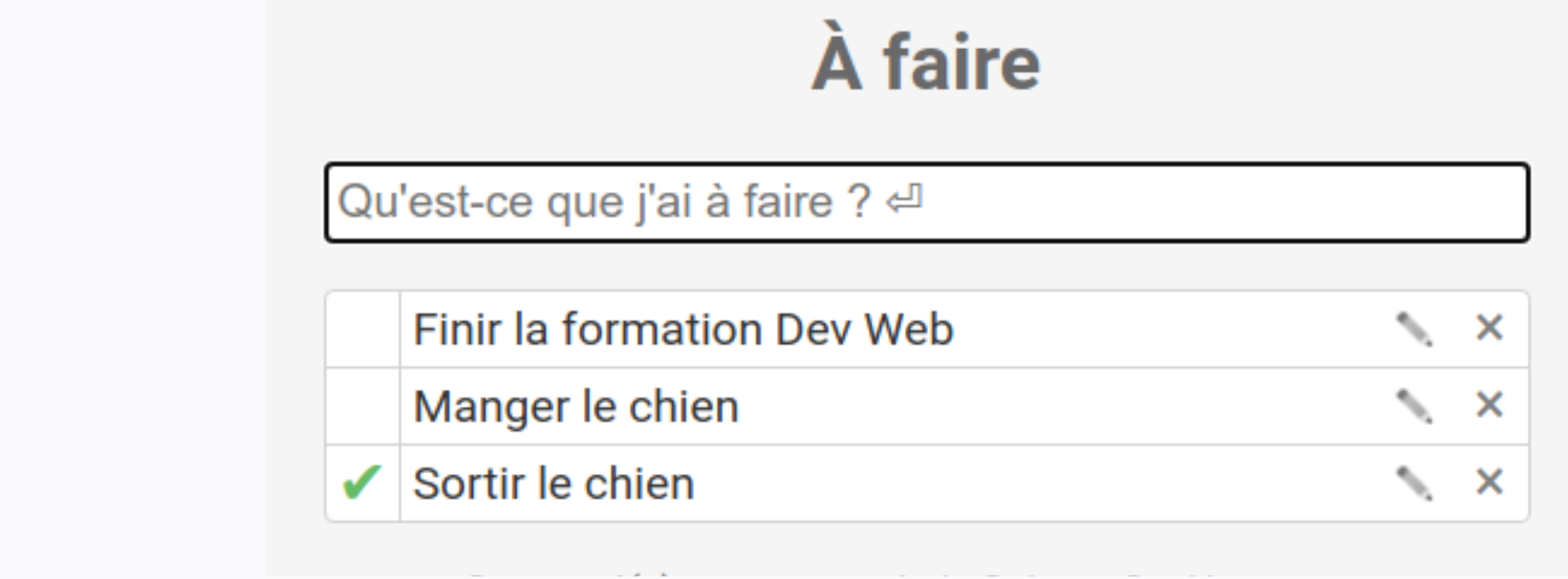
Formation Dev Web Front

Séance 2 : JavaScript

Ressources à télécharger sur url.viarezo.fr/front2

La situation

Une liste de tâches très belle… Mais pas très interactive !



→ La solution : Utiliser du **JavaScript** !

Ressources à télécharger sur url.viarezo.fr/front2

JavaScript (*JS*)

→ Langage de programmation (top 10 des plus utilisés !)

→ Scripts exécutés dans le navigateur, qui peuvent interagir avec la page

→ Typage dynamique, similaire à Python, mais en plus laxiste

→ Rien à voir avec Java !

Syntaxe – Types de valeurs

Comme sur Python :

Nombres (`10, 0.1, 1e3`), **Chaines de caractères** (`"hello, world!"`), **listes** (`[1.0, "un"]`)

Booléens : `true, false`
(Pas de majuscule !)

Valeurs spéciales : `undefined, null`

Objets (≈ dictionnaires) :

```
{
   clé: "valeur",
   "hissez haut": santiano,
   0: true
}
```

Syntaxe – Variables

Variable locale mutable :

```
let variable = 0;
```

Variable locale immutable :

```
const variable = 0;
```

(Il existe aussi une syntaxe avec 'var' et une syntaxe pour les variables globales)

Assignement :

```
variable = 0;
```

(Attention à avoir déclaré la variable !)

```
variable += 1; variable++;
```

Syntaxe – Opérations de base

Comme sur Python : Opérations numériques de bases (sauf //), appel de fonction, indexation de liste/objet, …

(Note : + fonctionne sur les nombres/strings, mais pas les listes)

Clé de type variable dans un objet : `objet.clé == objet["clé"]`

Existence d'une clé dans un **objet** : `"clé" in objet`

(Pas pour une liste !)

Opérations logiques : `! (non), && (et), || (ou)`

(Attention à doubler les symboles !)

Comparaisons : `==, ===, !=,!==, <, >=`

Syntaxe – Les conditions

```
// Commentaire uniligne
/* Commentaire
   multiligne */

if(ma condition) {
   instruction;
   instruction;
} else if(ma condition) {
   ...
} else {
   ...
}
```

Syntaxe – Les boucles

```
while(ma condition) { ... }

do { ... } while(ma condition);

for(let i = 0; i < 10; i++) { ... }

for(const clé in objet) { ... }

for(const élément of liste) { ... }
```

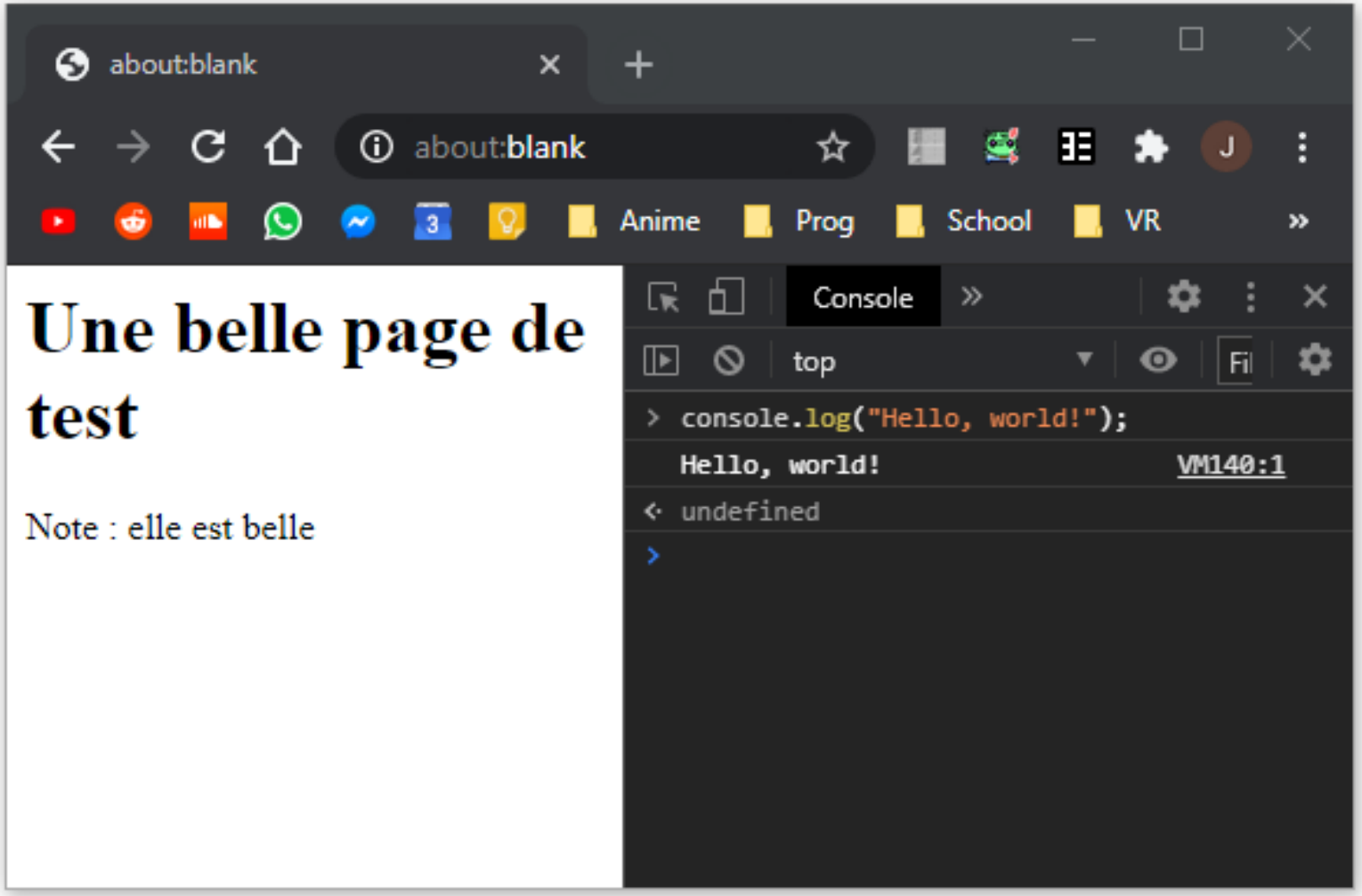
Syntaxe – Fonctions

```
const maFonction = (arg1, arg2) => {
   ...
   return valeur;
};
```

Ou :

```
function maFonction (arg1, arg2) {
   ...
   return valeur;
};
```

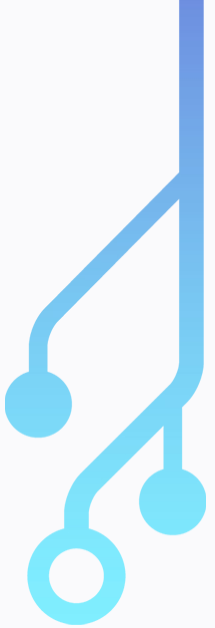
La console



Ctrl + Maj + J (Win/Linux)
⌘ + Option + J (MacOS)

Ctrl + Maj + K (Win/Linux)
⌘ + Option + K (MacOS)

Trucs utiles de la bibliothèque standard



(Référez-vous au PDF docu, ou à MDN (developer.mozilla.org))

- Array
- String
- Object
- Date
- JSON
- Math
- console
- parseFloat, parseInt

Le TD

Ecrire une fonction qui renvoie le nom de la personne la plus agée parmi une liste de personne.

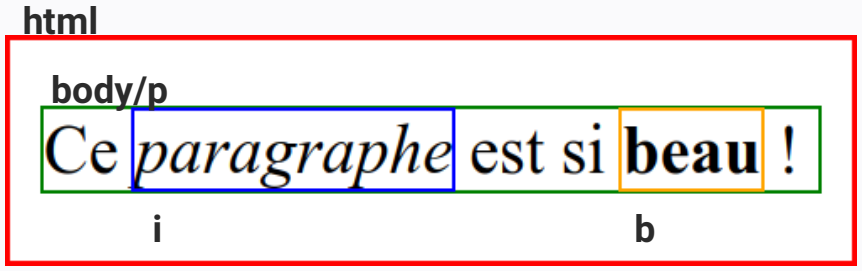
```
Personne =  {
                name: “Jean”,
                age: 19,
                }
```

Correction

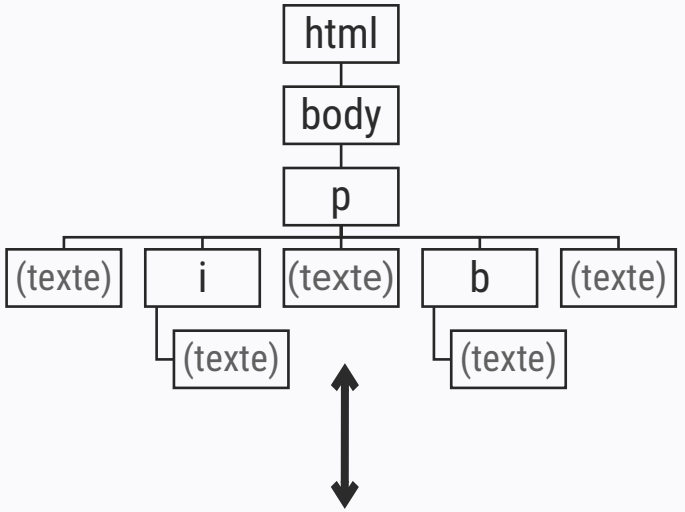
```
function older(l) {
    let max = l[0].age;
      let max_id = 0;
      for (const i in l) {
      if (l[i].age > max) {
        max = l[i].age;
              max_id = i;
      }
    }
      return l[max_id].name;
  }
```

Interaction avec la page

Rendu de la page



DOM



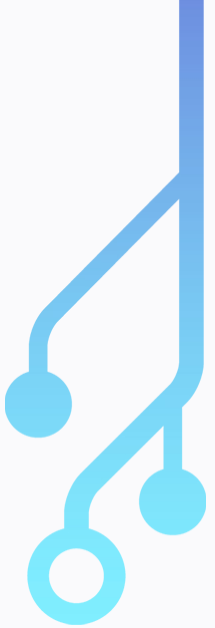
CSS

```css
p {
  color: red;
  font-size: 15px;
}
div {
  display: block;
  background: blue;
}
```

JS

(Chargé via un `<script src="script.js"></script>`)

Séléctionner un noeud du DOM



```
const element = document.querySelector(“.class”)
```

- **document** est la **racine** du document elle correspond à la balise **<html>**

- **querySelector**, **querySelectorAll** utilise des sélecteurs **CSS**

Manipuler un noeud du DOM

Quelques propriétés utiles :

`attributes`, `children`, `parentElement`, `classList`, `id`, `style`, `value` (pour les champs de formulaire)

Pour **ajouter un noeud** il faut d'abord le créer puis l'ajouter aux enfants du noeud désiré

```
const texte = document.createElement('p')
texte.innerText="texte"
body.appendChild(texte)
```

Les évènements

Pour réagir aux actions de l'utilisateur :

```
element.addEventListener("type d'évènement", event => {
   ...
});
```

Quelques types d'évènement utiles : `click`, `keydown`, `change`, …

Propriétés utiles sur l'objet élément : `key`, `target`, …

Exemple de manipulation du DOM

```
taskDelete.addEventListener('click', event => {
  const taskLi = event.target.parentElement;
  taskLi.remove();
});
```

Exemple de manipulation du DOM

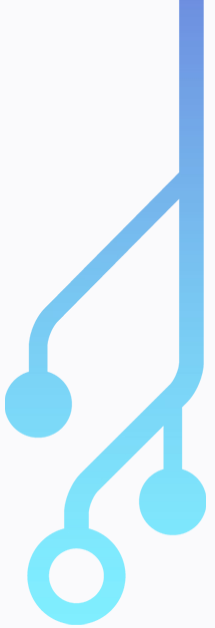
```
const taskDelete = taskLi.querySelector('.delete-button');
taskDelete.addEventListener('click', event => {
  const taskLi = event.target.parentElement;
  taskLi.remove();
});
```

Exemple de manipulation du DOM

```
const todoList = document.querySelector('.todo-list');

for(const taskLi of todoList.querySelectorAll('li.todo-item')) {
  const taskDelete = taskLi.querySelector('.delete-button');
  taskDelete.addEventListener('click', event => {
    const taskLi = event.target.parentElement;
    taskLi.remove();
  });
}
```

Ajouter un script à une page HTML



La balise **<script>** s'utilise de deux manière différente :

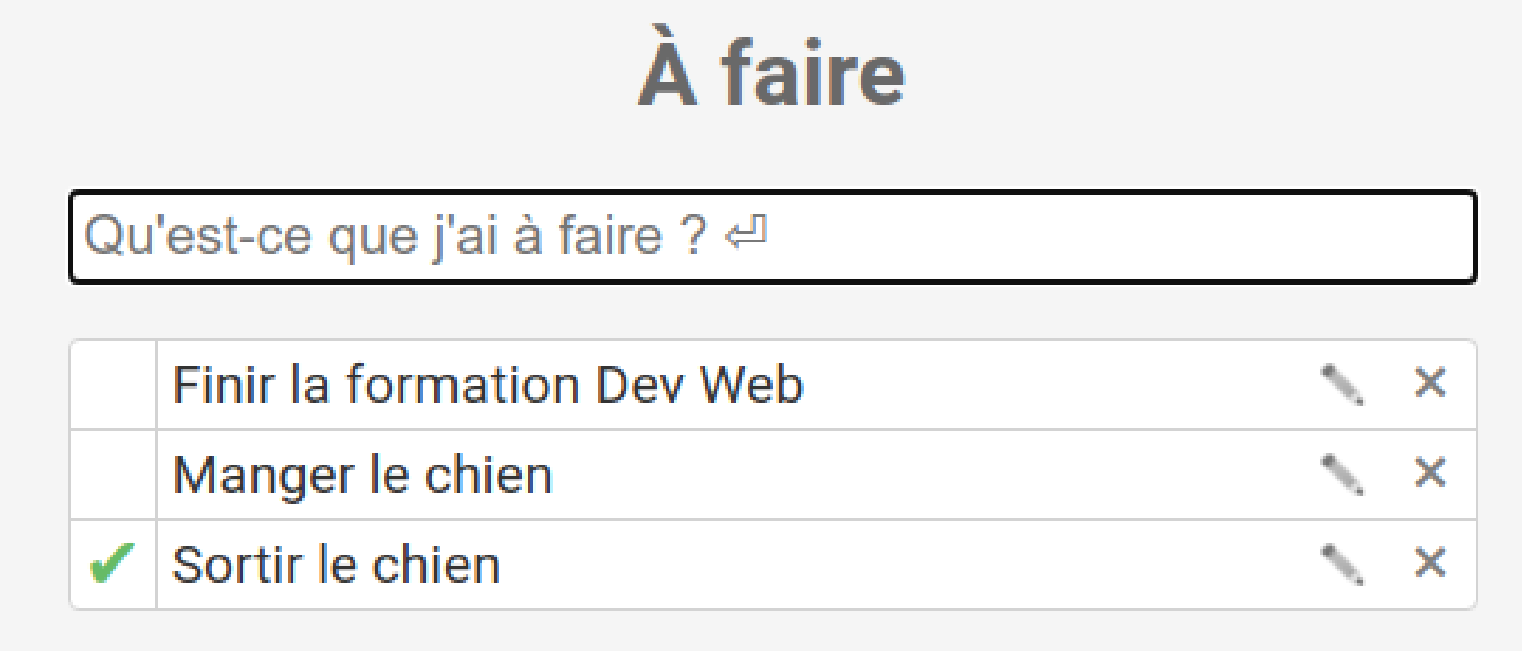
1) En écrivant directement du code **JS** à l'intérieur

2) En donnant à l'attibut **'src'** le lien d'un fichier en .js

Le TD !

Exercice : Modifier la liste de tâches du TD de la dernière fois (ou corrigé) de manière à pouvoir :

- éditer les items avec le bouton "crayon"
- rajouter des items en appuyant sur entrée dans le champ du haut



C'est la fin

On se revoit (j'espère) pour Back !