Web-Client, correction Annales

VAN DE MERGHEL Robin

Semestre 4

Table des matières

Note supplémentaire : Sur certaines questions, il se peut que j'hésite sur une réponse, je donnerai une réponse (mon interprétation), et je rajouterai une note ainsi qu'une astérisque pour indiquer que je ne suis pas sûr de la réponse.

1 Annale CC 2021-2022

1.1 Exercice 1

1.1.1 Question 1

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

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

1.1.2 Question 2

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

Le principe du DNS est de convertir une **URI** en **adresse IP**.

L'utilité est de pouvoir accéder à une ressource sur internet sans avoir à connaître son adresse IP (il est plus facile de retenir le nom d'un site, qu'une suite de 12 chiffres).

1.1.3 Question 3

Expliquez le concept d'hypermédia.

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.

1.1.4 Question 4*

Expliquez le concept d'opacité des URL. En quoi est-ce important?

Une adresse URL n'est pas censée être lisible par l'utilisateur. C'est le serveur qui doit comprendre l'URL et savoir comment la traiter.

L'utilisateur ne fait que suivre un lien, il ne doit pas avoir à comprendre ce qu'il y a derrière. Exemple : les URL des vidéos youtubes sont illisibles pour l'utilisateur, mais le serveur sait comment les traiter.

Note : je ne suis pas sûr de la réponse suivante, mais je pense que c'est ça :

C'est important car l'utilisateur ne doit pas avoir à comprendre ce qu'il y a derrière, il doit juste suivre le lien.

1.2 Exercice 2

1.2.1 Question 1

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

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.

1.2.2 Question 2

Expliquer le principe de la cascade des règles CSS.

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.

1.2.3 Question 3

Que sont les « media queries » en CSS ? Quels en sont les intérêts ?

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 */
}
```

1.3 Exercice 3

1.3.1 Question 1

Qu'est-ce que le HTML?

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, …).

1.3.2 Question 2

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

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.

1.3.3 Question 3

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

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).

1.4 Exercice 4

1.4.1 Question 1

Donnez-la valeur de x et expliquez

```
let v = (function(x) {
    let c = x;
    return function(y) {
        c *= y;
        return c;
    }
})(7);

v(2);
let x = v(3);
```

$x = 42$ car la fonction `v` est appelée avec l'argument `3`, donc $c = 7 \times 3 = 21$. Ensuite, la fonction `v` est appelée avec l'argument `2`, donc $c = 21 \times 2 = 42$.

1.4.2 Question 2

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

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`, …).

1.4.3 Question 3

Donner et expliquer la valeur de :

```
[1, 2, 3] == [1, 2, 3]
```

La valeur est `false` car les tableaux sont des objets, et les objets sont comparés par référence. Donc même si les tableaux ont les mêmes valeurs, ils ne sont pas égaux.

```
[1, 2, 3] === [1, 2, 3]
```

La valeur est `false` car les tableaux sont des objets, et les objets sont comparés par référence. Donc même si les tableaux ont les mêmes valeurs, ils ne sont pas égaux.

1.4.4 Question 4

Expliquer le fonctionnement de la programmation événementielle.

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"); });`.

1.4.5 Question 5*

Quelle est la praticularité du système objet de javascript ? Comment fonctionne-t-il ?

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

2 Annales 2021-2022 1ère session

2.1 Exercice 1

2.1.1 Question 1*

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

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.

2.1.2 Question 2

En quoi le design des URL est important ? Qu'est-ce qu'uene "bonne" URL ?

Une URL doit être comphénsible par le serveur, mais pas forcément par les utilisateurs. On arrive sur une ressource qu'après avoir navigué URL par URL, donc on peut se permettre d'avoir des URL un peu longues et complexes.

De plus, l'utilisateur n'est pas sencé comprendre l'URL. Même des fois, il ne vaut mieux pas comme pour Google Drive sinon on pourrait accéder à des fichiers dont on n'a pas les droits. Avoir une ressource avec le même nom aurait provoqué une erreur car les deux ressources auraient eu la même URL.

2.1.3 Question 3*

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

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, …)

2.1.4 Question 4

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

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.

2.2 Exercice 2

2.2.1 Question 1

Donner les différentes manières de spécifier une couleur en CSS.

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)`, …)

2.2.2 Question 2

Donner la syntaxe générale d'une règle CSS.

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

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

2.2.3 Question 3

Qu'est-ce qu'un design de page flexible (ou fluide) ? Pourquoi est-ce important ?

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.

2.2.4 Question 4

Quelles sont les différentes manières d'appliquer du CSS aux éléments HTML ? Donner les avantages et inconvénients de chacunes, et dans quel cas les utiliser.

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">`

2.2.5 Question 5

Quel est le principe des web fonts ?

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.

2.3 Exercice 3

2.3.1 Question 1

Que signifie pour un document d'être bien formé ?

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, …

2.3.2 Question 2

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

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.

2.3.3 Question 3*

Note : jsp quoi rep, c'est assez spécial

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

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, …

2.3.4 Question 4

Donner des exemples d'éléments HTML permettant de structurer le document de manière sémantique (au moins cinq).

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
- …

2.3.5 Question 5

Quelles sont les différences en le XHTML et le HTML ?

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.

2.4 Exercice 4

2.4.1 Question 1

À quoi sert le JavaScript dans une page Web ?

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").

2.4.2 Question 2

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

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(){ ... });`).

2.4.3 Question 3*

Note : je ne sais pas quoi rajouter

Expliquer la propagration des évènements dans le navigateur.

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.

2.4.4 Question 4

Quelle est la différence en Javascript entre `==` et `===` ?

En Javascript, `==` compare deux valeurs, et `===` compare deux valeurs et deux types. Par exemple, `1 == "1"` est vrai, mais `1 === "1"` est faux car le type de `1` est `number` et le type de `"1"` est `string`.

2.4.5 Question 5

Que vaut `this` en Javascript ?

`this` vaut l'objet qui appelle la fonction. Par exemple, si on a une fonction `myFunction` qui est appelée par un objet `myObject`, alors `this` vaudra `myObject` dans la fonction `myFunction`.

Dans une classe, `this` vaudra l'instance de la classe.

2.4.6 Question 6

Expliquer le système de transtypage en Javascript.

Le système de transtypage en Javascript c'est le fait que le langage Javascript est un langage faiblement typé. Cela signifie que le langage Javascript est capable de convertir automatiquement les types de données. Par exemple, si on fait `1 + "1"`, le résultat sera `"11"` car le langage Javascript convertit automatiquement le `1` en `string`.

3 Annale 2020-2021 (1ère session)

3.1 Exercice 1

3.1.1 Question 1

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

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.

3.1.2 Question 2

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

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`).

3.2 Exercice 2

3.2.1 Question 1

Donner les différentes manières de spécifier une couleur en CSS.

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)`, …)

3.2.2 Question 2

Comment mettre en oeuvre un design de page flexible ?

Pour mettre en oeuvre un design de page flexible, on peut utiliser les unités relatives suivantes :

- `em` : l'unité relative `em` est relative à la taille de la police de l'élément. Par exemple, si on a un élément avec une taille de police de 16px, et qu'on met une marge de 1em, alors la marge sera de 16px.

- `rem` : l'unité relative `rem` est relative à la taille de la police de la racine. Par exemple, si on a un élément avec une taille de police de 16px, et qu'on met une marge de 1rem, alors la marge sera de 16px.
- `%` : l'unité relative `%` est relative à la taille de l'élément parent. Par exemple, si on a un élément avec une taille de 100px, et qu'on met une marge de 50%, alors la marge sera de 50px.
- `vw` ou `vh` : l'unité relative `vw` ou `vh` est relative à la taille de la fenêtre du navigateur. Par exemple, si on a un élément avec une taille de 100px, et qu'on met une marge de 50vw, alors la marge sera de 50% de la largeur de la fenêtre du navigateur.

3.2.3 Question 3

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

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.

3.2.4 Question 4*

Je ne sais pas pour le 3.

Comment sélectionner en CSS : 1. Les zones de navigation ayant la classe `skip`; 2. Les paragraphes fils direct d'une section; 3. Les éléments mis en évidence dans un encart; 4. Les images suivant un titre de deuxième niveau ?

Pour sélectionner en CSS :

1. Les zones de navigation ayant la classe `skip`, on peut utiliser la sélecteur `.skip`. Par composition, on peut utiliser la sélecteur `nav.skip`.
2. Les paragraphes fils direct d'une section, on peut utiliser la sélecteur `section > p`.
3. *Je ne sais pas trop quoi dire, en admettant que ce soit `aside` :* `aside em`.
4. Les images suivant un titre de deuxième niveau, on peut utiliser la sélecteur `h2 + img`.

3.2.5 Question 5

Donner la syntaxe générale d'une règle CSS.

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

```
sélecteur {
    propriété: valeur;
}
```

3.3 Exercice 3

3.3.1 Question 1

Qu'est-ce que le HTML ?

Le HTML est un langage de balisage. Il permet de décrire la structure d'une page web. Il est composé de balises, qui sont des éléments qui décrivent la structure de la page web. Par exemple, on peut avoir une balise `<p>` pour décrire un paragraphe, une balise `<h1>` pour décrire un titre de niveau 1, une balise `<img>` pour décrire une image, …

3.3.2 Question 2

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

Un document HTML est valide s'il respecte la syntaxe du langage HTML. C'est à dire que toutes les balises sont bien fermées, et que les attributs sont bien écrits, et que les balises existent bien.

C'est important car cela permet de s'assurer que le document HTML est bien structuré, et que les navigateurs peuvent l'afficher correctement.

3.3.3 Question 3*

Note : jsp quoi rep, c'est assez spécial

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

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, …

3.4 Question 4

Qu'appelle-t-on du HTML sémantique ? Quels en sont les intérêts ?

Le HTML sémantique permet de pouvoir décrire la structure d'une page web de par son contenu et non sa forme. Par exemple, on peut avoir une balise `<p>` pour décrire un paragraphe, une balise `<h1>`

pour décrire un titre de niveau 1, une balise `<img>` pour décrire une image, … On indique le sens d'une balise, et non sa forme.

De faire cela permet aussi de pouvoir rendre accessible une page web, et de pouvoir la rendre compréhensible par les moteurs de recherche.

3.4.1 Question 5*

Note : jsp quoi rep, c'est assez spécial

> **Quelles sont les avantages et inconvénients du XHTML par rapport au HTML ?**

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.

3.5 Exercice 4

3.5.1 Question 1

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

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.

3.5.2 Question 2

> **En le design des URL est important ? Qu'est-ce qu'une bonne URL ?**

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.

3.5.3 Question 3

> **Donnez les différentes méthodes (ou verbes) du protocole HTTP et leur sémantique.**

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.

3.6 Exercice 5

3.6.1 Question 1

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

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

3.6.2 Question 2

Que valent les expressions suivantes ?

1.

```
new Date(0) + 0
```

Le code ci-dessus va afficher la date du 1er janvier 1970.

```
new Date(0) - 0
```

Le code ci-dessus va afficher `0`.

3.6.3 Question 3

Donner et expliquer le résultat des expressions suivantes :

- `{reponse: 42} == {reponse: 42}`
- `{reponse: 42} === {reponse: 42}`

Les deux expressions ci-dessus vont afficher `false`. C'est parce que les deux objets sont différents, même si ils ont la même valeur.

3.6.4 Question 4

Donner et expliquer le résultat des expressions suivantes :

- `new String("aa")== "aa"`
- `new String("aa")=== "aa"`

On a `true` pour la première expression, et `false` pour la deuxième expression. C'est parce que la première expression compare les valeurs des deux objets, alors que la deuxième expression compare les objets eux-mêmes.

3.6.5 Question 5

Donnez la valeur de x et expliquez :

```
let v = (function(x) {
    let c = x;
    return function(y) {
        c *= y;
        return c;
    }
})(7);

v(2);
let x = v(3);
```

$x = 42$ car la fonction `v` est appelée avec l'argument `3`, donc $c = 7 \times 3 = 21$. Ensuite, la fonction `v` est appelée avec l'argument `2`, donc $c = 21 \times 2 = 42$.

3.6.6 Question 6

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"]]`.

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

4 Annale 2027-2018 (1ère session)

4.1 Exercice 1

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

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.

4.2 Exercice 2

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 ?

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`.

4.3 Exercice 3

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

Le principe du DNS est de convertir une **URI** en **adresse IP**.

L'utilité est de pouvoir accéder à une ressource sur internet sans avoir à connaître son adresse IP (il est plus facile de retenir le nom d'un site, qu'une suite de 12 chiffres).

4.4 Exercice 4

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

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

4.5 Exercice 5

Expliquer le concept d'hypermédia.

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.

4.6 Exercice 6

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

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.

4.7 Exercice 7

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 ?

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">`

4.8 Exercice 8

Comment sélectionner en CSS :

- **un paragraphe ayant la classe `abstract`;**

- **la section identifiée par `sec1`;**
- **un lien survolé par la souris;**
- **un champ de formulaire actif;**
- **un paragraphe fils direct d'une section;**
- **un élément mis en évidence dans une note;**
- **une image suivant un titre de deuxième niveau ?**
- `p.abstract`
- `section#sec1`
- `a:hover`
- `input:focus`
- `section > p`
- `p em`
- `img + h2`

4.9 Exercice 9

Quel est le principe des web fonts ?

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.

4.10 Exercice 10

Donner les différentes manières de définir une couleur en CSS.

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)`, …)

4.11 Exercice 11

Qu'indique la ligne `DOCTYPE` d'un fichier HTML ? Pourquoi est-ce important et comment est-ce utilisé ?

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.

4.12 Exercice 12

Que signifie pour un document d'être bien formé ?

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, …

4.13 Exercice 13

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

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
- …

4.14 Exercice 14*

Note : jsp quoi rep, c'est assez spécial

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

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, …

4.15 Exercice 15

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

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.

4.16 Exercice 16

Donnez la valeur et expliquez :

```
(function (x,z) {
    return function (y) {return ((x<y) ? y-x : x+y) * z} })(6,10)(11) - 5
```

La valeur de cette expression est 45 car :

- D'abord on appelle la fonction anonyme avec les paramètres `6` et `10`. On obtient une fonction qui prend en paramètre `y` et qui renvoie `((6<y) ? y-6 : 6+y) * 10`.
- Puis on appelle cette fonction avec le paramètre `11`. On obtient `((6<11) ? 11-6 : 6+11) * 10` qui vaut `50`.
- Enfin on soustrait 5 à 50, on obtient 45.

4.17 Exercice 17*

Note : je ne sais pas quoi rajouter

Expliquer la propagration des évènements dans le navigateur.

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.

4.18 Exercice 18*

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

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

4.19 Exercice 19

Que vaut `this` ?

`this` vaut l'objet qui appelle la fonction. Par exemple, si on a une fonction `myFunction` qui est appelée par un objet `myObject`, alors `this` vaudra `myObject` dans la fonction `myFunction`.

Dans une classe, `this` vaudra l'instance de la classe.

4.20 Exercice 20

Donner et expliquer le résultat des expressions suivantes :

- `{reponse: 42} == {reponse: 42}`
- `{reponse: 42} === {reponse: 42}`

Les deux expressions ci-dessus vont afficher `false`. C'est parce que les deux objets sont différents, même si ils ont la même valeur.

4.21 Exercice 21

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

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);
```

4.22 Exercice 22

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

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.

4.23 Exercice 23

Que signifie l'accessibilité pour un contenu web ? En quoi est-ce important ?

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.

5 Annale 2017-2018 (2e session)

5.1 Exercice 1

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

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

- Le client : c'est le navigateur web, les crawlers, les bots, … qui interagissent avec le serveur.
- Le serveur : c'est le serveur web qui reçoit les requêtes du client et qui renvoie les réponses au client.
- Le proxy : c'est un serveur qui permet de faire passer les requêtes du client au serveur. C'est utile pour les entreprises qui veulent filtrer les requêtes des clients.

5.2 Exercice 2

Quels sont les buts recherchés dans l'architecture du WEB ?

Les buts recherchés dans l'architecture du WEB sont les suivants :

- Le WEB doit être ouvert : tout le monde peut créer des pages web et les héberger sur le WEB.
- Le WEB doit être distribué : les serveurs sont répartis dans le monde entier.
- Le WEB doit être évolutif : il doit être possible d'ajouter des fonctionnalités au WEB sans casser les anciennes fonctionnalités.
- Le WEB doit être stable : les pages web doivent rester accessibles même si un serveur tombe en panne.

5.3 Exercice 3

Qu'est-ce qu'une ressource ?

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.

5.4 Exercice 4

En quoi le design des urls est important ? Qu'est-ce qu'une bonne url ?

Une URL doit être comphénsible par le serveur, mais pas forcément par les utilisateurs. On arrive sur une ressource qu'après avoir navigué URL par URL, donc on peut se permettre d'avoir des URL un peu longues et complexes.

De plus, l'utilisateur n'est pas sencé comprendre l'URL. Même des fois, il ne vaut mieux pas comme pour Google Drive sinon on pourrait accéder à des fichiers dont on n'a pas les droits. Avoir une ressource avec le même nom aurait provoqué une erreur car les deux ressources auraient eu la même URL.

5.5 Exercice 5

Que sont les media queries en CSS ? Quels en sont les intérêts ?

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 */
}
```

5.6 Exercice 6

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

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.

5.7 Exercice 7

Donner les unités de mesures relatives en CSS.

Les unités de mesures relatives en CSS sont les suivantes :

- `em` : 1em = la taille de la police du parent
- `rem` : 1rem = la taille de la police de la racine
- `%` : 1% = 1% de la taille du parent
- `vw` et `vh` : 1vw = 1% de la largeur de l'écran, 1vh = 1% de la hauteur de l'écran
- `ch` : 1ch = la largeur d'un caractère

5.8 Exercice 8

Quels sont les différents modes de positionnement d'élements en CSS ?

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.

5.9 Exercice 9

Qu'est-ce que le HTML ?

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, …).

5.10 Exercice 10

Qu'appelle-t-on du HTML sémantique ? Quels sont les intérêts ?

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.

5.11 Exercice 11

Que signifie être valide pour du HTML ? En quoi est-ce important ?

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).

5.12 Exercice 12

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

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.

5.13 Exercice 13

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

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

5.14 Exercice 14

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

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
```

5.15 Exercice 15

Quelle est la différence en JavaScript entre == et === ?

`==` compare deux valeurs, et `===` compare deux valeurs et deux types. Par exemple :

```
1 == "1"; // true
1 === "1"; // false
```

5.16 Exercice 16

Comment faire de l'héritage en JavaScript ?

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;
    }
}
```

5.17 Exercice 17

Quels sont les paradigmes du JavaScript ?

Les paradigmes du JavaScript sont les suivants :

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

5.18 Exercice 18

Expliquer le système de transtypage en JavaScript.

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
```

5.19 Exercice 19

Comment utiliser le Javascript dans une page Web ? (inclusion, exécution, ...)

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(){ ... });`).

5.20 Exercice 20

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"]]`.

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

5.21 Exercice 21

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 :

```
var add = function(a, b) { return a + b; };
var addCurry = function(a) { return function(b) { return a + b; }; };
```

La méthode `bind(this, arg1, arg2...)` de l'objet fonction crée une version partialisée de la fonction. Par exemple :

```
var add2 = add.bind(null, 2);
console.log(add2(40)); // 42
```

En utilisant le paramètre spécial `arguments`, la propriété `length` de l'objet fonction qui donne son nombre de paramètres, et la méthode `bind`, créer une fonction `curry` (miam c bon d'ailleurs) générique, qui accepte en paramètre n'importe quelle fonction et qui retourne une version curryfiée de cette fonction.

```
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));
            }
        }
    };
}
```

5.22 Exercice 22

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

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`.

6 Annale 2016-2017 (1ère session)

6.1 Exercice 1

Quelles sont les trois couches fondamentales dans une application ? Où sont-elles situées dans une application WEB ?

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.

6.2 Exercice 2

En quoi le design des URL est important ? Qu'est-ce qu'uene "bonne" URL ?

Une URL doit être comphénsible par le serveur, mais pas forcément par les utilisateurs. On arrive sur une ressource qu'après avoir navigué URL par URL, donc on peut se permettre d'avoir des URL un peu longues et complexes.

De plus, l'utilisateur n'est pas sencé comprendre l'URL. Même des fois, il ne vaut mieux pas comme pour Google Drive sinon on pourrait accéder à des fichiers dont on n'a pas les droits. Avoir une ressource avec le même nom aurait provoqué une erreur car les deux ressources auraient eu la même URL.

6.3 Exercice 3

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

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

6.4 Exercice 4*

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.

Quelles sont les propriétés principales du protocole HTTP ?

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.

6.5 Exercice 5

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

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.

6.6 Exercice 6

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

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.

6.7 Exercice 7*

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

Quels sont les apports du HTML5 ?

6.8 Exercice 8

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
- …

6.9 Exercice 9

Expliquer le principe de la cascade des règles CSS.

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.

6.10 Exercice 10

Donner la syntaxe générale d'une règle CSS.

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

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

6.11 Exercice 11

Qu'est-ce qu'un design de page flexible (ou fluide) ? Pourquoi est-ce important ?

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.

6.12 Exercice 12

Quel est le principe des web fonts ?

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.

6.13 Exercice 13

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

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

6.14 Exercice 14

Que vaut :

```
(function (x) {
    return function (y) { return (x % y == 0) ? x / y : x * y; } }
)(16)(4) * 334 + 1;
```

La réponse est 1337.

En effet, on a :

- `x = 16`
- Le code de la fonction est `return (x % y == 0)? x / y : x * y;`
- `y = 4`
- `x % y == 0` est vrai
- `x / y` est égal à 4
- `x * y` est égal à 64
- `x` est égal à 4
- `x * 334 + 1` est égal à 1337

On en parle de l'exercice ?

6.15 Exercice 15*

Note : je ne sais pas quoi rajouter

Expliquer la propagration des évènements dans le navigateur.

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.

6.16 Exercice 16

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 :

```
var add = function(a, b) { return a + b; };
var addCurry = function(a) { return function(b) { return a + b; }; };
```

La méthode `bind(this, arg1, arg2...)` de l'objet fonction crée une version partialisée de la fonction. Par exemple :

```
var add2 = add.bind(null, 2);
console.log(add2(40)); // 42
```

En utilisant le paramètre spécial `arguments`, la propriété `length` de l'objet fonction qui donne son nombre de paramètres, et la méthode `bind`, créer une fonction `curry` (miam c bon d'ailleurs) générique, qui accepte en paramètre n'importe quelle fonction et qui retourne une version curryfiée de cette fonction.

```
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));
            }
        }
    };
}
```

6.17 Exercice 17

Que valent les expressions suivantes ?

1.

```
new Date(0) + 0
```

Le code ci-dessus va afficher la date du 1er janvier 1970.

```
new Date(0) - 0
```

Le code ci-dessus va afficher `0`.

6.18 Exercice 18

Pourquoi dit-on que JavaScript est un langage dynamique ?

JavaScript est un langage dynamique car on peut changer le type d'une variable à tout moment. Par exemple, on peut faire :

```
var a = 1;
a = "Hello";
```

De plus on peut faire :

```
var a = 1;
a = a + "Hello";
```

6.19 Exercice 19

Comment faire de l'héritage en JavaScript ?

En JavaScript, on peut faire de l'héritage en utilisant le mot clé `extends`. Par exemple, on peut faire :

```
class A {
    constructor() {
        this.a = 1;
    }
}

class B extends A {
    constructor() {
        super();
        this.b = 2;
    }
}
```

6.20 Exercice 20

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

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`.

6.21 Exercice 21*

Note : Euh je sais pas quoi dire de plus à part Captain Obvious, on a eu aucune info

Pourquoi dit-on que Javascript est un langage interprété ?

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.

6.22 Exercice 22

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

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.

6.23 Exercice 23*

Note : Je sais pas quoi dire de plus, pour le coup ça on l'a absolument pas vu

Donner une expression régulière qui valide un numéro de téléphone et acceptant différentes écritures :

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
```

6.24 Exercice 24

Que vaut `this` en Javascript ?

`this` vaut l'objet qui appelle la fonction. Par exemple, si on a une fonction `myFunction` qui est appelée par un objet `myObject`, alors `this` vaudra `myObject` dans la fonction `myFunction`.

Dans une classe, `this` vaudra l'instance de la classe.

6.25 Exercice 25

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"]]`.

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

6.26 Exercice 26

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

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
```