

A solid yellow square is centered on a light gray background. Inside the yellow square, the letters 'JS' are written in a bold, black, sans-serif font.

JS

Pourquoi apprendre le JavaScript

Langage de script comme PHP qui peut s'exécuter de deux façon

Coté navigateur :

- Créer des interactions
- Des animations
- Des SPA
- ...

Coté serveur :

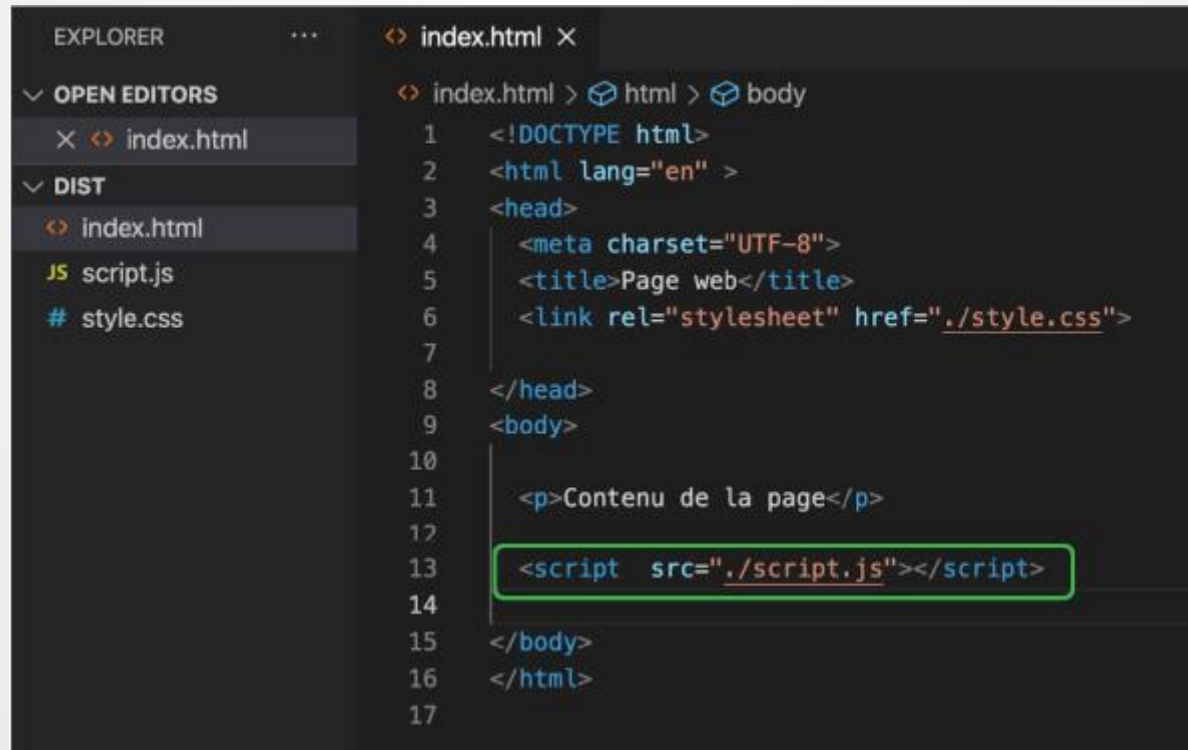
- NodeJS
- IO.JS

⚠ JavaScript != Java

Ou exécuter du code JavaScript

Sur internet : JSBin, JSFiddle, CodePen...

Sur votre navigateur web :



The screenshot shows a code editor interface with a dark theme. On the left, the 'EXPLORER' sidebar is visible, showing a file tree with 'index.html' selected. The main editor area displays the content of 'index.html'. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en" >
3 <head>
4   <meta charset="UTF-8">
5   <title>Page web</title>
6   <link rel="stylesheet" href="./style.css">
7
8 </head>
9 <body>
10
11   <p>Contenu de la page</p>
12
13   <script src="./script.js"></script>
14
15 </body>
16 </html>
17
```

The line containing the script tag, `<script src="./script.js"></script>`, is highlighted with a green rectangular border.

Les bases de JavaScript

Les variables

Extension de fichier : .js

Différents type de variable

Opération et concaténation

```
/* Initialisation d'une variable */  
let prenom;  
  
/* Affectation d'une variable */  
let age = 18;
```

La mutabilité des variables

```
let compteur = 0;  
compteur++;  
compteur = 10;
```

```
const nombrePostParPage = 20;
```

```
nombrePostParPage = 30; // Retournera une erreur dans la console
```

Exercice d'application

- 1 - Créez 4 variables : note1, note2... avec des valeurs comprises entre 0 et 20
- 2 – A l'aide des 4 variables, créez une variable moyenne qui calcul la moyenne des 4 notes que vous avez initialisées
- 3 – Enfin, affichez le résultat dans un console.log en concaténant avec :
« La moyenne de la classe est de ... »

Les objets

Notation en JSON

```
let movie = {  
  title: "Back To The Future",  
  director: "Robert Zemeckis",  
  year: "1985"  
}
```



```
let movie = {  
  title: "Back To The Future",  
  director: "Robert Zemeckis",  
  year: "1985"  
}  
  
console.log(movie.year); /* ou alert(movie.year) */  
  
/* OU */  
  
console.log(movie['year']);
```

Exercice d'application

1 – Créez un objet et stockez-le dans une variable appelée music. Utilisez des accolades et mettez les trois attributs suivants :

- Title : le titre de la musique
- group : Le groupe ou le chanteur de la musique
- Year : L'année de la musique
- Duration : La durée en seconde de la musique

2 – Affichez dans un console.log toutes les informations d'une musique en utilisant la notation dot.

Notation en classe

```
class Movie
{
    constructor(title, director, year)
    {
        this.title = title
        this.director = director
        this.year = year
    }
}
```

```
let myMovie = new Movie("Back To The Future", "Robert Zemeckis", "1985")  
  
console.log(myMovie)
```

Exercice d'application

- 1 – Refaite le premier exercice mais avec la notation classe
- 2 – En utilisant le mot clé `this`, assignez les propriétés via un constructeur
- 3 – Créez trois instance de la classe musique et affichez les dans un `console.log`

Les tableaux

Stockage de données (entier, booléen, tableaux, objet...)

```
let personnages = ["Luke Skywalker", "Ben Kenobi", "Han Solo"];  
  
console.log(personnages[0]) // Luke Skywalker
```

```
let movie = {  
  title: "Back To The Future",  
  director: "Robert Zemeckis",  
  year: "1985"  
}  
  
let allMovie = [movie]  
  
console.log(allMovie) // [{title: "Back To The Future", director: "Robert Zemeckis", year: "1985"}]
```

Fonction sur les tableaux

```
let personnages = ['Luke Skywalker', 'Ben Kenobi', 'Han Solo']

/* Pour connaître la taille d'un tableau */

personnages.length

/* Pour ajouter un élément à la fin d'un tableau */

personnages.push('Chewbacca')

/* Pour ajouter un élément au début d'un tableau */

personnages.unshift('Yoda')

/* Pour supprimer le dernier élément d'un tableau */

personnages.pop() // Supprimera Han Solo
```


Exercice d'application

- 1 – Créez un tableau comportant 5 couleurs
- 2 – Affichez une couleur présente dans le tableau grâce au `console.log`
- 3 –
 - i - Avec la méthode `push()`, ajoutez une nouvelle couleur à la fin
 - ii – Avec la méthode `unshift()`, ajoutez une nouvelle couleur au début
 - iii – Avec la méthode `pop()`, supprimez la dernière couleur
 - iv – Affichez la taille du tableau dans un `console.log`

Les conditions

```
if(condition)
{
    /* Code */
}
```

```
if(condition)
{
    /* Code */
}
else
{
    /*
        Si la condition en haut
        n'est pas respecter
    */
}
```

```
if(condition)
{
    /* Code */
}
else if(condition)
{
    /* Code */
}
else if(condition)
{
    /* Code */
}
else
{
    /* Code */
}
```

```
switch(variable)
{
    case 'value':
        // code...
        break;

    default:
        // code...
}
```

Les opérateurs logiques

And : &&

Affectation : =

Or : ||

Not : !

Différent de : !=

Supérieur à : >

Inférieur à : <

Supérieur ou égal : >=

Inférieur ou égal : <=

Égalité simple : ==

Égalité stricte : ===

Table de vérité de ET		
a	b	a ET b
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité de OU		
a	b	a OU b
0	0	0
0	1	1
1	0	1
1	1	1

Exercice d'application

- 1 – Créer une variable booléenne isConnected et initialisez la à true ou false
- 2 – Utilisez une condition pour afficher un message si l'utilisateur est connecté ou non

Exercice d'application

3 – Demandez à l'utilisateur ce qu'il veut regarder comme film (utilisez le prompt)

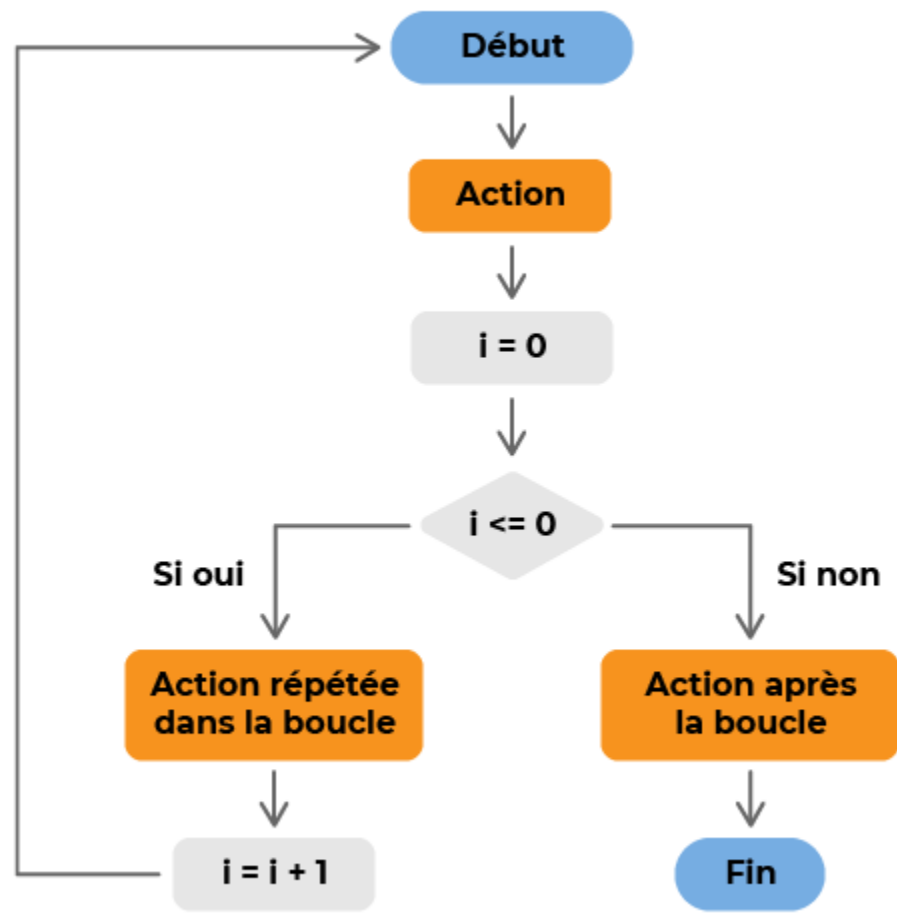
4 – Utilisez des conditions ou un switch pour afficher :

- « Tu veux regarder un film d'action » si l'utilisateur a rentré « Film d'action »
- « Tu veux regarder un film d'aventure » si l'utilisateur a rentré « Film d'aventure »
- « Tu veux regarder un film romantique » si l'utilisateur a rentré « Film romantique »
- « Je ne sais pas ce que tu veux regarder » si l'utilisateur rentre n'importe quoi

Les boucles

```
/** 1 - Initialisation */  
let variable  
  
/** 2 - Condition */  
while(variable <= 10)  
{  
    /* Code */  
  
    /** 3 - Incrémentation */  
    variable++  
}
```

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



La boucle *for... in*

```
const personnages = ["Luke Skywalker", "Ben Kenobi", "Han Solo"]

for(let i in personnages)
{
    console.log(personnages[i])
}
```

La boucle *for... of*

```
const personnages = ["Luke Skywalker", "Ben Kenobi", "Han Solo"]  
  
for(let personnage of personnages)  
{  
    console.log(personnage)  
}
```

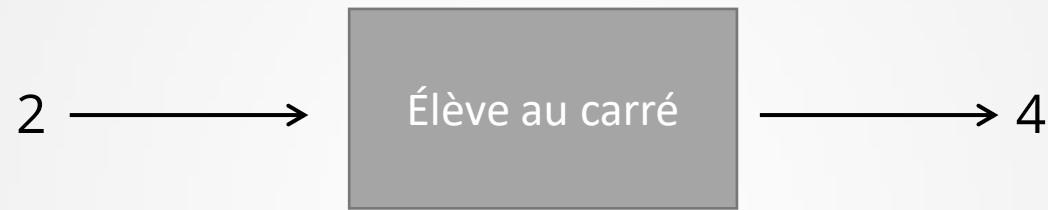
```
const personnages = [  
  {  
    nom: "Luke Skywalker",  
    arme: "Sabre laser"  
  },  
  {  
    nom: "Han Solo",  
    arme: "Blaster"  
  },  
  {  
    nom: "Chewbacca",  
    arme: "Arbalète"  
  }  
]  
  
for(let personnage of personnages)  
{  
  console.log("L'arme de " + personnage.nom + " est : " + personnage.arme)  
}
```

Exercice d'application

- 1 – Créez un tableau comportant 10 villes française
- 2 – Utilisez :
 - i – Une boucle for classique pour afficher le tableau
 - ii – Une boucle for...in
 - iii – Une boucle for...of
- 3 – Créez une boucle while avec une condition qui affiche tous les nombres qui sont multiples de 8

Les fonctions

Qu'est ce qu'une fonction ?



```
function function_name(parametre1, parametre2)
{
    // Code
}
```

Exercice d'application

1 – Créez une fonction qui calcul le volume d'une sphère. Elle prendra en paramètre le rayon de la sphère

Rappel de la formule pour calculer une sphère :

$$\frac{4\pi \times R^3}{3}$$

Et avec les objets ?

```
class Personnage
{
    constructor(name, category, weapon)
    {
        this.name = name
        this.category = category
        this.weapon = weapon
    }

    sayMyName()
    {
        console.log('My name is ' + this.name)
    }
}

let personnage = new Personnage("Luke Skywalker", "Jedi", "Lightsaber")

personnage.sayMyName() // My name is Luke Skywalker
```

Les méthodes statique : exemple avec l'objet Math

```
// crée un nombre aléatoire sur l'intervalle [0, 1]
const randomNumber = Math.random();

// arrondit vers le bas à l'entier le plus proche, renvoie 495
const roundMeDown = Math.floor(495.966);
```


DRY !

Exemple de factorisation d'une fonction

```
const getWordCount = (stringToTest) => {  
  const wordArray = stringToTest.split(" ");  
  return wordArray.length;  
}
```

Exercice d'application

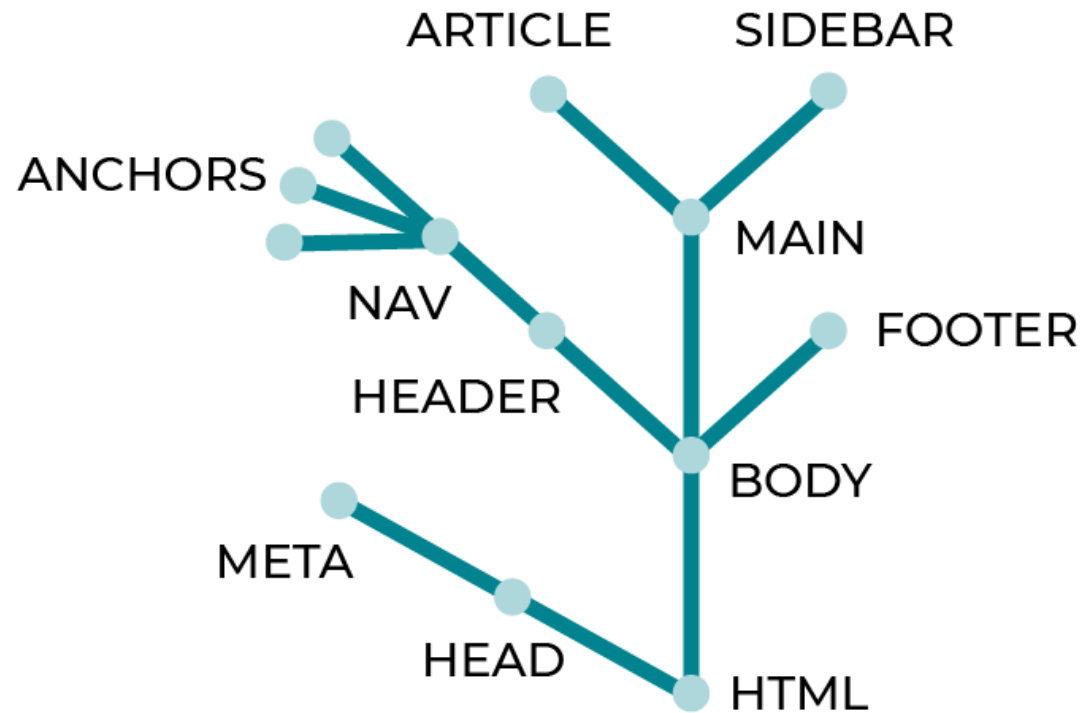
- 1 – Créez un objet Personnage avec les attributs suivant : name, life, atk et def. Pensez bien à les initialiser dans le constructeur
- 2 – Créez une méthode qui affiche le nom du personnage
- 3 – Créez une méthode qui prend en paramètre un objet personnage et qui l'attaque en lui retirant de la vie en fonction de l'attaque du personnage
- 4 – Bonus : Utilisez des fonctions fléchées

Après la théorie, passons à la
pratique !

Le JavaScript et le web

Manipuler le DOM

Qu'est ce que le DOM



Le document

Le point de départ de la page web

document.getElementById()

```
<p id="personnage">Luke Skywalker</p>
```

```
const luke = document.getElementById('personnage');
```

`document.getElementsByClassName()`

```
<div>  
  <div class="personnage">Luke Skywalker</div>  
  <div class="personnage">Ben Kenobi</div>  
  <div class="personnage">Han Solo</div>  
</div>
```

```
const personnages = document.getElementsByClassName('personnage');  
const luke = personnages[0];
```

`document.getElementsByTagName()`

```
<div>  
  <article>Luke Skywalker</article>  
  <article>Ben Kenobi</article>  
  <article>Han Solo</article>  
</div>
```

```
const personnages = document.getElementsByTagName('article');  
const han = personnages[2];
```


document.querySelector()

```
<div id="myId">
  <p>
    <span><a href="#">Lien 1</a></span>
    <a href="#">Lien 2</a>
    <span><a href="#">Lien 3</a></span>
  </p>
  <p class="article">
    <span><a href="#">Lien 4</a></span>
    <span><a href="#">Lien 5</a></span>
    <a href="#">Lien 6</a>
  </p>
  <p>
    <a href="#">Lien 7</a>
    <span><a href="#">Lien 8</a></span>
    <span><a href="#">Lien 9</a></span>
  </p>
</div>
```

```
/** Retourne le lien 6 */
```

```
const elt = document.querySelector("#myId p.article > a");
```

Exercice d'application

- 1 – Accédez à l'éditeur en ligne a cette adresse : <https://codepen.io/louis-zerri/pen/wvdeWWg>
- 2 - Récupérez l'élément ayant pour ID main-content grâce à son ID ;
- 3 - Récupérez les éléments ayant pour classe important ;
- 4 - Récupérez les éléments de type article ;
- 5 - Récupérez les éléments de type li qui sont dans une liste ul ayant la classe important . Cette liste doit elle-même être dans un article (article) ;
- 6 - En reprenant le résultat de la tâche précédente, récupérez l'élément li suivant de la liste ul .

Les recherches depuis un élément

```
<div id="parent">
  <div id="previous">Précédent</div>
  <div id="main">
    <p>Paragraphe 1</p>
    <p>Paragraphe 2</p>
  </div>
  <div id="next">Suivant</div>
</div>
```

```
const elt = document.getElementById('main');
```

Modifier le contenu d'un élément

```
let elt = document.getElementById('main');  
elt.innerHTML = "<ul><li>Elément 1</li><li>Elément 2</li></ul>";
```

```
<div id="main">  
  <ul>  
    <li>Elément 1</li>  
    <li>Elément 2</li>  
  </ul>  
</div>
```

Modifier les classes

```
// Ajoute la classe nouvelleClasse à l'élément  
elt.classList.add("nouvelleClasse");
```

```
// Supprime la classe nouvelleClasse que l'on venait d'ajouter  
elt.classList.remove("nouvelleClasse");
```

```
// Retournera false car on vient de la supprimer  
elt.classList.contains("nouvelleClasse");
```

```
// Remplacera oldClass par newClass si oldClass était présente sur l'élément  
elt.classList.replace("oldClass", "newClass");
```

Changer les styles

```
// Change la couleur du texte de l'élément à blanche  
elt.style.color = "#fff";  
  
// Change la couleur de fond de l'élément en noir  
elt.style.backgroundColor = "#000";  
  
// Met le texte de l'élément en gras  
elt.style.fontWeight = "bold";
```

Modifier des attributs

Référence à un élément de type input :

```
// Change le type de l'input en un type password
elt.setAttribute("type", "password");

// Change le nom de l'input en my-password
elt.setAttribute("name", "my-password");

// Retourne my-password
elt.getAttribute("name");
```

Créer des nouveaux éléments

```
const newElt = document.createElement("div");
```

Ajouter des enfants

```
const newElt = document.createElement("div");  
let elt = document.getElementById("main");  
  
elt.appendChild(newElt);
```


Supprimer et remplacer des éléments

```
const newElt = document.createElement("div");
let elt = document.getElementById("main");
elt.appendChild(newElt);

// Supprime l'élément newElt de l'élément elt
elt.removeChild(newElt);

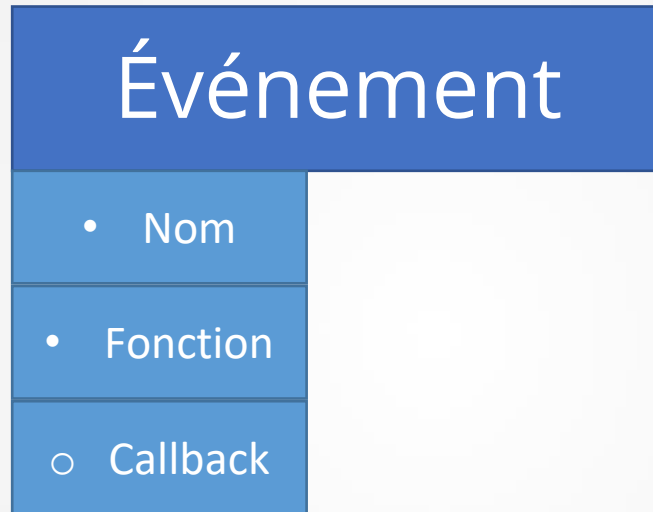
// Remplace l'élément newElt par un nouvel élément de type article
elt.replaceChild(document.createElement("article"), newElt);
```

Exercice d'application

- 1 – Accédez à l'éditeur en ligne a cette adresse : <https://codepen.io/louis-zerri/pen/QWvgEGq>
- 2 - Créez un nouvel élément de type paragraphe p ;
- 3 - Ajoutez votre nouvel élément dans l'élément ayant pour id main ;
- 4 - Ajoutez ce contenu HTML dans l'élément que vous avez créé lors de la première tâche : Mon **grand** contenu ;
- 5 - Ajoutez la classe important à l'élément que vous avez créé lors de la première tâche ;
- 6 - Modifiez les styles de votre élément pour qu'il soit vert.

Les événements

Qu'est ce qu'un événement ?



Un événement est par défaut propagé

Réagir lors d'un clic sur un élément

```
addEventListener(event, callback)
```

Exemple avec l'événement onclick

```
// On récupère l'élément sur lequel on veut détecter le clic
const elt = document.getElementById('mon-lien');

// On écoute l'événement click
elt.addEventListener('click', function() {
    // On change le contenu de notre élément pour afficher "C'est cliqué !"
    elt.innerHTML = "C'est cliqué !";
});
```

La fonction preventDefault()

```
// On récupère l'élément sur lequel on veut détecter le clic
const elt = document.getElementById('mon-lien');

// On écoute l'événement click, notre callback prend un paramètre que nous avons appelé event ici
elt.addEventListener('click', function(event) {

    // On empêche le comportement par défaut de cet élément lors du clic de la souris
    event.preventDefault();

});
```

La fonction stopPropagation()

```
elementInterieur.addEventListener('click', function(event) {
    event.stopPropagation();
    elementAvecMessage.innerHTML = "Message de l'élément intérieur";
});
```

Détecter le mouvement de la souris

```
elt.addEventListener('mousemove', function(event) {  
    const x = event.offsetX; // Coordonnée X de la souris dans l'élément  
    const y = event.offsetY; // Coordonnée Y de la souris dans l'élément  
});
```

Lire le contenu d'un champ texte

```
input.addEventListener('input', function(event) {  
    output.innerHTML = event.target.value;  
});
```

Exercice d'application

- 1 – Accédez à l'éditeur en ligne a cette adresse : <https://codepen.io/louis-zerri/pen/JjNJKEq>
- 2 - Écoutez les événements click depuis l'élément #parent. Puis affichez le nombre de clics dans l'élément #parent-count.
- 3 – Idem mais avec l'élément #child. Il faudra afficher le nombre de clics sur cet élément dans l'élément #child-count.
- 4 - Maintenant, faites de sorte à ce que lorsque vous cliquez sur l'enfant, seul le compteur de l'enfant se mette à jour. N'oubliez pas que l'élément enfant se trouve à l'intérieur de l'élément parent.
- 5 - Evitez qu'un clic sur le lien ne vous fasse changer de page: supprimez ce comportement par défaut.

Exercice d'application

- 1 - Accédez à l'éditeur en ligne a cette adresse : <https://codepen.io/louis-zerri/pen/mdmwEwP>
- 2 - Écoutez les événements input sur l'élément #name afin de savoir quand le contenu du champ texte est changé. Affichez le contenu actuel dans l'élément #res-name
- 3 - Maintenant nous voulons écouter l'événement du changement de choix du genre (#gender), et afficher le résultat dans l'élément #res-gender.
- 4 - Nous souhaitons maintenant afficher les coordonnées de la souris à l'intérieur de l'élément #result dès que celle-ci passe par dessus. Ce que nous voulons, c'est avoir les coordonnées relatives au coin en haut à gauche de l'élément #result.

Les services web

Qu'est ce qu'un service web ?



Un service web est un programme s'exécutant sur un serveur accessible depuis Internet et fournissant un service.

Exemples : Google, une application météo, un réseau social...

Le but d'un service web est donc de fournir un service à celui qui le demande. Et pour ce faire, il met à disposition une **API**.

Qu'est ce qu'une API ?



Une API, ou *Application Programming Interface*, est une interface de communication. Il en existe différents types, mais celle qui nous intéresse est celle qui permet de communiquer avec les services web.

L'API correspond à l'ensemble des demandes que l'on peut faire à un service web. Ces demandes sont appelées des **requêtes**.



Exemples : demander la météo actuelle est une requête, faire une demande d'ami sur un réseau social est une requête ou encore, envoyer un message via une application de messagerie est une requête

Une requête est une donnée qui respecte **le protocole de communication** et qui sont envoyées au serveur. Il existe plusieurs protocoles mais celui qui nous intéresse ici est **le protocole HTTP**

Qu'est ce que le protocole HTTP ?

HTTP signifie *HyperText Transfer Protocol*. C'est un protocole qui permet de **communiquer** avec un site Internet.

Il va permettre de charger des **pages HTML**, des **styles CSS**, des **polices de caractères**, des **images**, etc. Le protocole HTTP nous permet aussi d'envoyer des formulaires et de récupérer et d'envoyer toutes sortes de données depuis ou vers un serveur implémentant ce protocole grâce à son API.

Plusieurs informations se trouvent dans une requête HTTP :

- La méthode : GET, POST, PUT, DELETE
- L'URL
- Les données

Une fois votre requête envoyée et traitée par le service web, celui-ci va vous répondre avec, entre autres, les informations suivantes :

- Les données
- Le code HTTP (200, 400, 401, 403, 404, 500...)

Utilisation de l'API Fetch

Fetch est un ensemble d'objets et de fonctions mis à disposition par le langage JavaScript, afin d'exécuter des requêtes HTTP de manière **asynchrone**.

L'API Fetch va nous permettre d'exécuter des requêtes HTTP sans avoir besoin de recharger la page du navigateur. Cela a plusieurs avantages :

- **Avoir un site plus réactif** car on n'a pas besoin de recharger toute la page dès qu'on a besoin de mettre à jour une partie du contenu
- **Améliorer l'expérience utilisateur** avec du nouveau contenu qui se charge au fur et à mesure qu'on le découvre, par exemple

```
fetch("https://www.une-url.com")  
  .then(response => response.json())  
  .then(response => alert(JSON.stringify(response)))  
  .catch(error => alert("Erreur : " + error));
```

Les verbes HTTP : GET, POST, PUT, DELETE...

GET

```
fetch("https://serviceweb.com/ressource")
  .then(function(res) {
    if (res.ok) {
      return res.json();
    }
  })
  .then(function(value) {
    console.log(value);
  })
  .catch(function(err) {
    // Une erreur est survenue
  });
```

GET avec id

```
fetch("https://serviceweb.com/ressource/" + id)
  .then(function(res) {
    if (res.ok) {
      return res.json();
    }
  })
  .then(function(value) {
    console.log(value);
  })
  .catch(function(err) {
    // Une erreur est survenue
  });
```

POST

```
fetch("http://serviceweb.com/ressource", {  
  method: "POST",  
  headers: {  
    'Accept': 'application/json',  
    'Content-Type': 'application/json'  
  },  
  body: JSON.stringify(jsonBody)  
});
```

PUT

```
fetch("http://serviceweb.com/ressource/" + id, {  
  method: "PUT",  
  headers: {  
    'Accept': 'application/json',  
    'Content-Type': 'application/json'  
  },  
  body: JSON.stringify(jsonBody)  
});
```

DELETE

```
fetch('https://serviceweb.com/ressource/' + id, {  
  method: 'DELETE',  
})  
  .then(res => res.json())  
  .then(res => console.log(res))
```

Utilisation de Postman

<https://www.postman.com/>



Le format JSON

JSON signifie *JavaScript Object Notation*. Il s'agit d'un **format textuel** (contrairement à un format binaire plus léger mais impossible à lire à l'œil humain), se rapprochant en termes de syntaxe de celui des objets dans le langage JavaScript.

Objet JavaScript

```
const obj = {  
  name: "Mon contenu",  
  id: 1234,  
  message: "Voici mon contenu",  
  author: {  
    name: "John"  
  },  
  comments: [  
    {  
      id: 45,  
      message: "Commentaire 1"  
    },  
    {  
      id: 46,  
      message: "Commentaire 2"  
    }  
  ]  
};
```

JSON

```
{  
  "name": "Mon contenu",  
  "id": 1234,  
  "message": "Voici mon contenu",  
  "author": {  
    "name": "John"  
  },  
  "comments": [  
    {  
      "id": 45,  
      "message": "Commentaire 1"  
    },  
    {  
      "id": 46,  
      "message": "Commentaire 2"  
    }  
  ]  
}
```

Valider les données saisie par vos utilisateurs

Ne faite jamais confiance aux données saisies par vos utilisateurs

Valider les données suite à des évènements

```
myInput.addEventListener('input', function(e) {  
    var value = e.target.value;  
    if (value.startsWith('Hello ')) {  
        isValid = true;  
    } else {  
        isValid = false;  
    }  
});
```

Utilisations des regex

```
function isValid(value) {  
    return /^e[0-9]{3,}$/.test(value);  
}
```

Les contraintes HTML5

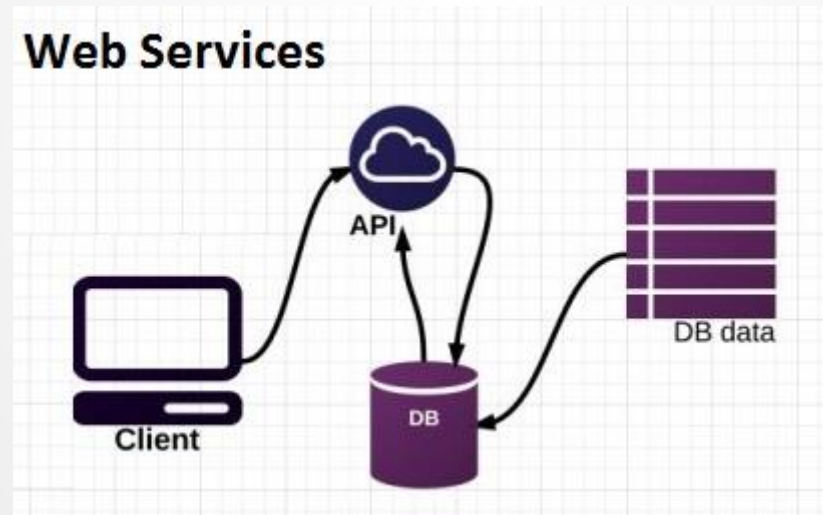
- L'attribut type pour les inputs
- Les attributs de validation simples
- Les patterns

Voyons un exemple pour sécuriser nos
formulaire

Consommer une API RESTful

RESTful ?

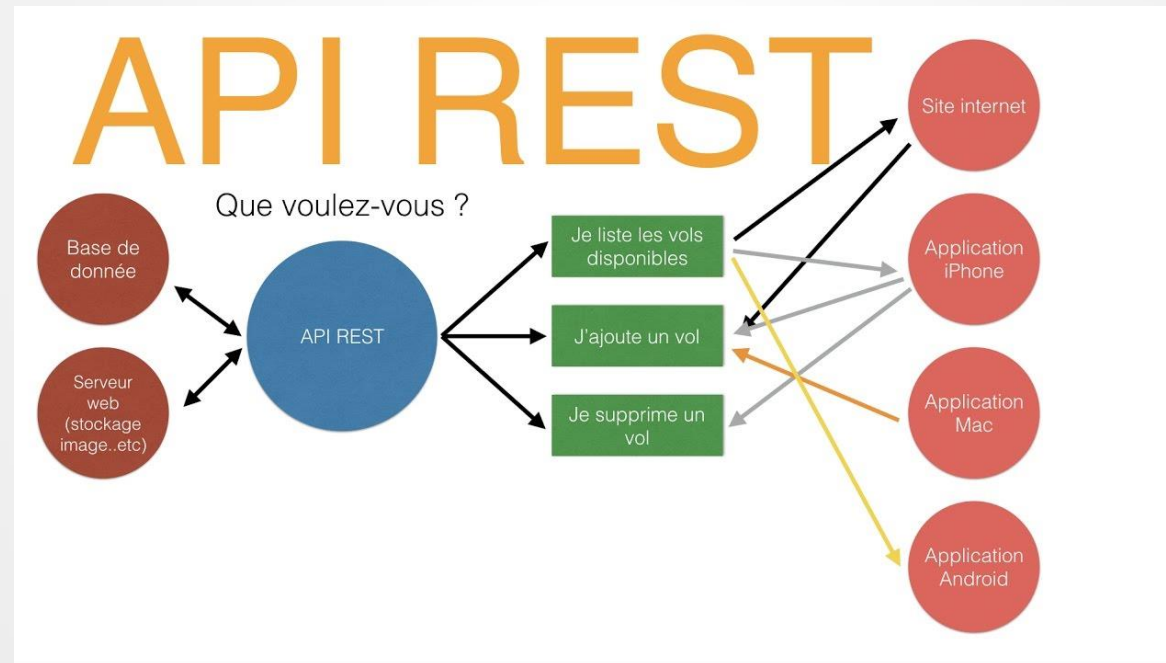
Une API REST (également appelée API RESTful) est une interface de programmation d'application (API ou API web) qui respecte les contraintes du style d'architecture REST et permet d'interagir avec les services web RESTful. L'architecture REST (Representational State Transfer) a été créée par l'informaticien Roy Fielding



Voyons un exemple avec une API qui
contient des films

Travaux pratique 1

Problème : Le client pour lequel vous travaillez possède une base de données et il ne veut pas que vous y accédiez. Par contre il vous met à disposition un service web pour permettre de travailler dessus



Travail à faire :

- 1 – Récupérez l'adresse de l'api d'articles sur :
<http://devserverlouis.ovh>
- 2 – Affichez sur une page web la liste des articles avec toutes les informations associées
- 3 – Avoir la possibilité d'ajouter un article
- 4 – Testez le PUT et le DELETE sur Postman sinon implémentés les sur votre page web

L'asynchrone en JavaScript

Fonctionnement et callback

JavaScript est synchrone et mono-thread

Utiliser l'évent loop

Les flux d'entrée et de sortie

Exécuter du code de manière asynchrone grâce à des fonctions

```
setTimeout(function() {  
  console.log("Je suis ici !")  
}, 5000);  
  
console.log("Où es-tu ?");
```

```
setInterval(() => {  
  console.log("Bonjour !")  
}, 3000)
```

Les callbacks dans les événements par exemple

Attention au *callback hell* et au potentielles erreurs !

```
fs.readFile(filePath, function(err, data) {  
  if (err) {  
    throw err;  
  }  
  // Do something with data  
});
```

Les promises

Lorsque l'on exécute du code asynchrone, celui-ci va immédiatement nous retourner une "promesse" qu'un résultat nous sera envoyé prochainement.

Cette promesse est en fait un objet Promise qui peut être *resolve* avec un résultat, ou *reject* avec une erreur.

```
functionThatReturnsAPromise()  
  .then(function(data) {  
    // Do something with data  
  })  
  .catch(function(err) {  
    // Do something with error  
  });
```

```
returnAPromiseWithNumber2()  
  .then(function(data) { // Data is 2  
    return data + 1;  
  })  
  .then(function(data) { // Data is 3  
    throw new Error('error');  
  })  
  .then(function(data) {  
    // Not executed  
  })  
  .catch(function(err) {  
    return 5;  
  })  
  .then(function(data) { // Data is 5  
    // Do something  
  });
```

Async / await

```
async function fonctionAsynchrone1() { /* code asynchrone */ }
async function fonctionAsynchrone2() { /* code asynchrone */ }

async function fonctionAsynchrone3() {
  const value1 = await fonctionAsynchrone1();
  const value2 = await fonctionAsynchrone2();
  return value1 + value2;
}
```

Pour gérer les erreurs, on utilise un bloc try / catch

Aller plus loin

Conclusion