IHM
Interface Homme Machine

David Dupont

Rubika – 2024/2025

Objectif du module

- Savoir comment construire une IHM statique
- 3 jours ensemble
  - 13 octobre
    - Présentation des concepts
    - Exercices
  - 19 novembre
    - Lancement sur un projet
  - 28 novembre
    - Finalisation du projet
    - Soutenance évaluée

Plan

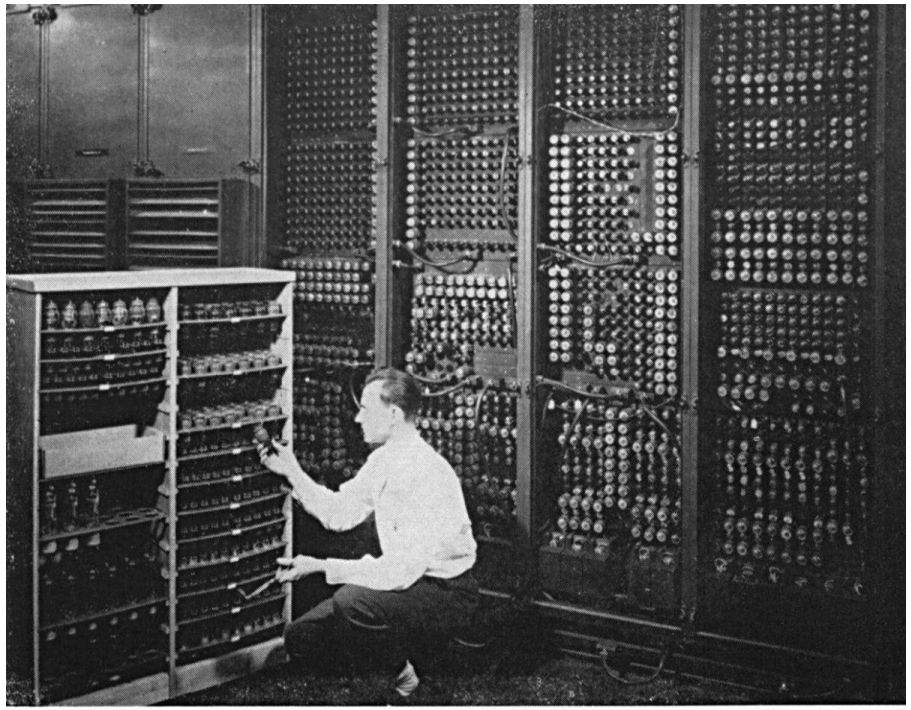
- Intro
  - Historique
  - Intérêt
- HTML
- CSS
- Javascript
- Principes ergonomiques

Intro
Une IHM c'est quoi?

- Interface Homme Machine
- Ensemble d'outils permettant à l'homme de communiquer avec une machine (site web, applications mobiles, JV…)
- Les principaux aspects des IHM: Ergonomie, Accessibilité, Design, UX, Feedback
- Il ne s'agit pas que d'un travail de développeur, c'est un domaine multidisciplinaire (optimisation, art, psychologie…)
- On verra une partie de ces concepts dans la partie ergonomie ☺

Historique

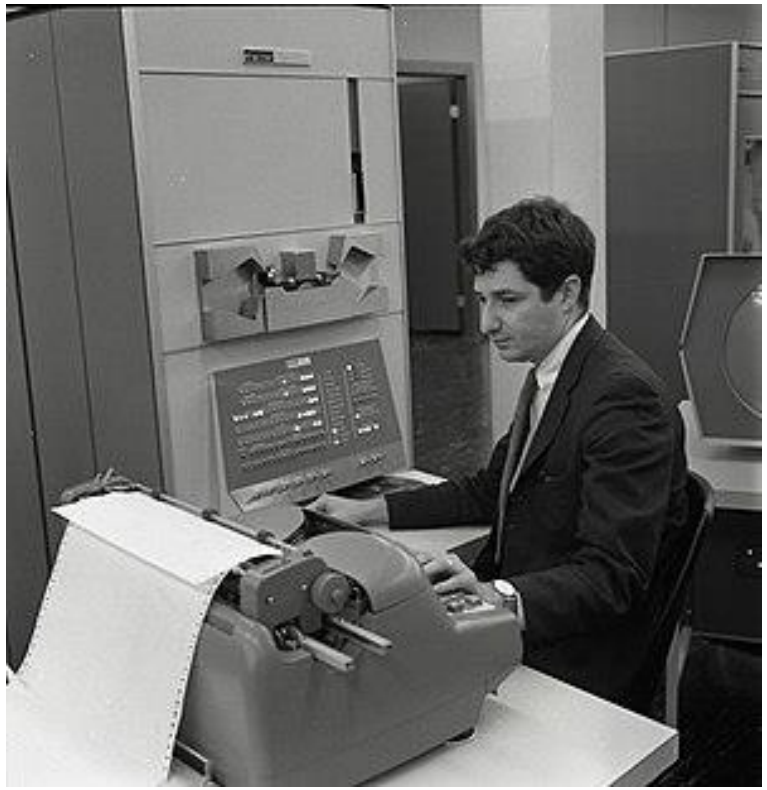
- Tout part de la création des ordinateurs dans les années 1940
- Les ordinateurs étaient gérés par des câbles et des interrupteurs pour configurer les calculs qu'ils devaient effectuer
- Les données envoyées aux ordinateurs étaient sur des cartes perforées et le résultat était imprimé
- Hééééé non, il n'y avait pas d'écrans 😱



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

Premières interactions avec le software

- Création des terminaux texte dans les années 60
- On parle ici d'entrer des commandes sur un clavier et magiquement ça procède aux calculs
- Déjà des systèmes conçus comme le time-sharing, qui permet de partager les ressources d'un ordinateur central



De plus en plus loin

- Premiers GUI créés dans les années 80: sortie du premier Macintosh, suivi de près par Windows
- Dans les années 90 : Avènement d'internet et naissance des premiers navigateurs web avec l'HTML, le CSS etc… Tout ce dont on va parler ☺
- À partir des années 2000, on a tout ce qu'on connait maintenant, les applis mobiles, les interfaces 3D, des expérimentations sur la VR etc…
- Et plus récemment la popularisation de l'IA, même si ça existe depuis un bon moment maintenant…

Les intérêts

- On comprend tous l'intérêt premier, on veut discuter avec les machines et leur faire faire plein de choses
- Apprendre à travers les ressources hébergées/IA
- Des algorithmes en profitant de leur puissance de calcul
- Communiquer (mail, messages, vocaux..)
- Améliorer notre productivité à travers de traitements automatiques de tâches
- Développer son activité pro avec des sites web

HTML

- HyperText Markup Language
- Fondement du www (World wide web) grâce à ses liens hypertextes
- Langage de balisage traduit par les navigateurs
- Première version en 1991 qui comprenait 18 balises
- Aujourd'hui on est à l'HTML5, qui prend en compte beaucoup plus de balises (115 balises non obsolètes)

Les balises

- Base de la création des pages HTML
- Permettent de structurer et formatter les données affichées
- Représentées par des crochets <> ou </>
- 2 types de balises:
  - Conteneur : <h1>MON TIIIIIIIITRE</h1>
  - Auto-fermantes:
- Les balises peuvent contenir des attributs: comme la balise img ci-dessus pour préciser la source de l'image
  - Autre Exemple: lien vers google
  - Permet d'ouvrir un autre onglet avec le target et d'aller directement sur google.com quand on clique dessus

Les blocs

- Il est possible que des balises contiennent d'autres balises (balises imbriquées)
- Certains éléments HTML se servent de ce fonctionnement pour mieux structurer la page.
- Exemple avec les balises : Cet élément permet de créer une section dans la page pouvant contenir d'autres balises. Cela permet de mettre en forme comme on le souhaite.

Ci-dessous, on voit qu'on a 3 div dans une div parente.

```
<div class="container">
    <div class="box">1</div>
    <div class="box">2</div>
    <div class="box">3</div>
</div>
```

Les principales balises

- Les fichiers HTML5 commencent par <!DOCTYPE html>
- Les balises structurant le fichier :
  - <html> : contient tout le contenu html de la page
  - <head> : contient les métadonnées (titre, styles, scripts..)
  - <body> : contient tout le contenu visible de la page
- Les balises courantes:
  - : c'est une balise conteneur qui va créer un section dans la page
  - <p>: paragraphe pour écrire du texte
  - <a> : liens hypertexte permettant de naviguer entre les pages
  - <img> : images
  - <form> avec <label> et <input>: pour faire des jolis formulaires

Les listes et les tableaux

- 2 types de listes
  - Non ordonnées :
  - Ordonnées :
- Ajouter un ligne ->
- <table> Pour les tableaux à plusieurs dimensions
  - <tr> pour ajouter une ligne
  - <th> pour ajouter une colonne
  - <td> pour ajouter une donnée
- Attention on utilise <th> pour les entêtes de colonne, écrit par défaut en gras à la différence de <td> pour les datas

liste ul

- ma
- super
- liste
- de
- valeur
- qui
- veulent
- rien
- dire

liste ol

1. ma
2. super
3. liste
4. de
5. valeur
6. qui
7. veulent
8. rien
9. dire

prénom	nom	place en F1
David	Dupont	1
Max	Verstappen	11

Les choix

faire un choix parmis vos pilotes de F1 favoris (y'aura pas Max dans la liste)

Quel est votre pilote préféré ?

- Charles Leclerc
- Lewis Hamilton
- Lando Norris

- La liste à choix unique : <select> et <option>
- Les boîte de sélection :
  - <input type=« radio »>
  - Pour sélectionner plusieurs éléments, il faut utiliser les inputs avec le type checkbox

    <input type=« checkbox »>

Quel est votre pilote préféré ?

- [x] Charles Leclerc
- [x] Lewis Hamilton
- [ ] Lando Norris

Plus de balises

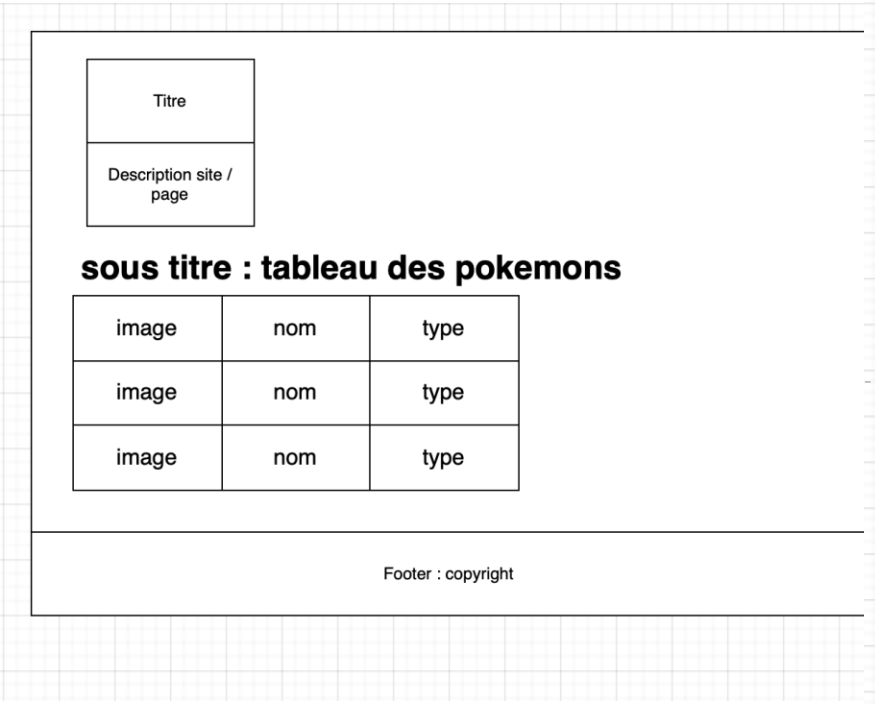
- Pour en voir plus au niveau des balises pour connaitre
  - Toutes les balises d'HTML5
  - Les attributs possibles
  - Les valeurs d'attribut
- Plusieurs documentations possible
  - Mozilla: https://developer.mozilla.org/fr/docs/Web/HTML/Element
  - Jaétheme (très complet aussi) : https://jaetheme.com/balises-html5

A vous de jouer!

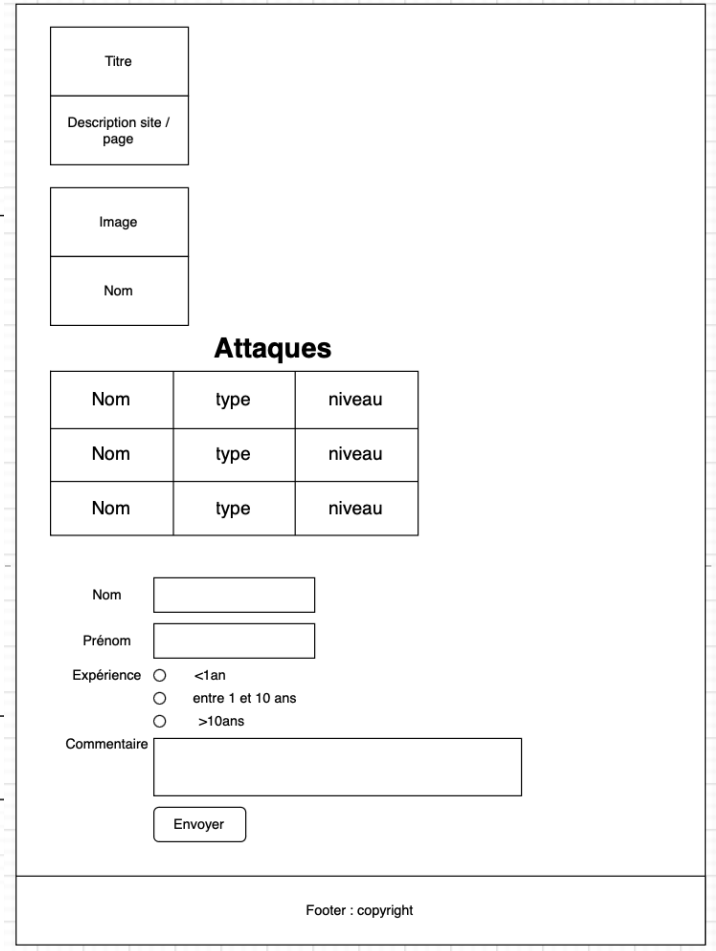
- Au fur et à mesure du cours, on va créer un site représentant un Pokédex (Vraie question: on parle encore des Pokémons?)
- Pour commencer, on va créer la structure du site sans réfléchir à l'apparence ni à l'interactivité (sans CSS ni JS)
- Nous allons créer 2 pages html, une listant quelques Pokémons dans une liste et une donnant le détail d'un seul
- Chaque Pokémon sera représenté par une image, un type, un nom dans la liste
- Quand on cliquera sur un Pokémon dans la liste, un nouvel onglet s'affichera avec la page de détail
- Dans le détail, on aura accès à ses attaques qui auront un type également et une puissance d'attaque
- Il y aura également dans la page de détail un formulaire, où on pourra inscrire un commentaire sur le Pokémon (mais qu'on ne pourra pas valider pour le moment), pour identifié la personne, on devra remplir son nom, prénom et son expérience Pokémon (<1an, entre 1 et 10 ans ou >10ans)
- Comme il n'y a pas d'interactivité, tous les Pokémons de la ligne auront le même détail, et c'est ok !
- Bien sûr, les pages auront un titre, une description du site et de la page actuelle et en bas, un copyright où apparaitra votre nom votre entreprise etc... (indice: utilisez un footer)

Idée de maquette

Page listing



Page de détail



CSS - Cascading style sheet

- Créé à la suite d'HTML pour ajouter du style au pages web dans les années 90
- Intégrée au W3C (World Wide Web Consortium) en 1996 avec le CSS1
- Le CSS2 est apparu en 1998 avec des nouveautés comme le système de positionnement des éléments HTML (position absolue, z-index..)
- Le CSS3 arrive en 2005 et est utilisé depuis. Il apporte les transitions/animations et notamment un système de média queries qui permet le responsivité du site. Il évolue de façon modulaire avec toujours plus de nouveautés : des variables css, encapsulation etc...

Faire appel à du css

- 3 méthodes:
  - Soit directement dans la balise HTML
    - <p style=« text-align:center; »>mon texte centré</p>
  - Soit dans les métadonnées (dans la balise <head>)
    - Avec une balise style

```
<head>
    <style>
        p{
            font-size: xx-large; /*met tous les paragraphes en taille très grande*/
        }
    </style>
</head>
```

    - Soit dans un fichier extérieur à l'HTML et on le référence dans les métadonnées

```
<head>
    <link rel="stylesheet" href="styles.css">
</head>
```

Identifier l'élément HTML à styliser

- Les éléments HTML peuvent être identifiés de plusieurs façons, les principales sont:
  - Type de balise
  - Classe
  - Id
- Exemple :
  - <p class=« text-bold » id=« listing-tab-title »>tableau des Pokémons</p>
- Par quelle approche il est préférable de donner un style à l 'exemple ci dessus?
  - Par le type de balise, tous les éléments p seront impactés
  - La classe peut être stylisée et a un impact sur tous les éléments l'utilisant, peu importe le type de balise
  - L'id est normalement unique à ce texte, donc seul cet élément sera impacté

Comment on modifie le style ?

- Format de fichier précis
- 3 choses pour définir un style
  - Le sélecteur
  - Bloc de déclaration
  - Association propriété : valeur
- Exemple :

```
a {
        color : red;
        font-family : Arial, Helvetica, sans-serif;
}
```

- Comme vu avant, le sélecteur peut désigner
  - Un type de balise, en ne précisant que la balise (exemple ci-dessus)
  - Une classe est référencée avec un point -> .text-bold
  - Un identifiant avec le Hashtag -> #listing-tab-title
- Mise en garde: un élément HTML peut être stylisé via un id, une classe et une balise en même temps, il faut faire attention aux styles contradictoires!

Multi-sélecteurs

- Un bloc css peut avoir plusieurs sélecteurs, c'est-à-dire qu'on peut donner le même style à plusieurs éléments HTML en un seul bloc
  - Exemple: p, h1, h2{

    color : red; /* ici, tous les paragraphes et les titres en h1 h2 seront écrits en rouge */

    }
- A l'inverse, un élément HTML peut aussi avoir plusieurs classes CSS associées
  - Exemple: <p class=« .couleur-rouge .texte-gras »>Lol</p>
  - Ici mon Lol prendra le style des 2 classes couleur-rouge et texte-gras
- Quelques bonnes pratiques pour ne pas se perdre dans les styles:
  - Regroupez les sélecteurs qui partagent vraiment le même style: permet en cas d'évolution de modifier le style qu'à un seul endroit
  - Essayez de ne pas rendre les blocs de styles trop complexes pour garder une bonne lisibilité
  - Ne pas regrouper les sélecteurs sans liens logiques (exemple: le titre de la page et un bouton d'un formulaire)

Quelques styles les plus utilisés

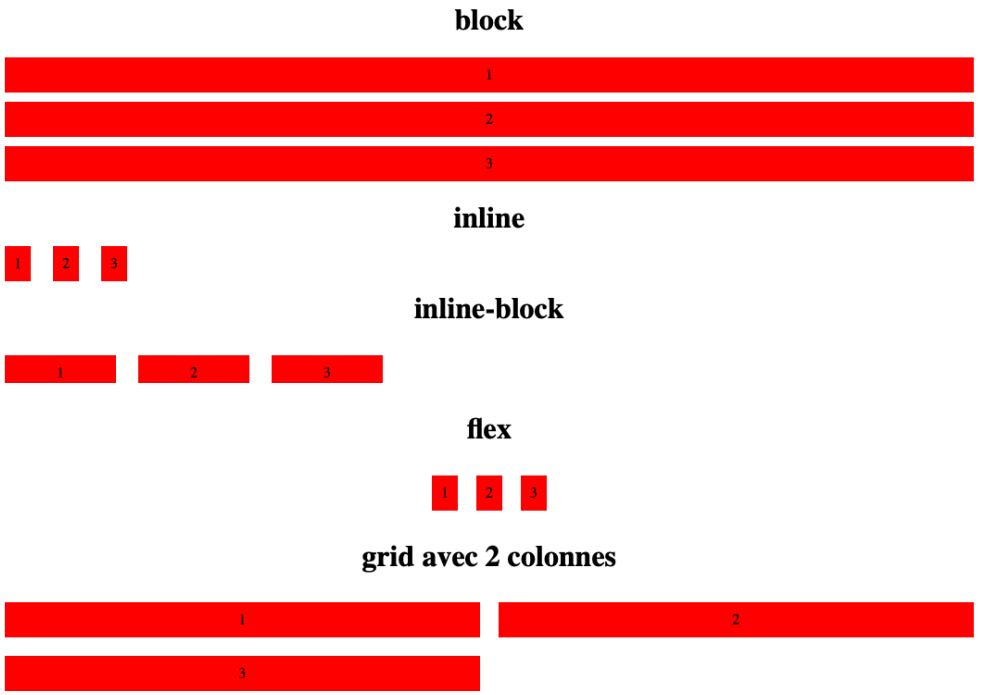
- Propriétés sur les dimensions:
  - Width : Largeur de l'élément
  - Height : Hauteur de l'élément
  - Margin: gère l'espacement extérieur de l'élément
  - Padding: gère l'espacement intérieur de l'élément
- Typographie d'un texte :
  - Font-size : taille du texte
  - Font-family : police
  - Text-align: alignement du texte
  - Text-decoration: déco du texte (souligné, barré…)
  - Color : couleur
- Propriétés sur l'arrière plan d'un élément
  - Background-color : couleur de l'arrière plan
  - Background-image : choisis l'image d'arrière plan à afficher
  - Background-size : gère la taille de l'image en arrière plan (étirement, répétition…)

Focus sur le positionnement des éléments

- Point super important dans un site web -> bien disposer ces informations pour que la vision et lecture soit le plus agréable au possible
- Utilisation généralement sur des
- Propriété display permet de gérer un ensemble d'élément
- Valeurs possibles:
  - display: block; -> prend toute la largeur de la div, les items se mettent les uns en dessous des autres
  - display: inline; -> aligne les items en ligne
  - display: inline-block; -> même que inline mais on peut régler la hauteur et la largeur des éléments
  - display: flex; -> aligne les items en lignes ou en colonnes selon la propriété flex-direction et align-items
- Z-index permet de gérer la profondeur de l'objet en cas de superposition
- Position gère le positionnement d'un élément par rapport à son conteneur parent (fixed, relative, absolute…)
- Associées à d'autres propriétés comme padding, margin qui gèrent l'espacement entre les éléments

Exemple visuel

```css
.container-block {
    display: block;
}
.container-flex{
    display: flex;
    justify-content: center;
}
.container-grid{
    display: grid;
    grid-template-columns: repeat(2, 1fr);
}
.box {
    background-color: red;
    margin: 10px;
    padding: 10px;
    text-align: center;
}
.box-inline {
    display: inline;
    background-color: red;
    margin: 10px;
    padding: 10px;
    text-align: center;
}
.box-inlineblock {
    display: inline-block;
    background-color: red;
    margin: 10px;
    padding: 10px;
    width: 100px;
    height: 10px;
    text-align: center;
}
```



Des styles interactifs

- Il est possible d'ajouter un style en fonction des actions de l'utilisateur
- Exemple: quand il passe sur un élément, un style se déclenche
- Ces pseudo-classes sont à préciser dans le sélecteur du css avec un « : »
- Exemple :

```
a:hover{
        color : red; /* met le lien en rouge lorsque la souris est dessus*/
}
```

- Les pseudos-classes courantes :
  - Hover -> s'applique quand la souris passe sur l'élément
  - Focus -> élément sélectionné ou qui a le focus sur la page
  - Active -> style s'appliquant lors du clic de l'utilisateur
  - Liste complète ici : https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes

Des transitions et des animations??

- Et oui c'est possible depuis HTML5!
- Les transitions permettent de changer graduellement le style d'un élément de façon simple
- Les animations sont plus complexes, et permettent de définir des séquences d'étapes dans le changement des éléments et de faire des boucles d'animations

Les transitions

- Se paramètrent lors de l'action de l'utilisateur sur un élément à travers les pseudo-classes(:hover, :active, etc...)
- Fluidifie la transformation de l'élément HTML
- Fonctionne en ajoutant la propriété transition dans le css
- Il doit forcément y avoir une action / transformation associée
- La doc sur la transition pour aller plus loin : https://developer.mozilla.org/en-US/docs/Web/CSS/transition
- Exemple

```css
.box{
  height: 100px;
  width : 100px;
  background-color: black;
  transition: .5s;
}

.box:hover{
  transform: translateX(20px) rotate(20deg);
  opacity : 0.5;
  background-color: green;
  height:50px;
  width:50px;
}
```

animations

- Utiles pour faires des changement d'états complexes
- On peut paramétrer l'état en fonction d'un pourcentage de l'animation
  - Exemple: à 0% (état initial), mon carré est rouge
  - À 50% : on met le carré en noir
  - À 100% : on le met en vert
- Mise en place de keyframes où l'on défini un état de départ et un état d'arrivée
- Dans le css de l'élément, on ajoute une propriété animation
- La doc : https://developer.mozilla.org/en-US/docs/Web/CSS/animation
- Exemple:

```
@keyframes spin {
  from {
    transform: rotate(0deg); }
  to {
    transform: rotate(360deg); } }

.box{
  height: 100px;
  width : 100px;
  background-color: black;
  animation: spin 1s infinite;
}
```

Quelques tips

- Attention, tous les navigateurs ne traduisent pas la même chose concernant le CSS, des outils en ligne peuvent vous aider
  - https://caniuse.com : montre quels navigateurs supportent quels propriété CSS
  - https://autoprefixer.github.io : génération du css spécifique pour chaque navigateur

A vous de jouer!

- Reprenez votre site Pokédex
- Le but ici est de bien positionner vos éléments sur la page
- Rendre le site plus joli en changeant la police d'écriture, mettre de la couleur dans certains encadrés, en arrière plan …
- Ajouter des transitions et des animations (exemple mettre une animation de chargement pour dire que la page est en cours de construction)

Javascript

- 3e partie du trio HTML/CSS/JS
- Langage objet et événementiel, on peut créer et utiliser des objets ayant des propriétés et des méthodes
- Comme l'HTML, c'est un langage interprété par les navigateurs et le code se charge à l'affichage des pages
- Que peut-on faire avec du JS?
  - Petites applis: calculettes, éditions de texte, jeu…
  - Du graphisme: modification d'images, gestion des fenêtres, modification dynamique de l'HTML
  - Tests de surface : validité des données saisies par un utilisateurs dans les formulaires…

Présentation du JS

- On a vu qu'il était possible de faire des choses spécifiques lors des actions de l'utilisateur:
  - Passage de souris
  - Clic
  - Saisie clavier
- À chacun de ces événements peut y être associé un code JavaScript
  - Exemple: lors du clic sur un bouton, une fonction onclick() peut être exécutée pour jouer un comportement spécifique
  - Autre exemple: au chargement de la page, une méthode onload() peut être appelée pour exécuter du JS avant l'affichage de la page

Structure d'un programme JS

- Structure qui se rapproche de la programmation en Python
- Pour déclarer une donnée
  - Var: pour les variables globales. Ces dernières sont accessibles partout dans le code
  - Let: pour les variables locales. Celles-ci ne sont utiliser qu'à l'intérieur des méthodes
  - Const : n'est pas une variable mais une constante
- Les variables sont aussi typées:
  - Nombres entiers ou à virgules flottante
  - Les booléens -> true ou false
  - Les caractères
  - Les chaînes de caractères
  - Comme en Python, il ne faut pas déclarer le type de la variable, le navigateur le détecte tout seul
- Il est conseillé de mettre un ';' à chaque fin de ligne
- Javascript utilise les accolades '{ }' pour délimiter un bloc de code
- Les commentaires :
  - // pour commenter une ligne
  - /* */ pour commenter un bloc de code

Les opérateurs

- Arithmétiques:
  - + : addition
  - - : soustraction
  - * : multiplication
  - / : division
  - % : modulo
- Opérateur de concaténation -> +
  - Exemple :

    let prenom = ''Max'';

    let nom = ''Verstappen'';

    let accueil = ''Bonjour'' + prenom + '' '' + nom;
  - Exercice :

    let w = 1 + 5 + 3;

    let x = 1 + '5' + 3;

    let y = 1 + 5 + '3';

    let z = '1' + 5 + 3;

    Que vaut w, x, y et z?
- Opérateurs logiques
  - && : ET
  - || : Ou
  - ! : NON
  - == : égalité
  - != : différent
- Pour aller plus loin sur les opérateurs: https://www.w3schools.com/jsref/jsref_operators.asp

Structures de contrôle

- Condition : si alors sinon
  - If / else classique
  - Switch case

```
let nom = "Verstappen";

if(nom === "Verstappen"){
    return "beurk";
}else{
    return "Oh Yeah (y)"
}

switch (nom){
    case "Verstappen" :
        return "Beurk"
    default :
        return "Oh Yeah (y)";
}
```

- Itération
  - For
  - For of
  - While
  - Do while

```
for(var i=0; i<10; i++){
  console.log("Lando will win")
}
var array = [1,2,3]
for (var x of array){
  console.log(x);
}
while (nom != "Verstappen"){
  console.log("Oh Yeah (y)");
}
do{
  console.log("première itération AVANT la
condition")
}while(nom === "Verstappen")
```

Les fonctions

- Les procédures, sans renvoi de données

```
// forme générale d'une procédure
function nom_de_la_procédure (paramètres) {
// des instructions ici
}
```

- Les fonctions, avec une donnée en retour

```
// forme générale d'une fonction
function nom_de_la_fonction (paramètres) {
  // des instructions ici
  return (résultat)
}
```

- Exemple

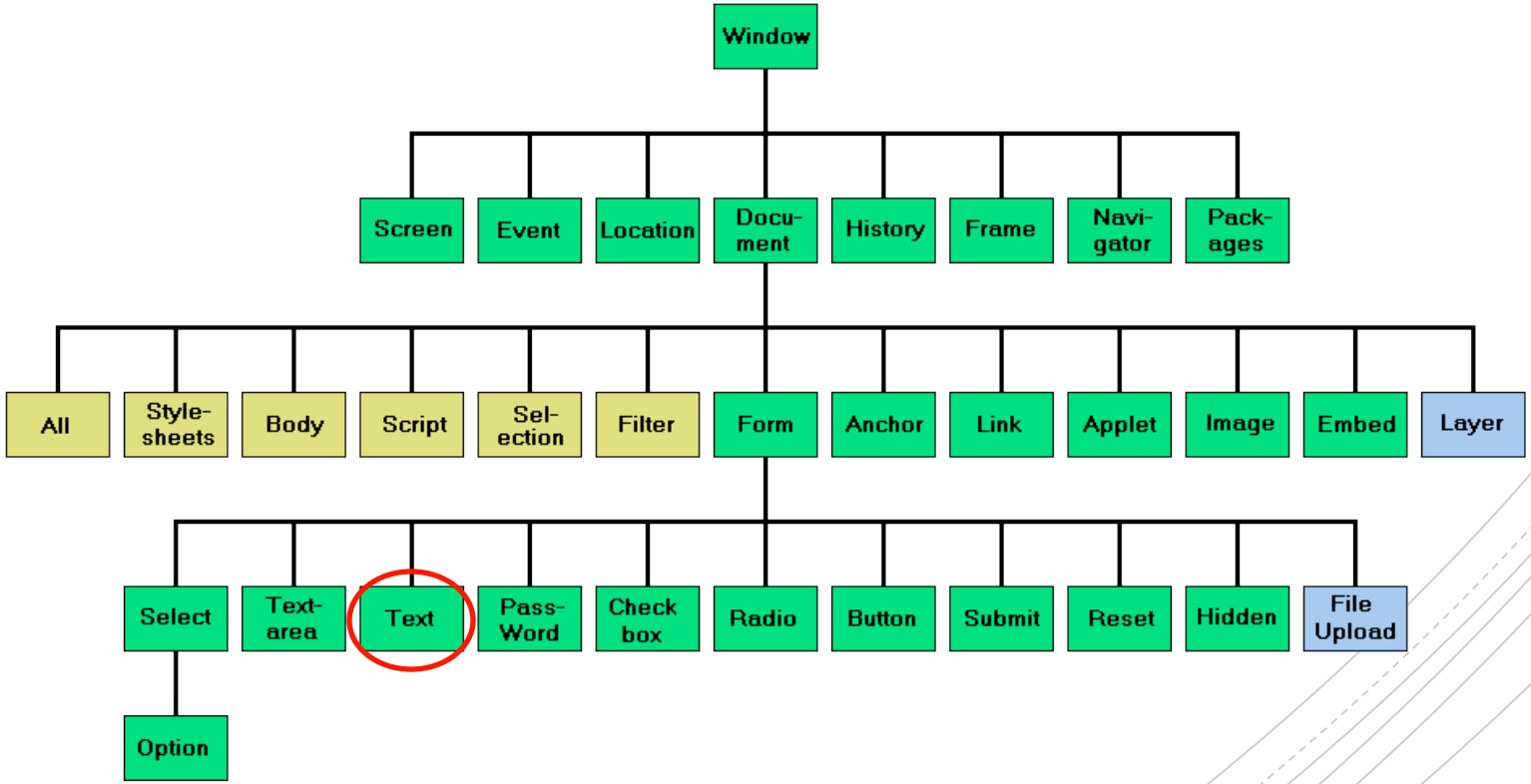
```
var mon_message = "Mon message";

// forme générale d'une procédure
function affiche_mon_message (messageEnEntrée) {
  window.document.write(messageEnEntrée);
}
affiche_mon_message(mon_message);

// forme générale d'une fonction
function addition (a, b) {
  // des instructions ici
  return (a+b);
}
```

JS, orienté objet?

- Les éléments manipulés par JavaScript et affichés dans votre navigateur sont des objets auxquels on associe des propriétés et des méthodes.
- En JS, quasiment tout est Objet:
  - String pour les chaînes de caractères
  - Boolean pour les booléens
  - Number pour les nombres
  - Array pour les tableaux
  - Date pour … Et oui les dates, bien joué ☺
  - RegExp pour les expressions régulières
- Exemple :
  - var v1 = 'je suis une valeur primitive';
  - var v2 = new String('je suis un objet');

Objets gérés de base

- JavaScript intègre d'origine plusieurs types d'objets.
- L'objet le plus grand est la fenêtre -> window
- Dans la fenêtre s'affiche une page -> document
- Pour accéder à l'élément « Text », on utilise la notation pointée window.document.form.text



Focus sur les chaînes de caractères

- Voici quelques commandes pour manipuler les chaînes de caractères -> Objet String
- Propriété length permet de montrer la longueur du la chaîne ‘’Max’’.length -> 3
- charAt(n) récupère nième caractère (Attention l’indice commence à 0): ‘’Max’’.charAt(1) -> a
- substring attend 2 paramètres : l'indice du premier caractère (inclus) et l'indice du dernier caractère (exclus): ‘’Rubika’’.subtring(0,4) -> Rubi
  - Si l’ordre est inversé (substring(4,0)), le JS rétablit l’ordre logique
  - Si on ne met qu’un nombre en entrée, la sous-chaîne commence à l’indice indiqué et se termine à la fin de la chaîne: ‘’Rubika’’.subtring(2) -> bika
- toUpperCase() pour mettre toute la chaîne en majuscule
- toLowerCase() pour la mettre en minuscule
- lastIndexOf() pour trouver la dernière occurrence d’une sous-chaîne:‘’Rubika’’.lastIndexOf(‘’ika‘’); -> 3
  - Retourne -1 si la sous-chaîne n’est pas trouvée

Manipulation des tableaux

- Initialisation d'un tableau
  - Var Tab = [ ];
  - t[0]=1; t[1]=''coucou''; t[2]=3.14;
  - var notes = [10, 5, 20, 14,2];
- Initialisation d'un objet Array
  - Var Tab1 = new Array();
  - Var Tab2 = new Array(3); //défini la taille du tableau
  - Var Tab3 = new Array(1, ''lol'', 3.45); //initialise le tableau en même temps que le créer
- L'aspect pratique est qu'on n'a pas de contrainte sur le contenu des cases
- Le premier indice commence à 0, donc la taille du tableau n s'arrête à l'indice n-1

Manipulation des tableaux (2)

- Il existe un système de tableau associatif (vous vous souvenez du polymorphisme? ☺)
- Exemple

```
var o = new Object();
o["un"] = 1;
o["deux"] = 2;
o["trois"] = 3;
```

  - Ici, o sera considéré comme un Array
  - Les indices ne seront pas des nombres, mais ‘’un’’, ‘’deux’’ et ‘’trois’’
- Plusieurs méthodes sont disponibles pour manipuler les éléments d’une liste:
  - Push()/pop(): ajoute/supprime un élément en fin de tableau
  - Unshift()/shift(): ajoute/supprime un élément en début de tableau
  - Sort(): trie le tableau
  - Reverse(): inverse les éléments d’un tableau
- Pour aller plus loin sur les tableaux: http://www.w3schools.com/jsref/jsref_obj_array.asp

Objet Date

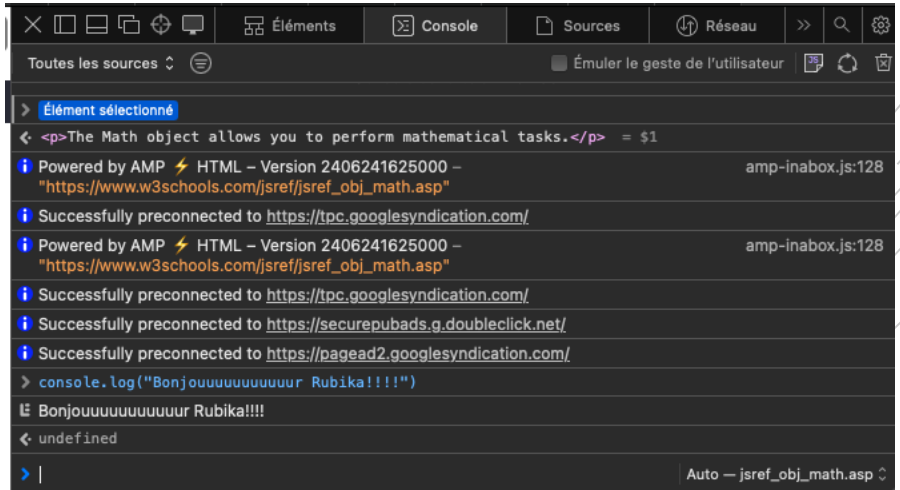
- Permet de gérer proprement des dates
- Initialisation :
  - var date_jour = new Date();
  - var une_date = new Date(annee, mois-1, jour, heure, min);
- Méthodes associées:
  - var now = new Date();
  - var annee = now.getFullYear();
  - var mois = now.getMonth() +1;//les mois vont de 0 à 11
  - var jour = now.getDate();
  - var heure = now.getHours();
  - var minute = now.getMinutes();
  - var seconde = now.getSeconds();

Objet Math

- La plupart des fonctions mathématiques sont des méthodes de l'objet Math.
- Permet de faire des arrondis sur des nombres:
  - Math.floor(x) : arrondit à l'entier inférieur ou égal à x
  - Math.round(x) : arrondit à l'entier le plus proche de x.
  - Math.ceil(x) : arrondit à l'entier supérieur ou égal à x
- Générer des nombres aléatoires:
  - Math.random() : génère un nombre aléatoire entre 0 et 1
  - Math.random()*100: entre 0 et 100
- Math.PI pour récupérer la valeur de PI
- Math.min() / Math.max() pour retrouver le plus petit/grand nombre d'une collection d'élément
- Plus d'infos : https://www.w3schools.com/jsref/jsref_obj_math.asp

L’outil indispensable

- console.log()
- Pour tester ou débugger vos scripts
- Permet d'afficher un message, une chaîne de texte, voire le contenu d'une variable (objet, tableau) de manière détaillée.
- Souvent dans un menu orienté pour les développeurs
- Accessible généralement avec un f12 sur le navigateur, ou en faisant clic droit + « inspecter l’élément »
- On fera un focus plus tard sur cet écran



Pour vous entrainer

- https://www.w3schools.com/js/tryit.asp?filename=tryjs_events
- Quelques exercice en JS pour que vous manipulez ☺
- 1- Écrire une procédure qui demande une chaîne de caractères et un entier à l'utilisateur et affiche « Je m'appelle prénom et j'ai âge ans ».
- 2 - Ecrire un algorithme permettant d'afficher le mois en lettre selon le numéro saisi au clavier. (Si l'utilisateur tape 1 le programme affiche janvier, si 2 affiche février, si 3 affiche mars…)
- 3 - Écrire un algorithme permettant de calculer la somme S=1+2+3+…+N, où N est saisi par l'utilisateur. Utilisez la boucle While
- 4 - Écrire un algorithme permettant de chercher la valeur maximale contenue dans un tableau
- 5 – Écrire une procédure qui compte le nombre d'occurrences d'un caractère dans une phrase

HTML et JS

- Intégration de JS dans un fichier HTML :
  - Le code Javascript est écrit directement dans le fichier HTML entouré d'une balise <script>
  - Le JS est contenu dans un fichier extérieur et est référencé dans la balise <head> de l'HTML avec cette balise <script src='fichier.js'></script>

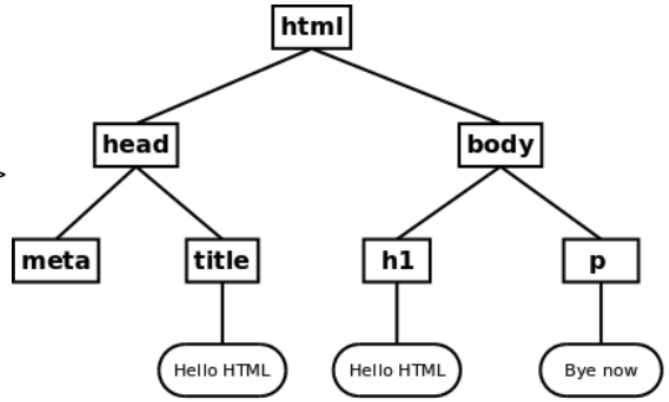
Le DOM

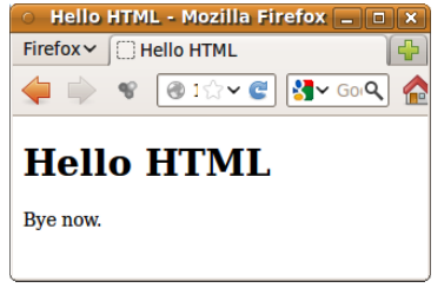
- Les navigateurs ont en mémoire un modèle de document, communément appelé le DOM (Document Object Model)
- C'est l'outil permettant l'accès aux éléments HTML
- Il permet deux choses aux développeurs
  - Fourni une représentation structurée du document. C'est cette structure que nous voulons modifier
  - Il codifie la manière dont un script peut accéder à cette structure, DOM est une API nous donnant les moyens d'interagir avec l'arbre-document.
- DOM désigne à la fois l'objet qu'on veut modifier et la boîte à outil qui va nous permettre de le faire

DOM (2)

- Le DOM est créé par le navigateur au moment du chargement de la page
- En HTML, tout est considéré comme un nœud:
  - Le document lui-même
  - Les éléments HTML
  - Les attributs
  - Le texte à l'intérieur des éléments
  - Etc…
- Les nœuds sont classés en 3 différents types:
  - Les nœuds-éléments
  - Les nœuds-attributs
  - Les nœuds-texte

```
1  <!doctype html>
2  <html lang="fr">
3  <head>
4    <meta charset="utf-8">
5    <title>Hello HTML</title>
6  </head>
7  <body>
8    <h1>Hello HTML</h1>
9    <p>Bye now.</p>
10 </body>
11 </html>
```





DOM (3)

- Aujourd'hui, le DOM est un standard et a été unifié : tous les navigateurs récents vont créer le même DOM à partir d'une même page et nous allons pouvoir utiliser les mêmes propriétés et méthodes.
- Ok c'est cool le DOM, mais concrètement, on peut faire quoi avec?
  - rendre visible/invisible une partie du document
  - modifier un élément de style de la page
  - changer une image
  - ajouter du contenu à la page
  - remplir automatiquement des formulaires ou tester la validité des données
  - saisies dans un formulaire avant l'envoi au serveur
  - etc...
- On a un contrôle total de la page HTML grâce à lui

Arborescence du DOM

- On a vu que tout les composantes du DOM étaient des nœuds
- Il existe différentes propriétés pour donner accès aux informations des nœuds:
  - attributes : Attributs du nœud
  - nodeName: le nom de la balise du nœud
  - nodeType: type du nœud (voir les différentes valeurs ici : https://developer.mozilla.org/fr/docs/Web/API/Node/nodeType)
  - nodeValue : Valeur de la balise (peut être nulle)
- Exemples:
  - <h1 class=''centre'' id=''test''>Bonjour</h1>

Exemple Arborescence

```
<!DOCTYPE html>
<html>
<head>
<script>
function displayDate() {
    p=document.getElementById("test");
    console.log(p.nodeName);
    console.log(p.nodeValue);
    console.log(p.nodeType);
    console.log(p.firstChild);
console.log(p.firstChild.nodeValue);
}
p=document.getElementById("test");
console.log(p.nodeName)



</script>
</head>
<body>

<h1 class="centre" id="test">Bonjour</h1>

<button type="button" onclick="displayDate()">Display Date</button>

</body>
</html>
```

Manipuler les éléments à travers le DOM

- L'une des grandes forces du JavaScript est de manipuler des éléments HTML en se servant du DOM et de ses méthodes et propriétés.
- Pour manipuler un élément, il faut d'abord pouvoir y accéder.
- Pour accéder aux éléments, il faut toujours passer par le document

getElementById()	Returns the element that has the ID attribute with the specified value
getElementsByClassName()	Returns an HTMLCollection containing all elements with the specified class name
getElementsByName()	Returns an live NodeList containing all elements with the specified name
getElementsByTagName()	Returns an HTMLCollection containing all elements with the specified tag name

querySelector()	Returns the first element that matches a specified CSS selector(s) in the document
querySelectorAll()	Returns a static NodeList containing all elements that matches a specified CSS selector(s) in the document

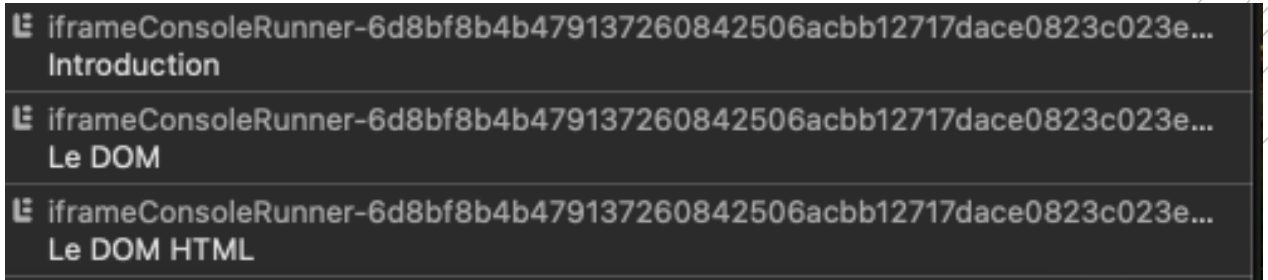
- Plus d'infos sur le DOM ici : http://www.w3schools.com/js/js_htmldom.asp

Modifier le contenu d'un élément

- La propriété innerHTML permet 2 choses
  - permet de récupérer le code HTML enfant d'un élément sous forme de texte. Ainsi, si des balises sont présentes, innerHTML les retournera sous forme de texte.
  - Servir à modifier le contenu d'un élément HTML. Pour cela, il suffit d'utiliser innerHTML sur un élément et de lui affecter une nouvelle valeur textuelle.
- D'autres propriétés peuvent être utiles
  - innerText : change ou retourne le contenu d'un nœud et de ses descendants en respectant le style CSS
  - textContent : même chose que innerTexte mais en ne respectant pas le CSS (plus de perf mais texte brut)
- Exemple

```
<h1 id="intro"> Introduction </h1>
```

```
1  var e = document.getElementById("intro");
2  console.log(e.innerHTML);
3  e.innerHTML = "Le DOM";
4  console.log(e.innerHTML);
5  e.innerHTML += " HTML";
6  console.log(e.innerHTML);
```



Modifier les attributs d'un élément

- Il est possible également de modifier les attributs des éléments HTML
- 2 manières existent:
  - En tapant l'attribut directement
  - En passant par les méthodes get/setAttribute( )
- Attention si on veut modifier la « class » d'un élément, le mot est réservé en javascript, il faut utiliser className
- Exemple

```
HTML
1 <a href="http://www.telecom-lille.fr"> IMT </a>
```

```
JS
1 var lien =
  document.querySelector("a");
2 lien.href = "http://www.imt-
  nord-europe.fr";
```

- 2e façon:

```
HTML
1 <a href="http://www.telecom-lille.fr"> IMT </a>
```

```
JS
1 var lien = document.querySelector("a");
  lien.setAttribute("href","http://www.imt-
  nord-europe.fr");
```

Modifier le CSS d'un élément

- Pour ajouter ou modifier le style CSS d'un élément HTML, on va utiliser la propriété style de l'objet Element suivie de la propriété CSS à ajouter / modifier.
- Attention, en passant par le JS, la syntaxe des attributs CSS passent du kebab-case au lowerCamelCase (ex: font-size devient fontSize)
- Plus d'infos ici : http://www.w3schools.com/jsref/dom_obj_style.asp
- Exemple:

HTML

```
1
2 <a href="http://www.telecom-lille.fr"> IMT
  </a>
```

CSS

JS

```
1 var lien = document.querySelector("a");
2 lien.style.color="green";
3 lien.style.backgroundColor="red";
```

Ajouter du contenu à un nœud

- Avec le DOM, il est aussi tout à fait possible d'ajouter un élément HTML
- Cela se fait en 3 temps
  - Création de l'élément: peut se faire avec la méthode createElement( )
  - Affectation des attributs et du texte: comme vu avant avec la méthode setAttribute( ) ou avec la bonne propriété. Pour ajouter du texte, on peut utiliser createTextNode( ), ça créer un nouveau nœud de type Text
  - Association du nœud créé à l'élément parent dans la page. La méthode appendChild( ) va insérer ce nœud en tant que dernier enfant d'un autre élement. Une méthode insertBefore est aussi disponible pour ajouter associer un élément AVANT un autre
- Une solution un peu plus « bourrine » existe en écrivant un morceau de code HTML dans une variable et le rajouter à la propriété innerHTML de l'élément parent
  - Pas forcément la manière la plus propre, mais vous savez que c'est possible ☺

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Ma première page web</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div>

  </div>


</body>
<script src="scripts.js"></script>
</html>
```

```js
var div = document.querySelector("div");

var sousDiv = document.createElement("div");
var monTitre = document.createElement("h1");
monTitre.innerText = "j'ai écrit mon titre dans une sous-div";
monTitre.style.color="black";
sousDiv.appendChild(monTitre);
div.appendChild(sousDiv);
```

```js
var div = document.querySelector("div");
console.log(div)
div.innerHTML = "<div><h1 style=\"color:black;\">j'ai écrit mon titre dans une sous-div</h1><div>"
```

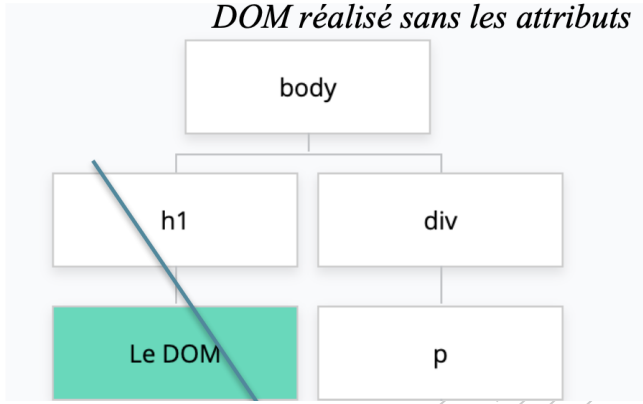
j'ai écrit mon titre dans une sous-div

Supprimer des éléments

- À Partir du DOM, on peut retirer complètement un élément HTML de l'arborescence, il sera donc perdu à jamais…
- La méthode pour effectuer ce traitement est removeChild( ), et prend en argument le nom de l'élément à retirer
- Exemple:

```
<body>
  <h1 id="titre"> Le DOM </h1>
  <div id = "myDiv">
     <p> Un peu de texte</p>
  </div>
</body>
```

```
var titre = document.getElementById("titre");
var parent = document.body;
parent.removeChild(titre);
```



Remplacer des éléments

- Pour modifier ou remplacer des nœuds / éléments HTML par d'autres, on va utiliser la méthode replaceChild().
- De ce fait il faut récupérer l'élément parent et l'enfant à remplacer
- Cette méthode va prendre deux arguments : la valeur de remplacement et le nœud qui doit être remplacé.
- Exemple

```
<h1 id="titre"> Le DOM </h1>
<div id = "myDiv">
  <p> Un peu de texte</p>
</div>
```

```
var titre = document.getElementById("titre");
var parent = document.body;
var nouveau = document.createElement('h2');
nouveau.innerHTML = "Nouveau titre";
nouveau.style.color="black"
parent.replaceChild(nouveau, titre);
```

Naviguer dans le DOM

- Comment accéder au nœud parent ou aux enfants d'un élément?
- Plusieurs propriétés d'un élément peuvent nous permettre de nous déplacer entre les noeuds

You can use the following node properties to navigate between nodes with JavaScript:

- `parentNode`
- `childNodes[nodenumber]`
- `firstChild`
- `lastChild`
- `nextSibling`
- `previousSibling`

Source : https://www.w3schools.com/js/js_htmldom_navigation.asp

Naviguer dans le DOM (2)

- La propriété parentNode va permettre d'accéder au nœud parent d'un certain nœud, et donc de se déplacer dans le DOM de la page HTML.
- childNodes, à l'inverse de parentNode, va permettre d'accéder aux nœuds enfants d'un certain nœud HTML. Cette propriété va renvoyer un tableau contenant les enfants d'un certain nœud.
- Remarque: Selon le DOM, un nœud ne contient pas de texte, il faut utiliser la propriété nodeValue en plus des childNodes si on veut le manipuler, ou utiliser la propriété innerHTML
- Remarque 2 : attention au fonctionnement de childNodes : les espaces dans le code (l'indentation par exemple) vont être considérés comme des nœuds textuels enfants !

Naviguer dans le DOM (3)

- firstChild et lastChild vont permettre d'accéder respectivement au premier et au dernier enfant d'un nœud.
- firstChild est équivalent à childNodes[0]
- Les propriétés nextSibling et previousSibling vont permettre d'accéder respectivement au nœud « frère » (c'est-à-dire de même niveau) suivant le nœud ciblé ou à celui précédent le nœud ciblé.

Évènements

- Les événements correspondent à des actions effectuées soit par un utilisateur, soit par le navigateur lui-même.
- Parfois, on va vouloir « attacher » une action spécifique à un événement.
- Pour faire cela, nous utilisons des attributs HTML de « type » événement, et ensuite en JavaScript nous créons le code correspondant à l'action que l'on souhaite attacher à notre évènement. Les attributs HTML de type événements sont nombreux.
- Pour exécuter du code au chargement de la page, la méthode onload va déclencher les évènements associés, par exemple on va chercher des données dans une bdd pour afficher la page

Évènements (2)

- Évènements associés aux clics et déplacements de la souris
- onclick : clic sur un élément
- onmousedown : appui sur le bouton de la souris
- onmouseup: relâchement du bouton de la souris
- onmousemove: déplacement du curseur sur l'élément
- onmouseout : sortie du curseur de l'élément
- onkeypress/onkeydown/onkeyup : événements provoqués par l'appui d'une touche au clavier
- Plus d'infos ici : http://www.w3schools.com/jsref/dom_obj_event.asp

Fonctions associées à un évènement

- JavaScript permet d'associer une instruction JavaScript à chaque événement.
  - Soit à l'appel à une fonction de base JS
    - Alert() : ouvre une pop-up avec un message
    - Window.location pour aller vers une autre page
    - Window.document.write pour écrire un message
    - setTimeout / setInterval pour différer et répéter une instruction
    - Autre méthode native..
  - Soit un appel à l'une de nos propres fonctions

Formulaires et JS

- Les formulaires sont la base des échanges entre le site et le visiteur.
- JavaScript ne peut pas échanger d'information à partir de fichier texte ou de base de données. L'interactivité avec l'utilisateur se fait via des zones de dialogues.
- Pour intégrer des éléments de formulaire, il faut encadrer les balises par <form> et </form>.
- Selon le choix de l'utilisateur, un traitement est associé aux zones de dialogue :
  - au niveau du serveur avec PHP (), Spring, Python…
  - au niveau du client avec JavaScript (assister et guider le visiteur, contrôler la saisie, faire des traitements (calcul, manipulation de chaînes de caractères), envoyer les résultats au serveur).

Accès aux éléments

- Le formulaire est un élément de l'objet document.
- Supposons le formulaire suivant

HTML

```html
<form name="general">
  <input type="text" name="champ1" value="Valeur init.">
</form>
```

- Pour accéder au formulaire, on peut écrire :
  - document.general
- Pour accéder à la zone de texte, on peut écrire :
  - document.general.champ1
- Ou on peut y accéder de la façon suivante:
  - document.forms[0]
  - document.form[0].elements[0]
- Pour changer la valeur d'un élément :
  - document.form[0].elements[0].value = ''NOUVEAU"

Évènement et formulaires

- Un événement est déclenché par le navigateur en réaction à une action de l'utilisateur ou d'un changement quelconque détecté.
- L'événement le plus classique est onclick.
- Exemple: plaçons "NOUVEAU" dans la zone de texte du formulaire à l'aide d'un bouton.

HTML

```
<form name="general">
<input type="text" name="champ1" value="Valeur init.">
<button type="button" onclick = "document.general.champ1.value='NOUVEAU'">
Changer</button>
</form>
```

L’Objet this

- Il est fastidieux d'accéder aux éléments de formulaire par toute la chaîne document.forms.elements
- Un objet JavaScript this permet de raccourcir ce chemin d'accès
- Pour rappel, en JavaScript, le mot clef this nous sert de référence à différents objets selon le contexte.
- Dans le contexte de la gestion d’événements, this va faire référence à l’objet (représentant l’élément HTML) qui est le sujet de l’événement.
- this représente donc l'objet JavaScript en cours d'utilisation

A vous de jouer!

- Reprenez votre projet Pokédex
- Dans le tableau de la page de listing, ajoutez une ligne en fin de tableau avec des entrées texte (input type text) et un bouton « Ajouter » à la fin de cette ligne
- Le but et de pouvoir écrire dans ces cases et le javascript va ajouter une ligne au tableau avec les données entrées par l'utilisateur
- Dans ce même tableau, ajoutez un bouton « supprimer » à chaque fin de ligne permettant de pouvoir retirer la ligne du tableau
- Pour le formulaire, finissez la fonctionnalité avec le JS pour pouvoir afficher le message au dessus du formulaire, bien sûr il faudra bien le présenter ☺
- Une fois qu'un message est posté, une alerte sera affichée pour
- Attention quand vous rechargez la page, c'est normal que vos modifications disparaissent, on verra ça avec le cours sur les frameworks!
- Ajoutez un bouton « modifier » au tableau permettant de modifier les données d'une ligne du tableau dans la page de listing
- Toute autre idée est le bienvenu :
  - un clic sur l'image du Pokémon fait apparaitre temporairement une boule disco en rotation
  - Si c'est une image Pikachu, des éclairs jaillissent de la photo..
  - Autres choses selon votre imagination

Principes ergonomiques

- L'ergonomie est la discipline qui vise à rendre un outil (produit, environnement, IHM) facilement utilisable par l'humain
- On cherche à rendre ces outils efficaces, confortables, sûrs et faciles à utiliser
- Il existe principalement 3 types d'ergonomie:
  - Physique: Il s'agit de concevoir ou ajuster des outils/équipements permettant à l'humain d'utiliser ses capacités physiques en manière efficace et sécurisée (exemple: chaise de bureau)
  - Cognitive: Ici, on créé des outils permettant de bien comprendre et traiter les informations venant de la machine pour réduire la charge mentale de l'humain (exemple: IHM)
  - Organisationnelle: Cherche à optimiser les tâches faites par l'humain pour améliorer la productivité et le bien-être en limitant le stress (exemple: l'IA)

Ergonomie des IHM

- Ici, on est clairement dans l'ergonomie cognitive
- Comment faire pour rendre les interactions entre le site et l'humain les plus fluides possible ?
- Il existe 9 principes ergonomiques importants
  - La simplicité
  - La cohérence
  - La rétroaction
  - Le contrôle utilisateur
  - La tolérance aux erreurs
  - L'efficacité et la rapidité
  - La souplesse et la personnalisation
  - La mémorisation facile

Simplicité

- L'interface doit être facile à comprendre et à utiliser. On arrive à respecter ces principes généralement en:
  - Comprenant les besoins de l'utilisateur, savoir pourquoi il est sur telle ou telle page, puis mettre les fonctionnalités importantes en évidence
  - Évitant la surcharge d'informations : Plus nous exposons d'informations sur le site, plus il est difficile de se repérer. Il faut savoir rester minimaliste
  - Utilisant des concepts familiers : bouton avec un icône d'avatar pour accéder aux informations du profil, un champ texte avec par défaut « rechercher » dedans pour montrer que c'est une fonctionnalité de recherche
  - Ayant un contenu facile à appréhender (bonne lisibilité) en évitant les informations inutiles (publicité qui clignotent sur le côté)

La cohérence

- La cohérence est essentielle dans l'expérience utilisateur d'un site web. Elle permet de réduire l'effort cognitif qu'un humain peut avoir en découvrant le site.
- Cette cohérence peut être mise en place avec ces actions:
  - Garder une cohérence visuelle, en maniant une même palette de couleur dans toutes les pages du site, la même police d'écriture, la même mise en page (marges, alignement …), les mêmes styles de boutons, de liens etc…
  - Garder une cohérence fonctionnelle : les boutons gardent le même comportement peut importe leur endroit dans le site, rendre prévisible les interactions possible (lien, boutons …)
  - Avoir une cohérence de navigation : garder le même menu de navigation dans toutes les pages et au même endroit. Les labels doivent rester les mêmes
  - Si le site est portable (web/ appli mobile/ autre), le style doit être le même peu importe la plateforme

Rétroaction

- La rétroaction (ou le feedback) permet de fournir des informations claires et immédiates à l'utilisateur en réponse à ses actions.
- Cela lui permet de comprendre que son interaction a été pris en compte et réduire son incertitude quant à son utilisation du site
- Les rétroactions peuvent être de différentes sources:
  - Visuelles comme des messages
  - Sonores comme le bruit du clavier quand on écrit un message
- Exemple de différents cas :
  - Un bouton s'anime quand on clique dessus
  - Une icône de chargement s'affiche quand un traitement long commence
  - Un message s'affiche quand on valide un formulaire
  - Des rétroactions préventives peuvent être pensées, comme désactiver un bouton tant que tous les champs d'un formulaire ne sont pas remplis

Contrôle utilisateur

- Le contrôle de l'utilisateur permet de le guider au travers des fonctionnalités en lui fournissant des outils pour qu'il se sente en contrôle sur les actions qu'il exécute
- Permet également de maîtriser au maximum les interactions qu'il a avec le site pour éviter les disfonctionnements et les blocage que l'utilisateur peut rencontrer
- Pour respecter et renforcer ce contrôle, voici quelques exemples à suivre:
  - Offrir des choix clairs : dans un tableau, on peut trier les données mais on laisse des choix limités (pour le pokédex, par nom ou par type)
  - Donner des options d'annulation ou de confirmation: dans un formulaire, on peut confirmer ou annuler la demande en étant redirigé vers l'accueil
  - Minimiser les actions automatiques: ne pas rediriger automatique vers un page sans l'accord de l'utilisateur

Tolérance aux erreurs

- L'objectif est de réduire le risque d'erreur et de permettre leur correction facile
- La meilleur manière de les limiter est de faire de la prévention avant qu'elles n'arrivent
  - Exemples: détection du format des numéros de tél. et des adresses mails, en écrivant un message d'erreur sur le site avant la confirmation du formulaire (concept fail fast)
  - Les messages d'erreurs doivent être compréhensibles et utiles: « le champ e-mail est obligatoire » ou « mot de passe non valide »
  - Annulation facile pour revenir sur une action via un bouton « annuler » ou « précédent »
  - Système de restauration ou de sauvegarde, pour revenir à un état antérieur des actions ou pour les reprendre plus tard

Efficacité et rapidité

- Ce principe joue extrêmement sur la satisfaction de l'utilisateur
- Ici nous parlons des temps de chargement des différentes pages mais également d'une navigation intuitive
- Le site doit être le plus optimisé possible, plus un site est long plus l'utilisateur aura tendance à fermer la page
- La règle des 3 clics: un utilisateur doit pouvoir trouver ce qu'il est venu chercher sur le site en 3 clics ou moins de manière simple et intuitive (bien sûr ce principe n'est pas applicable dans tous les contextes)
- Exemples de ce concept:
  - Optimiser le stockage des ressources utilisées par le site, moins de données stockées sur le serveur permet de réduire la latence de ce dernier -> Utiliser des formats d'images compressés ou stocker vos ressources sur une machine distante ou dans le cloud
  - Utiliser des menus bien structurés pour rendre la recherche d'information simple et rapide
  - Simplifier des processus en réduisant le nombre d'étape d'un formulaire, pré-remplissement des données etc..

La souplesse et la personnalisation

- Le but de ce concept est d'améliorer l'expérience utilisateur pour que l'interface soit complètement adaptée à ses préférences et ses besoins spécifiques
- On parle aussi d'accessibilité, pour que les personnes souffrant d'un handicap puissent consulter le site
- Exemples:
  - Laisser l'utilisateur choisir la langue, la taille du texte, utiliser une dictée vocale, proposer des sous titres pour les médias
  - Fournir plusieurs thèmes pour que l'utilisateur puisse s'adapter à la luminosité ambiante (fond light en pleine lumière, thème dark dans le lit)
  - Permettre aux utilisateurs de sauvegarder leur contenu préféré pour avoir un accès rapide (exemple : sauvegarde de produits dans un site e-commerce)

Mémorisation facile

- Ce concept vise à améliorer l'expérience utilisateur afin que ceux-ci puissent se souvenir aisément des éléments et des fonctionnalités du site même après une longue période d'absence
- Il s'agit d'un mix des concepts précédents afin de marquer l'esprit de l'utilisateur, donc si vous respectez les précédents, celui là viendra tout seul ! ☺
- Pour ce faire, on travaille surtout autour de
  - La simplicité et de l'aspect minimaliste de l'application en ne montrant que l'essentiel
  - La cohérence des éléments de l'interface (fonctionnels et visuels)
  - Des feedbacks et confirmations claires
  - Navigation intuitive avec des fils d'Ariane si nécessaires

Merci pour votre écoute

Vous avez des questions?

Pour me contacter

d.dupont@rubika-edu.com

david.dupont@imt-nord-europe.fr