application soit considérée comme une PWA, elle doit impérativement être servie via un protocole sécurisé (HTTPS ou localhost)
    b. ✅ L'ajout d'un fichier manifest.json est optionnel pour le fonctionnement de base d'une PWA, mais il est nécessaire pour permettre l'installation de l'application sur l'écran d'accueil du mobile et définir son apparence (icône, nom, thème)
    c. ❌ Un Service Worker est indispensable au fonctionnement d'une PWA, pour permettre d'afficher une interface offline
    d. ❌ Le concept de progressivité signifie entre autres que pour les APIs des capteurs et des actionneurs, une PWA doit utiliser les APIs natives des OS et non celles implémentées par les navigateurs
    e. ❌ Contrairement aux applications web classiques, une PWA ne nécessite pas de fichier HTML racine, car le Service Worker génère dynamiquement l'interface à partir des API du navigateur
24. **Concernant ces technologies présentées dans les exposés, lesquelles peuvent fonctionner ensemble (sans conflit car elles n'adressent pas les mêmes "couches" applicatives) ?**
    a. ❌ *Svelte* et *Angular*
    b. ✅ *Next* et *React*
    c. ✅ *Nest* et *D3*
    d. ✅ *Selenium* et *Storybook*
    e. ❌ *A-Frame* et *Pixi*