Visualisation de données avec d3.js – TP1

Arthur Katossky

28/02/2019

1. HTML (~30 min)

Le HTML est un langage de description de la **structure** et de la **sémantique** d'une page web. Il est composé de **texte** entrecoupé d'un nombre limité de **balises** :

```
<html>
  <div>
    <h1>Gap Minder</h1>
    <p>
      The information visualization technique used by
      Trendalyzer is an interactive bubble chart.
    </p>
  </div>
  <div>
    <img src="http://bi.gazeta.pl/im/9d/5c/14/z21349277Q,Gapminder.jpg">
  </div>
</html>
```

Les balises peuvent aller par paire (une balise ouvrante `<div>` et une fermante `</div>`) ou être autonomes (`<img>`). Le nom des balises est limité à une liste définie par les spécifications du HTML par le W3C (World Wide Web Consortium). Les balises peuvent enfin avoir des *attributs* (dans la balise ouvrante) comme ici `src` pour la source de l'image dans la balise `<img>`.

Le HTML code simmultanément deux aspects de la page :

- sa structure. On entend par là la hiérarchie des éléments de la page (ou objets), ce qu'on appelle le DOM (*document object model*) : il s'agit d'un arbre dont la racine est le document entier (*document*) et les feuilles sont les zones de texte de la page (potentiellement vides).
- sa sémantique. En effet, les balises décrivent le contenu de la page et le sens de ce qui est représenté, et le plus possible indépendamment de la forme. Ainsi la balise `<img>` annonce une image, `<h1>` le titre principal, `<p>` un paragraphe, etc.

Certaines balises (`<div>` par exemple) dénotent uniquement la structure sans apporter aucune information sur le sens.

Exercices

JsFiddle est un site qui permet d'exécuter un code HTML sans se préoccuper d'installer un serveur web (ou autres considérations techniques). Créez un compte puis ouvrez et sauvegardez la trame suivante : https://jsfiddle.net/katossky/4he3ftqz/

1. Listez toutes les balises utilisées dans l'exemple. Dans un tableau, vous distinguerez les balises simples des balises doubles, et les balises porteuses de sens de celles qui n'indiquent que la structure du document.
2. Comment écrit-on des commentaires en HTML ?
3. Quelle est la différence entre `<div>` et `<span>` ?
4. À votre avis, à quoi sert l'attribut `id` ? Quelle est la différence entre les attributs `id` et `class` ?
5. Dans la colonne destinée à contenir l'interface de contrôle, ajoutez :

— un bouton-poussoir “Réinitialiser”
— un bouton-poussoir “Graphique aléatoire”

Indice : https://www.w3schools.com/html/html_form_elements.asp

En attendant :

a. Par quoi pourrait-on remplacer `<div id="header">` ? (*Indice ici.*) `<span class='citation'>` ? Remplacez ainsi toutes les balises génériques par des balises prévues à cet effet.
b. Dans votre navigateur, affichez le code source de la page https://fr.wikipedia.org/wiki/Hans_Rosling. Expliquez la différence d'affichage entre la première URL et : https://fr.wikipedia.org/wiki/Hans_Rosling#Études_et_carrière
c. Dans le code source de la page https://fr.wikipedia.org/wiki/Hans_Rosling, à l'intérieur de la balise `<body>...</body>` (qui représente le corps du document, c'est-à dire sa partie visible), repérez des balises encore inconnues et cherchez leurs cas d'utilisation.
d. Dans la colonne destinée à contenir l'interface de contrôle, ajoutez églement :
   — un menu-déroulant “Ordonnées” avec comme valeurs “Espérance de vie” et “Émissions de CO ”
   — un bouton-glissière “Années”
e. Quelle est la différence entre le HTML 5 et le XHTML ? En quoi le HTML 5 ne respecte pas le XML ?

Correction : https://jsfiddle.net/katossky/b402pr9v

2. CSS (~50 min)

Le CSS est un langage de description de **l'apparence** des objets du DOM. Il est composé de **sélecteurs** (des expressions qui détaillent quels sont les objets concernés), chacun suivi d'une série de **déclarations**, placées entre accolades et terminées par un point-virgule :

```
div {
  border: solid 1px red;
  background: pink;
  /* color: black */
}

h1, p {
  margin: 10px;
  border: dashed 1px green;
  background: white;
}

.citation{
  font-style: italic;
  background: #AAA;
  border-radius: 5px;
  padding: 2px;
}
```

Exercices

Copiez le code précédent dans votre JSFiddle.

1. Repérez : comment affecter un élément donné par son identifiant ; comment affecter simmultanément deux types d'éléments différents ; comment affecter tous les objets d'une classe donnée.
2. Repérez : comment modifier la bordure, le fond, la police de caractères d'un élément.

3. Quelle est la différence entre `padding: 10px` et `margin: 10px` ?
4. Changez la taille d'affichage du texte en pied de page. (*Indice (ici)[https://www.w3schools.com/css/css_font.asp].*)
5. Peut-on ajouter des commentaires en CSS ?

Il existe des sélecteurs plus complexes, dont vous trouverez la lise ici. Le CSS permet également de disposer convenablement sur la page les différents composants ou blocs :

```
#sidebar {
  float: left;
  width: 30%;
  height: 300px;
}

#content {
  float: left;
  width: 70%;
  height: 300px;
}

#content img {
  width: 100%;
}
```

6. Que signifie `float: left` ?
7. Pourquoi est-ce important de préciser l'attribut `width` de l'image ? Pourquoi l'image n'occupe-t-elle pas 100% de la largeur de la zone d'affichage ?
8. Ce qui vaut pour `width` vaut également pour d'autres paramètres comme `font-size`. Changez la taille d'affichage du texte en pied de page, **relativement à** la taille du texte du document principal.

Enfin. le CSS permet un niveau minimal d'interactivité, grace à un système de pseudo-classes. L'interactivité est utile lorsqu'elle permet de suggérer une action possible à l'utilisateur (un bouton qui change de couleur au survol) ou pour confirmer qu'une action a été effectuée (un lien qui change couleur une fois cliqué). Elle est à éviter le reste du temps !

```
a {
  color: grey;
  text-decoration: none;
  font-weight: bold;
}

a:hover {
  color: #2ea297;
}

a:visited {
  color: #93380d;
}
```

9. Commentez le code précédent.
10. Que se passe-t-il si on rajoute `transition: color 0.5s;` au sélecteur `a` ?
11. Adaptez le code CSS pour que la couleur des boutons “Réinitialiser” et “Graphique aléatoire” change au survol.

Gérer l'apparence d'une page, même très simple, avec du CSS n'est pas une tâche aisée du tout ! En effet, sans même parler des goûts esthétiques, les navigateurs n'interprètent pas toutes les balises de la même

façon, certains paramètres par défaut sont inadéquats (ex : cadre bleu autour d'une image cliquable) et il est complexe de concevoir une page qui s'adapte convenablement à l'extrême diversité des tailles d'écran surlesquels des internautes peuvent être amenés à consulter une page (*reactive design*). Il existe une grande quantité de bibliothèques spécialisées, comme Bootstrap, pour assurer un design stable. Mais il est vain pour un statisticien de vouloir créer des interfaces parfaites : c'est un autre métier, alors faîtes confiance à des spécalistes !

En attendant : renseignez-vous sur Bootstrap et le *design* adaptatif de pages Web (*reactive design*).

3. JavaScript (~1h20) [1]

JavaScript est un langage de programmation qui s'est développé (depuis son apparition sur le navigateur Netscape en 1996) pour compenser les limitations du duo HTML + CCS, qui ne permet pas d'interaction complexe sur une seule et même page Web. En effet, initialement, l'interaction supposait d'utiliser un langage côté serveur comme PHP pour générer de *nouvelles* pages. Or télécharger une nouvelle page peut prendre plusieurs secondes, ce qui limite énormément les cas où ce type d'interaction est pertinent (envois de formulaires, téléchargement de données, etc.).

Le JavaScript n'a rien à voir avec Java, si ce n'est quelques similitudes de syntaxe. C'est un **langage multi-paradigme** (ni purement impératif, ni purement orienté-objet, ni purement fonctionnel) dont nous n'allons explorer ensemble qu'une toute petite partie : la **programmation événementielle**.

La programmation événementielle est un paradigme de programmation permettant l'appel d'une fonction lors du déclenchement d'un événement, comme un clic, le survol d'un élément, le déplacement de la souris, etc. Les fonctions "s'abonnent à" (*listen to*) certains événements et se déclenchent (*are triggered*) lorsque ces évenements adviennent. (Dans ce contexte, la fonction est appelée *callback*. En effet, la fonction sera invoquée, non de façon impérative à l'endroit où elle apparaît dans le code, mais plus tard, au moment où l'évènement se produit. En anglais : "*the event listener calls the function back*", d'ou le nom.) Les fonctions peuvent à leur tour déclencher de nouveaux événements personnalisés (`graphiqueMisAJour`, `donneesTelechargees`), auxquelles d'autres fonctions sont abonnées, et ainsi de suite. Cela permet de décomposer le code en petits composants, chacun spécialisé dans une action précise, **pouvant être exécutés simultanément**.

Remarque : JavaScript est un langage au *typage* très (trop) permissif, encore plus permicif que R. Par exemple avec `a=1` et `b="1"` on a `a==b -> true` *mais* `a+b -> 11` ! Cela génère de nombreux bugs.

Exercices

J'ai implémenté pour vous une fonction qui charge une image quand on appuye sur le bouton "Graphique aléatoire", et revient à l'image de départ avec le bouton "Réinitialiser" [2]. Le code du jsfiddle (https://jsfiddle.net/katossky/ogfeqc8d) est reproduit ci-dessous :

```
// récupération des éléments du DOM
var graphic = document.getElementById("graphic"),
var buttons = document.getElementsByTagName("button");
var reinitialize = document.getElementById("reinitialize");
let sidebar = document.getElementById("sidebar");
// var random       = document.getElementById("random");

// création d'un nouvel élément du DOM
var warning = document.createElement("p");
warning.id = 'warning';
warning.textContent = "Chargement en cours";
```

1. Cette partie est inspirée de la présentation de Florent Grélard.
2. Cet exemple ne fonctionne que dans quelques cas simples. S'assurer que tous les cas d'utilisation sont couverts demande généralement un code plus long et plus complexe.

```
// définitions des variables
var url_initial = graphic.src;
var url_current = graphic.src;
var urls_for_random_graphic = [
  "http://www.vackerunderbar.se/wp-content/uploads/2010/08/gapminder.png",
  "https://cdn.quotesgram.com/img/16/29/360148697-DSC8242hans-rosling.jpg",
  "https://i0.wp.com/econlife.com/wp-content/uploads/2017/02/Gapminder_World-1.jpg",
  "http://www.stashmedia.tv/wp-content/uploads/2010/12/rosling.png"
];

// définition des fonctions
function load_image(event) {

  var button = event.target;

  if (button.id == "reinitialize") {

    url_current = url_initial;

  } else if (button.id == "random") {

    var urls = urls_for_random_graphic.filter(
      url => url != url_current
    );
    url_current = urls[Math.floor(Math.random() * urls.length)];
  }

  /* charger une image peut prendre du temps
  nous simulons ça par une attente de 2s
  avant de générer l'événement "image_ready" ;
  pendant ce temps, le reste du code continue à être exécuté */
  graphic.src = url_current;
  setTimeout(function() {
    var image_ready = new Event('image_ready');
    /* Le fait que l'image soit prête à être affichée
    ne concerne pas le bouton lui même! L'événement
    est domc généré directement au niveau
    du document tout entier. */
    document.dispatchEvent(image_ready);
  }, 2000); // 2000 ms = 2s
}

var add_warning = function() {
  sidebar.insertBefore(warning, null);
  // Le javascript standard requiert à la fois un élément parent ET un élément frère pour insérer un nou
}

function remove_warning() {
  sidebar.removeChild(warning);
}

// initialisation
reinitialize.addEventListener("click", load_image);
```

```
reinitialize.addEventListener("click", add_warning);
random.addEventListener("click", load_image);
random.addEventListener("click", add_warning);
document.addEventListener("image_ready", remove_warning);
```

Syntaxe

1. Comment déclare-t-on une variable en JavaScript ? (2 réponses)
2. Comment déclare-t-on une fonction ? (2 réponses) **Remarque :** les fonctions déclarées avec le mot clé `function` peuvent être utilisées *avant* (!) d'avoir été déclarées, un comportement connu sous le nom de *hoisting* (litt. treuillage).
3. Quelle est la syntaxe pour une fonction anonyme ? (2 réponses)
4. Comment signale-t-on des commentaires ?
5. Repérez la variable de type liste/tableau (`array`). Comment accède-t-on à un élément de la liste ? À une propriété comme la longueur de la liste ?
6. Comprenez vous la syntxe de la m´thode `filter` ?
7. Créez une liste qui contienne une description sommaire de chaque image.

Détails de la syntaxe de JavaScript ici.

Manipulation du DOM

8. À quoi correspond l'objet `document` ?
9. Comment récupère-t-on l'objet du DOM via son identifiant ?
10. Comment modifie-t-on l'attribut `src` de l'objet `img` du DOM ? Modifiez le code pour modifier l'attribue `alt` de chaque image à partir de la liste crée à la question (7). (`alt` donne un texte à afficher pour les malvoyants ou en cas d'échec du téléchargement de l'image.)
11. Repérez les fonctions utilisées créer, ajouter et supprimer un élément du DOM
12. Modifiez l'apparence de l'objet `warning`. (La propriété CSS `p` d'un élément du DOM `e` est accessible en lecture et en écriture par `e.style.p`. Voir ici pour une liste complète.)

Événements

13. Comment indique-t-on qu'une fonction `f` doit être activée à chaque fois que l'événement de type `"evt"` se passe au niveau de l'élément du DOM `elt` ? (On parle d'**abonnement**, et `f` est appelée le ***callback***.)
14. Si la fonction `f` accepte un argument, elle reçoit automatiquement un objet repésentant l'événement. Dans la fonction `load_image`, à quoi correspond `event.target` à votre avis ?
15. Il existe de nombreux événement prédéfinis. (Voir ici pour une liste complète.) Comment génère-t-on un événement personalisé comme `image_ready` ?
16. Faire un schéma qui représente l'enchaînement des déclenchements-réactions de ce code.
17. Rajouter 2 fonctions `image_blur` et `image_unblur` pour rendre l'image semi-transparente (propriété CSS `opacity`, voir ici) tant que celle-ci n'est pas complètement chargée. Abonner ces fonctions aux événements adéquats.

Correction : https://jsfiddle.net/katossky/f4ghvrc1