Web-Client, stats sur les Annales

VAN DE MERGHEL Robin

Semestre 4

Table des matières

Notes : Il ne faut pas se fier non plus à la fréquence ! Je noterai certaines questions qui me paraissent moins probables. À chaque question j'ai noté le nombre d'apparition dans les derniers Annales.

1 Très Fréquentes (globalement)

1.1 7) Donnez-la valeur de x et expliquez

1.1.0.1 Explication Globalement, on a une fonction f qui prend en paramètre une valeur de x, et qui renvoie une fonction g qui prend en paramètre une valeur de y.

Par exemple :

```
(function(x) {
    return function(y) {
        return x + y;
    }
```

```
})(1)(2)
```

Dans ce cas, on applique à la fonction principale la première valeur de la parenthèse, qui est 1. On obtient donc une fonction g, où on remplace x par 1 : `function(y) { return 1 + y; }`. On applique ensuite cette fonction à la valeur 2, et on obtient 3.

1.2 6) En le design des URL est important ? Qu'est-ce qu'une bonne URL ?

1.2.0.1 Explication

On a deux variantes de cette question, mais globalement :

Une URL est destinée au serveur, et non aux utilisateurs. On peut le voir dans les URL de youtube qui contiennent un Hash (une chaîne de caractères aléatoire). Cela permet de pouvoir identifier de manière unique une ressource, et de pouvoir la retrouver facilement.

Il faut donc éviter (toujours dans le cas de youtube) d'utiliser le nom dans les URL, car d'avoir deux vidéos avec le même nom, car cela peut poser problème sinon.

1.3 5) Donner la syntaxe générale d'une règle CSS.

1.3.0.1 Explication

La syntaxe générale d'une règle CSS est la suivante :

```
selector {
    property: value;
}
```

1.4 5) Donner et expliquer le résultat des expressions suivantes :

1.4.0.1 Explication

Un classique est de comparer `==` et `===`. En effet, `==` compare les valeurs, alors que `===` compare les valeurs et les types.

Par exemple, `1 == "1"` est vrai, mais `1 === "1"` est faux.

1.5 4) Qu'est-ce qu'un design de page flexible (ou fluide) ? Pourquoi est-ce important ? Comment le mettre en oeuvre ?

1.5.0.1 Explication

Un design de page flexible (ou fluide) c'est un design de page qui s'adapte à la taille de l'écran. C'est important car les utilisateurs peuvent avoir des écrans de différentes tailles, et on veut que le site soit lisible sur tous les écrans.

En effet, sur un écran de 1920x1080, on peut afficher 1920 pixels de largeur. Mais sur un écran de 1280x720, on ne peut afficher que 1280 pixels de largeur. Si on a un design de page fixe, on ne pourra pas afficher tout le contenu sur un écran de 1280x720.

1.6 4) Qu'appelle-t-on du HTML « sémantique » ? Quels en sont les intérêts ?

1.6.0.1 Explication

La sémantique HTML c'est l'utilisation des balises HTML pour définir le sens des éléments du document. Ex : on utilise la balise `<h1>` pour définir un titre, et la balise `<p>` pour définir un paragraphe.

On ne veut pas définir la forme d'un élément, mais son sens. Ex : on ne veut pas définir un titre avec la balise `<p>`, mais avec la balise `<h1>`.

Ça servira ensuite aux agents utilisateurs (navigateurs, moteurs de recherche, …) pour comprendre le sens du document.

1.7 3) Quel est le principe du DNS ? Quelle en est l'utilité ?

1.7.0.1 Explication Le principe du DNS est de convertir une adresse IP en un nom de domaine. L'utilité du DNS est de pouvoir utiliser des noms de domaine plus facilement que des adresses IP. En effet, on peut se souvenir plus facilement d'un nom de domaine (par exemple `google.com`) que d'une adresse IP (par exemple `172.217.171.196`).

1.8 3) Quel est le principe des web fonts ?

1.8.0.1 Explication Le principe des web fonts c'est de pouvoir utiliser des polices de caractères qui ne sont pas installées sur l'ordinateur de l'utilisateur. On peut utiliser des polices de caractères payantes ou gratuites.

Cela permet de pouvoir vraiment choisir la police de caractères qu'on veut utiliser, et de pouvoir utiliser des polices de caractères qui ne sont pas installées sur l'ordinateur de l'utilisateur.

1.9 3) Qu'est-ce que le HTML ?

1.9.0.1 Explication Le HTML ou **HyperText Markup Language** est un langage de balisage qui permet de définir la structure d'un document. Il utilise des balises pour définir les éléments du document (ex : titre, paragraphe, image, …).

1.10 3) Quelle est la particularité du système objet de javascript ? Comment fonctionne-t-il ?)

1.10.0.1 Explication

Note : je n'ai aucune idée pour cette question, j'attends le cours / que je vois par moi-même.

1.11 3) Comment fonctionnent les formulaires HTML, du points de vue de l'intéraction avec le serveur ?

1.11.0.1 Explication

Note : je ne sais pas quoi dire d'autre

Le serveur reçoit les données du formulaire quand l'utilisateur clique sur le bouton `submit`. Le serveur peut ensuite traiter les données du formulaire. Selon la nature de la requête, le serveur peut renvoyer une page HTML, une image, un fichier, …

1.12 3) Expliquer la propagration des évènements dans le navigateur.

1.12.0.1 Explication

Note : je ne sais pas quoi dire d'autre

La propagation des évènements dans le navigateur c'est le fait que les évènements se propagent dans le DOM. Par exemple, si on a un bouton dans un formulaire, et qu'on ajoute un évènement sur le bouton, l'évènement se propagera aussi au formulaire.

1.13 3) Qu'implique le fait pour HTTP d'être un protocole sans état ? En quoi est-ce souhaitable ?*

Note : Question plutôt orientée serveur, pas sûr qu'on l'ait.

1.13.0.1 Explication Le fait pour HTTP d'être un protocole sans état implique que le serveur ne garde pas en mémoire les informations de la requête précédente. Cela permet de pouvoir traiter plusieurs requêtes en même temps, et de pouvoir traiter plusieurs requêtes différentes en même temps.

1.14 3) Créer une fonction javascript `zip` prenant deux listes en paramètre et retournant une liste de couples. Par exemple `zip([1,2,3], ["a", "b", "c"])` doit retourner `[[1, "a"], [2, "b"], [3, "c"]]`.

Note : On ne doit pas avoir de programmation normalement.

```
function zip(list1, list2) {
    let result = [];
    for (let i = 0; i < list1.length; i++) {
        result.push([list1[i], list2[i]]);
    }
    return result;
}
```

1.15 3) Que signifie être valide pour du HTML ? Pourquoi est-ce important ?

1.15.0.1 Explication Un document HTML est valide s'il respecte la syntaxe du langage HTML : il n'y a pas de balises mal fermées, les balises sont bien imbriquées, les balises existent, …

C'est important car les agents utilisateurs (navigateurs, moteurs de recherche, …) peuvent utiliser ces informations pour comprendre le sens du document (exemple : les malvoyants utilisent des lecteurs d'écran qui lisent le document HTML pour les aider à comprendre le document).

1.16 3) Que vaut `this` en Javascript ?

1.16.0.1 Explication `this` est le contexte d'exécution. Par exemple, si on a une fonction `myFunction()`, et qu'on appelle `myFunction()`, alors `this` vaudra `window`. Si on a une fonction `myFunction()` dans un objet `myObject`, et qu'on appelle `myObject.myFunction()`, alors `this` vaudra `myObject`.

1.17 3) Qu'est-ce que le web ? Quels en sont les principes ?

1.17.0.1 Explication Le Web c'est l'ensemble des ressources qui sont accessibles et distribuées sur internet (sites, images, documents, …).

Ses principes sont les suivants :

- **évolutif** : le web est un système qui évolue constamment, il est donc nécessaire de pouvoir le mettre à jour facilement
- **universel** : le web est accessible à tous, il n'y a pas de restriction d'accès

1.18 3) Qu'indique un DOCTYPE d'un fichier HTML ? Pourquoi est-ce important et comment est-ce utilisé ?

1.18.0.1 Explication Un DOCTYPE d'un fichier HTML indique le type de document. C'est important car cela permet au navigateur de savoir comment interpréter le document. Cela permet aussi

de savoir si le document est bien formé.

On peut ainsi préciser la version du langage de balisage utilisé, et le navigateur va interpréter le document en fonction de la version du langage de balisage utilisé.

On peut l'utiliser en écrivant `<!DOCTYPE html>` en haut du fichier HTML.

1.19 3) Donner les différentes manières de spécifier une couleur en CSS.

1.19.0.1 Explication On peut spécifier une couleur en CSS de plusieurs manières :

- En utilisant le nom de la couleur (ex : `red`, `blue`, `green`, …)
- En utilisant le code hexadécimal de la couleur (ex : `#ff0000`, `#0000ff`, `#00ff00`, …)
- En utilisant le code RGB de la couleur (ex : `rgb(255, 0, 0)`, `rgb(0, 0, 255)`, `rgb(0, 255, 0)`, …)
- En utilisant le code HSL de la couleur (ex : `hsl(0, 100%, 50%)`, `hsl(240, 100%, 50%)`, `hsl(120, 100%, 50%)`, …)
- En utilisant le code CMYK de la couleur (ex : `cmyk(0, 100%, 100%, 0)`, `cmyk(100%, 0, 100%, 0)`, `cmyk(100%, 100%, 0, 0)`, …)
- En utilisant une couleur transparente (ex : `transparent`, `rgba(0, 0, 0, 0)`, `hsla(0, 0%, 0%, 0)`, …), on rajoute un canal alpha
- En utilisant une couleur dégradée (ex : `linear-gradient(90deg, red, blue)`, `radial-gradient(red, blue)`, …)

1.20 3) Expliquer le système de transtypage en JavaScript.

1.20.0.1 Explication Peut-être formulé d'une autre manière, mais en gros :

En JavaScript, les types sont dynamiques. Cela signifie que les variables peuvent changer de type. Par exemple :

```
let a = 1; // a est un nombre
a = "1"; // a est une chaîne de caractères
```

2 Plutôt Fréquentes (globalement)

2.1 2) Quelle est la valeur de `.2 + .4 == .6` ? Pourquoi ?

2.1.0.1 Explication La valeur de `.2 + .4 == .6` est `false`. C'est parce que les nombres flottants ne sont pas représentés de manière exacte en mémoire. Par exemple, `.2` est représenté par `0.19999999999999998` en mémoire. C'est pour cela que `.2 + .4` n'est pas égal à `.6`.

2.2 2) Quels sont les buts recherchés dans l'architecture du Web ?

2.2.0.1 Explication Les buts recherchés dans l'architecture du Web sont :

- La portabilité : le Web doit être accessible depuis n'importe quel appareil (ex : ordinateur, téléphone, tablette, …)
- La simplicité : le Web doit être simple à utiliser (ex : pas besoin de télécharger une application pour utiliser un service)
- La performance : le Web doit être performant (ex : pas de latence, pas de temps de chargement long, …)
- La fiabilité : le Web doit être fiable (ex : pas de pannes, pas de défaillance, …)

2.3 2) Qu'est-ce que le CSS ? Quel est l'intérêt de son utilisation ?

2.3.0.1 Explication Le CSS ou **Cascading Style Sheets** est un langage de style qui permet de définir l'apparence d'un document HTML. Il utilise des sélecteurs pour cibler les éléments HTML, et leur appliquer des styles (ex : couleur, taille, …).

L'intérêt de son utilisation est de pouvoir définir l'apparence d'un document HTML sans avoir à modifier le document lui-même. On peut garder la même structure HTML (en définissant des mots importants, des titres, etc…) et changer l'apparence du document en modifiant le CSS.

2.4 2) Expliquer le principe de la cascade des règles CSS.

2.4.0.1 Explication Le principe de la cascade, c'est que si un élément est ciblé par plusieurs règles CSS, la règle qui s'applique est celle qui a le plus de poids. Dans le cas où plusieurs règles ont le même poids, la règle qui s'applique est la dernière définie.

2.5 2) Que signifie pour un document d'être bien formé ?

2.5.0.1 Explication Un document est bien formé s'il respecte la syntaxe du langage de balisage. C'est-à-dire que les balises sont bien fermées, que les attributs sont bien écrits, que les balises sont bien imbriquées, …

2.6 2) Quel est l'intérêt de la méthode `pushState` de l'historique ?

Note : Assez poussé comme savoir, pas sûr qu'on l'ait.

2.6.0.1 Explication La méthode `pushState` de l'historique permet de modifier l'URL sans recharger la page. C'est utile pour les applications web qui utilisent le système de routage. Par exemple, si on a une application web qui utilise le système de routage, et qu'on veut afficher une page qui correspond à l'URL `/users/1`, alors on peut utiliser la méthode `pushState` pour modifier l'URL sans recharger la page.

2.7 2) Qu'appelle-t-on une closure ? Dans quel cas est-ce utile ?

Note : Assez poussé comme savoir, pas sûr qu'on l'ait.

2.7.0.1 Explication Une closure est une fonction qui peut accéder aux variables définies dans la portée de la fonction qui l'a créée. C'est utile pour créer des fonctions qui peuvent accéder à des variables qui ne sont pas définies dans la fonction.

Par exemple, on peut créer une fonction qui incrémente une variable :

```
function createIncrementer() {
    let counter = 0;
    return function() {
        counter++;
        return counter;
    }
}

let incrementer = createIncrementer();
incrementer(); // 1
incrementer(); // 2
incrementer(); // 3
```

2.8 2) Comment faire de l'héritage en JavaScript ?

2.8.0.1 Explication En JavaScript, on peut faire de l'héritage avec le mot-clé `extends`. Par exemple :

```
class Person {
    constructor(name) {
        this.name = name;
    }
}

class Student extends Person {
    constructor(name, school) {
        super(name);
        this.school = school;
    }
}
```

2.9 2) Que sont les media queries en CSS ? Quels en sont les intérêts ?

2.9.0.1 Explication Les media queries sont des règles CSS qui permettent de définir des styles en fonction du type d'appareil qui affiche le document. Ex : si on affiche le document sur un ordinateur, on peut appliquer un style différent de celui qui s'applique sur un téléphone.

On peut donc définir des règles pour dire "Si l'appareil a une taille d'écran inférieure à 500px, alors applique ce style".

```
@media screen and (max-width: 500px) { /* Si l'écran a une taille inférieure à 500px */
    /* style */
}
```

2.10 2) La curryfication consiste à transformer une fonction à plusieurs paramètres en une série de fonction à un seul paramètre. Par exemple, la première fonction n'est pas currifiée, mais la seconde oui [...]

Note : On ne doit pas coder normalement, donc on aurait pas ça.

```
function curry(f) {
    return function curried(...args) {
        if (args.length >= f.length) {
            return f.apply(this, args);
        } else {
            return function(...args2) {
                return curried.apply(this, args.concat(args2));
            }
        }
    };
}
```

2.11 2) Quels sont les différents modes de positionnement d'un élément en CSS ?

2.11.0.1 Explication Les différents modes de positionnement d'un élément en CSS sont les suivants :

- `static` : c'est le mode de positionnement par défaut. L'élément est positionné selon le flux normal du document.
- `relative` : l'élément est positionné selon le flux normal du document, mais on peut le déplacer par rapport à sa position normale.
- `absolute` : l'élément est positionné par rapport à son parent le plus proche qui a un mode de positionnement différent de `static`. Si aucun parent n'a un mode de positionnement différent de `static`, alors l'élément est positionné par rapport à la fenêtre du navigateur.
- `fixed` : l'élément est positionné par rapport à la fenêtre du navigateur.
- `sticky` : l'élément est positionné par rapport à la fenêtre du navigateur, mais il reste collé à son parent le plus proche qui a un mode de positionnement différent de `static`.
- `block` : l'élémént est positionné obligatoirement sur une nouvelle ligne, et il prend toute la largeur disponible.
- `inline` : l'élément est positionné sur la même ligne que son parent, et il ne peut pas avoir de largeur ni de hauteur.

2.12 2) Quelles sont les différentes manières d'utiliser du CSS dans une page Web ? Donner les avantages et inconvénients de chacunes, et dans quels cas les utiliser ?

2.12.0.1 Explication On a 3 manières d'appliquer du CSS aux éléments HTML :

- Modifier `inline` : on modifie l'attribut `style` de l'élément HTML. C'est pratique pour faire des tests, mais c'est pas pratique pour appliquer du CSS à plusieurs éléments. `<p style="color: red;">Hello world</p>`
- Modifier `internal` : on modifie la balise `<style>` dans le `<head>`. C'est pratique pour appliquer du CSS à plusieurs éléments, mais c'est pas pratique pour faire des tests. `<style> p { color: red; } </style>`
- Modifier `external` : on modifie le fichier CSS externe. C'est pratique pour appliquer du CSS à plusieurs éléments et pour partager le code CSS à plusieurs pages. `<link rel="stylesheet" href="style.css">`

2.13 2) Expliquez le concept d'hypermédia.

2.13.0.1 Explication L'hypermédia c'est l'ensemble des données qui sont reliées entre elles. Ex : un site web, un document, …

Ils ont une URI qui permet de les identifier.

2.14 2) Que signifie l'accessibilité pour un contenu Web ? En quoi est-ce important ?

2.14.0.1 Explication L'accessibilité pour un contenu Web c'est que le contenu est accessible à tous les utilisateurs, y compris les personnes handicapées. C'est important car on veut que tout le monde puisse accéder au contenu.

Par exemple, une image de chat ne sera pas accessible pour les malvoyants car ils ne pourront pas la voir. Il faudrait donc ajouter un texte alternatif (`alt`) pour que les malvoyants puissent comprendre ce que représente l'image.

2.15 2) Donnez les différentes méthodes (ou verbes) du protocole HTTP et leur sémantique.

2.15.0.1 Explication Les différentes méthodes (ou verbes) du protocole HTTP sont les suivantes :

- `GET` : permet de récupérer une ressource.
- `POST` : permet de créer une ressource.
- `PUT` : permet de modifier une ressource.
- `DELETE` : permet de supprimer une ressource.
- `HEAD` : permet de récupérer les métadonnées d'une ressource.
- `OPTIONS` : permet de récupérer les méthodes HTTP supportées par le serveur.
- `TRACE` : permet de récupérer les requêtes HTTP envoyées par le client.
- `CONNECT` : permet de créer un tunnel vers le serveur.
- `PATCH` : permet de modifier une ressource.

2.16 2) Comment utiliser le JavaScript dans une page Web ? (inclusion, exécution, etc.)

2.16.0.1 Explication On peut utiliser le JavaScript dans une page Web en utilisant la balise `<script>`, puis on a le choix entre écrire le code JavaScript dans la balise `<script>`, ou bien inclure un fichier JavaScript externe (`<script src="script.js"></script>`).

Le code javascript est alors exécuté lorsque le navigateur rencontre la balise `<script>`. On peut alors créer des évènements sur des éléments HTML, et le code JavaScript sera exécuté lorsque l'évènement se produit (`document.getElementById("myBtn").addEventListener("click", function(){ ... });`).

2.17 2) Quels sont les différents types de composants ou d'acteurs du web (au niveau infrastructures) ? Expliquez brièvement leurs rôles.

2.17.0.1 Explication Les différents types de composants ou d'acteurs du web sont les suivants :

- **Le client** : c'est l'utilisateur qui utilise le web. Il peut utiliser un navigateur web, un client mail, un client de messagerie instantanée, …
- **Le serveur** : c'est le serveur qui héberge les sites web. Il peut s'agir d'un serveur web, d'un serveur mail, d'un serveur de messagerie instantanée, …

Puis dans les serveurs, on peut avoir des serveurs de base de données, des serveurs de fichiers, des serveurs de DNS, … Chaque serveur a un rôle précis.

2.18 2) Donner des exemples d'élément HTML permettant de structurer un document.

2.18.0.1 Explication On peut structurer le document de manière sémantique avec les éléments suivants :

- `<header>` : pour le header
- `<nav>` : pour la navigation
- `<main>` : pour le contenu principal
- `<article>` : pour un article
- `<section>` : pour une section
- `<aside>` : pour un contenu secondaire
- `<footer>` : pour le footer

- `<h1>` à `<h6>` : pour les titres
- …

2.19 2) Quelles sont les trois couches fondamentales dans une application ? Où sont-elles situées dans une application Web ?

Note : Pas le souvenir qu'on l'ait vu.

2.19.0.1 Explication Les trois couches fondamentales sont la couche présentation, la couche métier et la couche persistance.

- La couche présentation est située dans le navigateur. Elle est chargée de l'affichage des données et de la gestion des événements.
- La couche métier est située sur le serveur. Elle est chargée de la logique métier de l'application.
- La couche persistance est située sur le serveur. Elle est chargée de la persistance des données.

Elles permettent de séparer les différentes responsabilités de l'application.

3 Plus rares (globalement)

3.1 1) Qu'est-ce que le DOM ? Comment l'utiliser ?

3.1.0.1 Explication Le DOM ou **Document Object Model** est une représentation de l'arbre HTML sous forme d'objet. On peut y trouver qui est le parent d'un élément, qui sont les enfants d'un élément, …

On peut l'utiliser en utilisant les méthodes de l'objet `document` (ex : `document.getElementById`, `document.getElementsByClassName`, …).

3.2 1) Expliquer le fonctionnement de la programmation événementielle.

3.2.0.1 Explication La programmation événementielle c'est l'utilisation d'événements pour déclencher des actions. Ex : on peut définir une action qui se déclenche quand on clique sur un bouton.

En français, on peut traduire "Quand on clique sur le bouton, on affiche un message" par `button.addEventListener("click", function(){ alert("Hello world"); });`.

3.3 1) Quelles sont les différences en le XHTML et le HTML ?

Note : Question assez poussée, pas sûr qu'on l'ait.

3.3.0.1 Explication Le XHTML et le HTML sont deux langages de balisage. Le XHTML est une version plus stricte du HTML mais qui est plus facile à parser. Le HTML est plus permissif, mais plus facile à écrire.

Le XHTML est plus strict, il faut par exemple fermer toutes les balises, et il faut utiliser des entités pour les caractères spéciaux. Le HTML est plus permissif, on peut par exemple ne pas fermer les balises, et on peut utiliser des caractères spéciaux sans utiliser d'entités.

3.4 1) À quoi sert le JavaScript dans une page Web ?

3.4.0.1 Explication Le JavaScript sert à ajouter de l'interactivité à une page Web. On peut par exemple ajouter des animations, des interactions avec l'utilisateur, …

Sela rajoute de la dynamique à une page Web (“Quand je clique sur ce bouton, il se passe quelque chose”).

3.5 1) Quelles sont les avantages et inconvénients du XHTML par rapport au HTML ?

3.5.0.1 Explication *Note : je ne sais pas quoi dire d'autre*

Les avantages du XHTML par rapport au HTML sont les suivants :

- Le XHTML est plus strict que le HTML. Il est plus facile de vérifier la validité d'un document XHTML.

3.6 1) Qu'appelle-t-on Ajax ? Quel en est l'intérêt ? Quelles en sont les limitations ?

3.6.0.1 Explication Ajax est un acronyme pour Asynchronous JavaScript And XML. C'est un ensemble de technologies permettant de pouvoir faire des requêtes HTTP en arrière-plan, sans avoir à recharger la page. Cela permet de pouvoir faire des requêtes HTTP sans avoir à recharger la page, et de pouvoir mettre à jour la page sans avoir à recharger la page.

Les limitations d'Ajax sont les suivantes :

- Les navigateurs ne supportent pas toutes les technologies d'Ajax.
- Les requêtes HTTP ne sont pas toujours possibles.
- Le code est en clair dans le navigateur, donc on ne peut pas faire de requêtes par exemple de connexion ou connection.

3.7 1) Quels autres format que le HTML peut-il être intéressant de fournir au client comme représentation d'une ressource, et pourquoi ? Comment délivrer ces formats ?

3.7.0.1 Explication Il peut être intéressant de fournir au client d'autres formats que le HTML comme représentation d'une ressource. Par exemple, on peut fournir au client un format XML, qui est un format de données, ou encore un format JSON, qui est un format de données. Cela permet de pouvoir fournir au client des données, et non pas seulement du HTML.

On peut délivrer ces formats en utilisant le header `Content-Type` dans la réponse HTTP : `Content-Type: application/json`.

3.8 1) Qu'est-ce qu'une fonction d'ordre supérieur ?

3.8.0.1 Explication Une fonction d'ordre supérieur est une fonction qui prend en paramètre une fonction ou qui renvoie une fonction. Par exemple la fonction `map` de javascript est une fonction d'ordre supérieur : elle prend en paramètre une fonction et renvoie une fonction.

```
const myArray = [1, 2, 3, 4, 5];

const myFunction = (x) => x * 2;

const myNewArray = myArray.map(myFunction);
```

3.9 1) Qu'est-ce qu'une ressource ?

3.9.0.1 Explication Une ressource c'est un objet qui peut être identifié par une URL (URI). Par exemple, une page web est une ressource. Une image est une ressource, un fichier est une ressource, …Elles sont stockées sur le serveur.

3.10 1) Donnez les cinq catégories de code de status HTTP et leurs significations.

3.10.0.1 Explication Les cinq catégories de code de status HTTP sont les suivantes :

- 1xx : Information
- 2xx : Succès
- 3xx : Redirection
- 4xx : Erreur du client
- 5xx : Erreur du serveur

3.11 1) Quels sont les paradigmes du JavaScript ?

Note : Question assez poussée, pas sûr qu'on l'ait.

3.11.0.1 Explication Les paradigmes du JavaScript sont les suivants :

- Programmation impérative
- Programmation fonctionnelle
- Programmation orientée objet
- Programmation asynchrone
- Programmation événementielle

3.12 1) Quelles sont les propriétés principales du protocole HTTP ?

3.12.0.1 Explication *Notes : je ne suis pas sûr de la réponse à cette question. MAIS cette question peut-être qu'on ne l'aura pas car plutôt orientée serveur.*

Les propriétés principales du protocole HTTP sont :

- Le protocole HTTP est un protocole sans état. Cela signifie que le serveur ne garde pas en mémoire les requêtes précédentes.
- Le protocole HTTP est un protocole orienté client. Cela signifie que le client envoie une requête au serveur, et le serveur répond à la requête.
- Le protocole HTTP est un protocole orienté texte. Cela signifie que les données sont envoyées sous forme de texte (hyper**text**).
- Le protocole HTTP est un protocole orienté requête/réponse. Cela signifie que le client envoie une requête au serveur, et le serveur répond à la requête.

3.13 1) Quels sont les apports du HTML5 ?

Notes : je ne suis pas sûr de la réponse à cette question. Mais frérot, frérot….. CHUIS PAS UNE DOC, JE DOIS VRMNT SAVOIR ÇA ?? :(

Note 2 : Euh le message en majuscule c'est une IA qui l'a écrit xD

3.14 1) Qu'est-ce qu'une application internet riche ? Quels en sont les avantages et inconvénients ?

3.14.0.1 Explication Une application internet riche c'est une application qui est exécutée dans le navigateur de l'utilisateur. Celui-ci n'a pas besoin d'installer l'application sur son ordinateur : pour les mise à jour, il suffit de recharger la page.

Les avantages sont :

- Pas besoin d'installer l'application sur l'ordinateur de l'utilisateur
- Pas besoin de mettre à jour l'application sur l'ordinateur de l'utilisateur

Les inconvénients sont :

- Le navigateur doit être à jour
- Le navigateur doit supporter l'application
- L'ordinateur fait tout les calculs, donc il peut être lent
- L'ordinateur doit avoir une bonne connexion internet des fois

3.15 1) Pourquoi dit-on que Javascript est un langage interprété ?

3.15.0.1 Explication JavaScript est un langage interprété car le code JavaScript est interprété par le navigateur. C'est pour cela que le code JavaScript est exécuté ligne par ligne.

3.16 1) Donner une expression régulière qui valide un numéro de téléphone et acceptant différentes écritures

Note : On ne l'aura sûrement pas, on a pas vu les Regex.

3.16.0.1 Explication On peut faire :

```
var regex = /^(\+33|0)[1-9](\d{2}){4}$/;
// Description :
// On commence par un +33 ou un 0
// On a un chiffre entre 1 et 9
// On a 4 groupes de 2 chiffres
```