



GEO 7630

Intégration et visualisation de données géographiques

Semaine 9 -

- **Bases HTML, CSS et Javascript**
- **Introduction au WebMapping FrontEnd**



Objectifs du cours

Introduction au webmapping (Front-end)

- Bases HTML, CSS et Javascript
- Introduction au format JSON/GeoJSON
- Introduction aux logiciels de gestion de versions décentralisés (GIT)
- Introduction aux Interfaces de programmation applicatives (APIs)
- Revue des APIs cartographiques

Laboratoire (Javascript + MapLibreGL):

- Création d'une application cartographique web de base (GeoJSON)



Bases HTML, CSS et Javascript

- HTML
- Pratique HTML
- CSS
- Pratique CSS
- JS
- Pratique JS

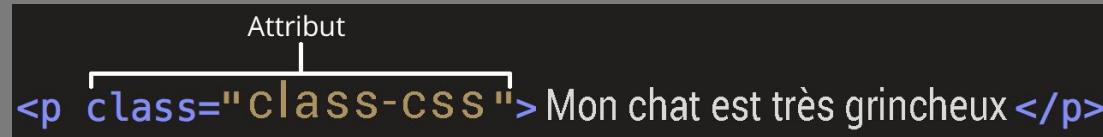


HTML

- Anatomie d'un élément HTML
- Éléments imbriqués
- Attributs
- Anatomie d'un document HTML

[Commencer avec le HTML - Apprendre le développement web | MDN](#)

HTML



Anatomie d'un élément HTML

HyperText Markup Language

- Balise d'ouverture
- Attributs
- Contenu
- Balise de fermeture

HTML

Éléments imbriqués

- Fermé dans l'ordre inverse
- Organiser le contenu hiérarchiquement
- Créer une structure complexe
- Structure plus complexe et hiérarchique pour votre contenu HTML


css

Copy code

```
<section>
  <h1>Titre de la section</h1>
  <p>Ceci est un paragraphe dans la section.</p>
  <ul>
    <li>Élément de liste 1</li>
    <li>Élément de liste 2</li>
  </ul>
</section>
```

HTML

html

 Copy code

```
<input
  id="nom"
  class="champ-texte"
  type="text"
  value="Entrez votre nom"
  required
  readonly
  disabled
>
```

Attributes

- id
- class
- style
- src
- href
- alt
- title
- target
- value
- checked
- disabled
- readonly
- required
- ...

HTML

Balises

html

Copy code

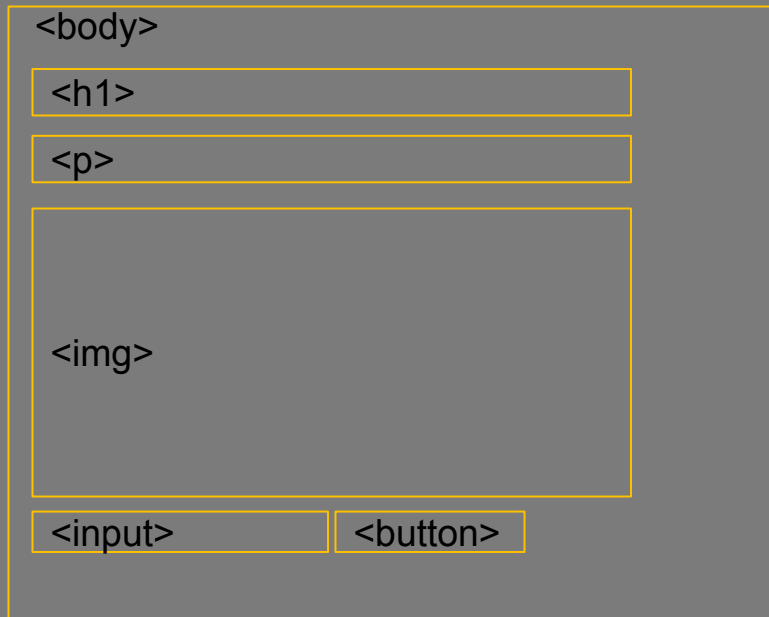
```
<!DOCTYPE html>
<html>
  <head>
    <title>Ma page web</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <h1>Bienvenue sur ma page web</h1>
    <p>Cette page est en construction, revenez bientôt pour découvrir son contenu p
    
      <button>Envoyer</button>
    </div>
  </body>
</html>
```

- `<html>` : balise racine d'un document HTML.
- `<head>` : balise contenant des informations sur le document HTML, telles que les méta-données, les liens vers des fichiers CSS et des scripts JavaScript.
- `<body>` : balise contenant le contenu principal d'un document HTML.
- `<h1>` à `<h6>` : balises de titre pour les différentes sections d'un document.
- `<p>` : balise pour les paragraphes de texte.
- `<a>` : balise pour les liens hypertexte.
- `` : balise pour les images.
- `<div>` : balise générique pour diviser une page en sections.
- `<input>` : balise pour les champs de saisie de formulaire.
- `<button>` : balise pour les boutons de formulaire.

HTML

Balises

- `<html>` : balise racine d'un document HTML.
- `<head>` : balise contenant des informations sur le document HTML, telles que les méta-données, les liens vers des fichiers CSS et des scripts JavaScript.
- `<body>` : balise contenant le contenu principal d'un document HTML.
- `<h1>` à `<h6>` : balises de titre pour les différentes sections d'un document.
- `<p>` : balise pour les paragraphes de texte.
- `<a>` : balise pour les liens hypertexte.
- `` : balise pour les images.
- `<div>` : balise générique pour diviser une page en sections.
- `<input>` : balise pour les champs de saisie de formulaire.
- `<button>` : balise pour les boutons de formulaire.



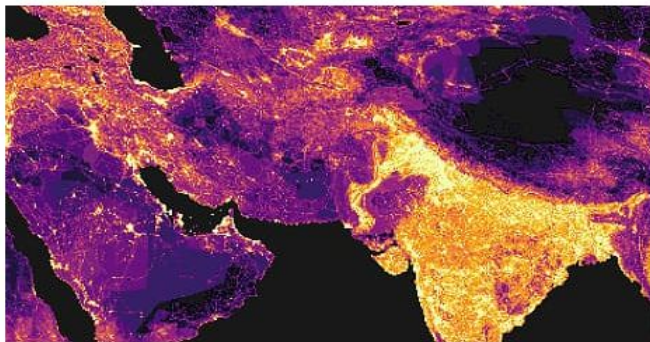
HTML

Balises

- `<html>` : balise racine d'un document HTML.
- `<head>` : balise contenant des informations sur le document HTML, telles que les méta-données, les liens vers des fichiers CSS et des scripts JavaScript.
- `<body>` : balise contenant le contenu principal d'un document HTML.
- `<h1>` à `<h6>` : balises de titre pour les différentes sections d'un document.
- `<p>` : balise pour les paragraphes de texte.
- `<a>` : balise pour les liens hypertexte.
- `` : balise pour les images.
- `<div>` : balise générique pour diviser une page en sections.
- `<input>` : balise pour les champs de saisie de formulaire.
- `<button>` : balise pour les boutons de formulaire.

Bienvenue sur ma page web

Cette page est en construction, revenez bientôt pour découvrir son contenu passionnant !



HTML

Balises

<link> : utilisée pour lier un fichier externe (CSS/icône ...)

```
html Copy code
```

```
<link rel="stylesheet" href="chemin/vers/fichier.css">
```

```
html Copy code
```

```
<link rel="icon" href="chemin/vers/fichier.ico">
```

<script> : utilisée pour incorporer ou lier un script JavaScript

```
html Copy code
```

```
<script src="chemin/vers/fichier.js"></script>
```

```
html Copy code
```

```
<script>
  // Code JavaScript ici
</script>
```

```
html
```

```
Copy code
```

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Titre de la page</title>
```

```
    <meta charset="UTF-8">
```

```
    <meta name="description" content="Description de la page">
```

```
    <meta name="keywords" content="Mots-clés de la page">
```

```
    <link rel="stylesheet" href="style.css">
```

```
    <script src="script.js"></script>
```

```
  </head>
```

```
  <body>
```

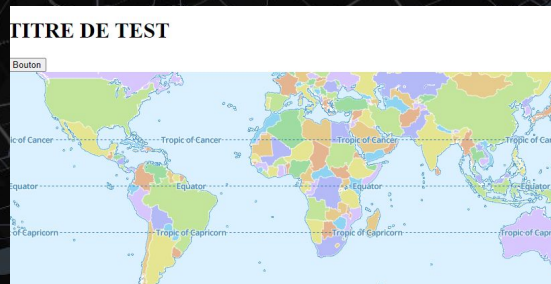
```
    <h1>Titre principal</h1>
```

```
    <p>Contenu de la page</p>
```

```
  </body>
```

```
</html>
```

Mise en pratique



<https://jsfiddle.net>

Copiez le code HTML

<https://maplibre.org/maplibre-gl-js/docs/examples/3d-terrain/>

Ajouter un div avec un titre :

```
<div id="titre">  
<h1>TITRE DE TEST</h1>  
</div>
```

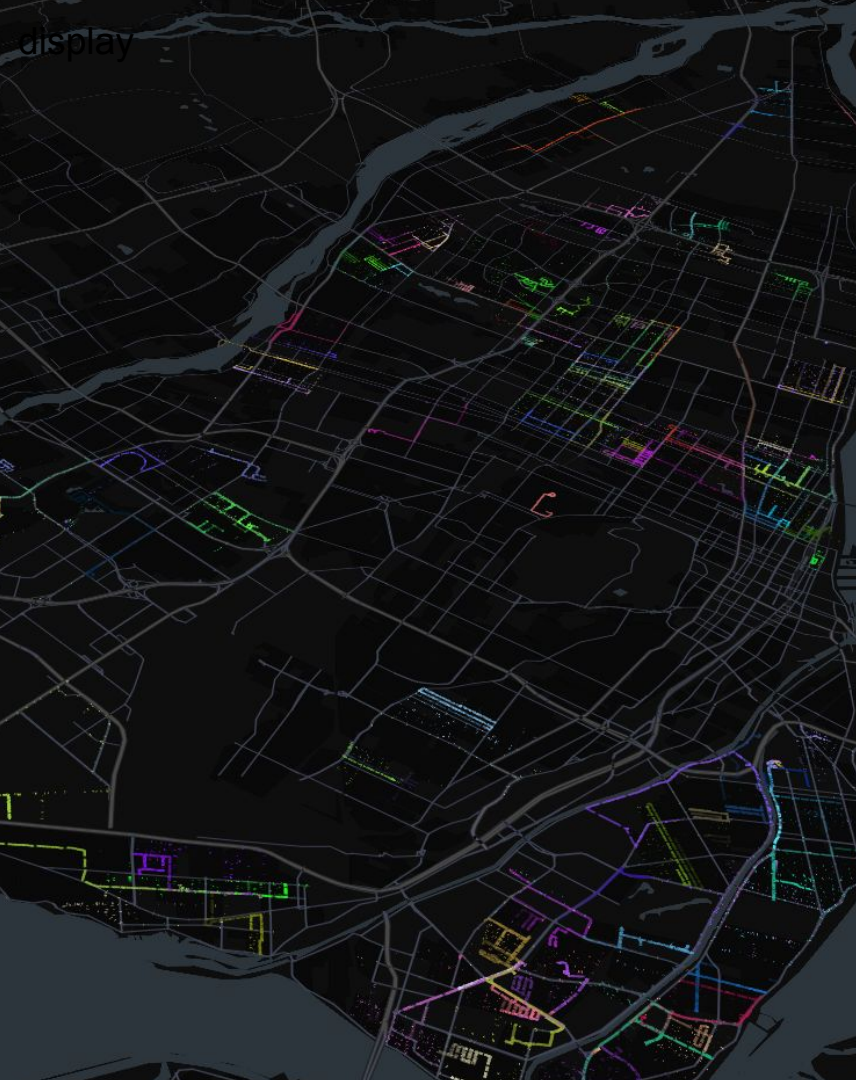
Changez la hauteur de la carte pour laisser apparaître votre titre

```
height:80%;
```

Ajoutez un bouton

```
<button> Bouton </button>
```


Rétro / Pause



CSS

- **C'est quoi (Cascading Style Sheets)**
- **Bases**
 - width, height
 - padding
 - margin
 - color
- **Display**
- **Position**

CSS

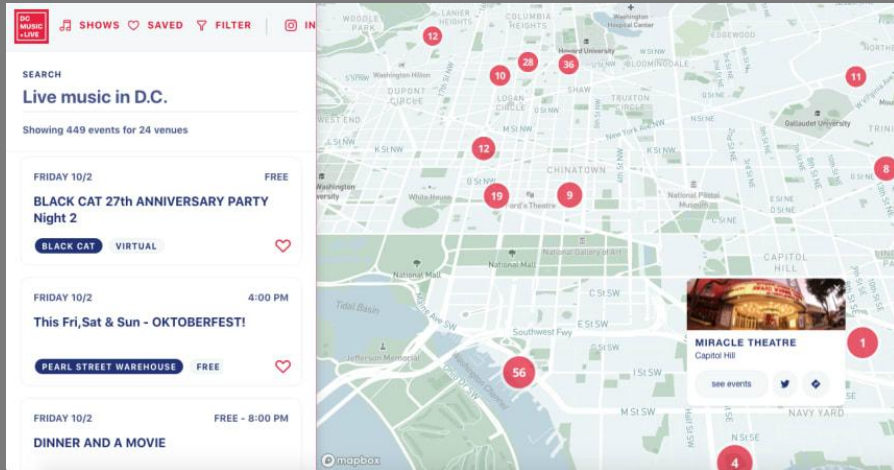
C'est quoi

Le CSS (Cascading Style Sheets) est un langage de feuilles de style utilisé pour décrire la présentation visuelle d'un document HTML ou XML.

- Couleur du texte
- Taille d'un élément
- Type de police
- Effets visuels
- Animation

[CSS : Feuilles de style en cascade](#)

[Awwwards Conference](#)

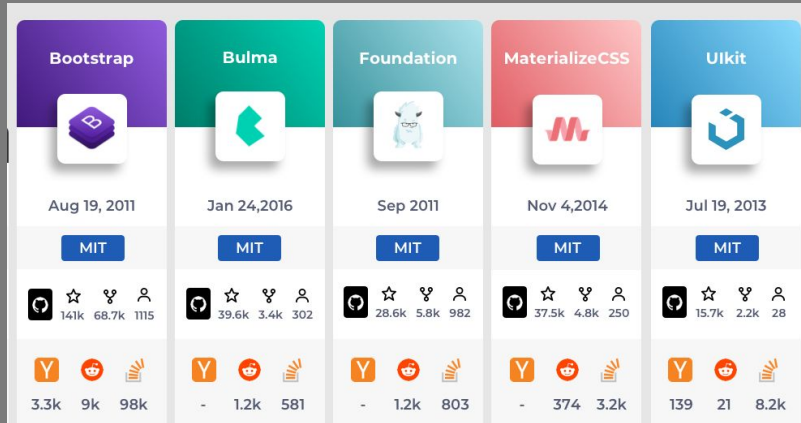


CSS

C'est quoi


Le CSS permet de séparer la présentation du contenu d'une page web

- Facilite la maintenance du site
- Permet de changer l'apparence de la page sans modifier le contenu HTML
- Appliquer des styles différents à différentes parties de la page
- Créer des designs complexes et personnalisés.



[The Most Trending ⚡ CSS Frameworks ⚡ Analogy 2020 - DEV Community](#)

css

 Copy code

```
/* Commentaire en CSS */

/* Déclaration de style pour les éléments de type h1 */
h1 {
  color: red; /* Déclaration de couleur */
  font-size: 2em; /* Déclaration de taille de police */
  text-align: center; /* Déclaration de l'alignement du texte */
}

/* Déclaration de style pour les éléments de type p */
p {
  color: blue;
  font-size: 1.2em;
  line-height: 1.5;
}

/* Déclaration de style pour une classe nommée "ma-classe" */
.ma-classe {
  font-weight: bold;
  text-decoration: underline;
}
```

CSS

Bases

Structure d'un fichier

- Commentaires
- Déclaration de style pour une balise
- Déclaration de style pour une classe
- Déclaration de style pour un id

```
css Copy code

/* Style pour les éléments avec la classe "ma-classe" */
.ma-classe {
  color: blue;
  font-weight: bold;
}

/* Style pour l'élément avec l'ID "mon-id" */
#mon-id {
  color: red;
  font-style: italic;
}

/* Style pour les éléments avec la classe "autre-classe" à l'intérieur d'un élément
#autre-id .autre-classe {
  color: green;
  font-size: 1.2em;
}
```

CSS

Bases

Structure d'un fichier

- Commentaires
- Déclaration de style pour une balise
- Déclaration de style pour une classe
- Déclaration de style pour un id

css

Copy code

```
/* Définir la taille d'une image en pixels */
img {
  width: 300px;
  height: 200px;
}

/* Définir la taille d'une section en pourcentage */
section {
  width: 80%;
  height: 50%;
}

/* Définir la taille d'une div en pourcentage de la fenêtre du navigateur */
div {
  width: 50vw;
  height: 50vh;
}
```

CSS

Width / Height

- px
- %
- em/rem
- vh/vw

[Valeurs et unités CSS - Apprendre le développement web | MDN](#)

```
css Copy code

/* Définir la taille d'une section en pourcentage de la hauteur de la fenêtre du n
section {
  width: 80%;
  height: 60vh;
}

/* Définir la taille d'une image en pourcentage de la hauteur de son conteneur par
img {
  width: 300px;
  height: 50%;
}
```

CSS

Width / Height

- px
- %
- em/rem
- vh/vw

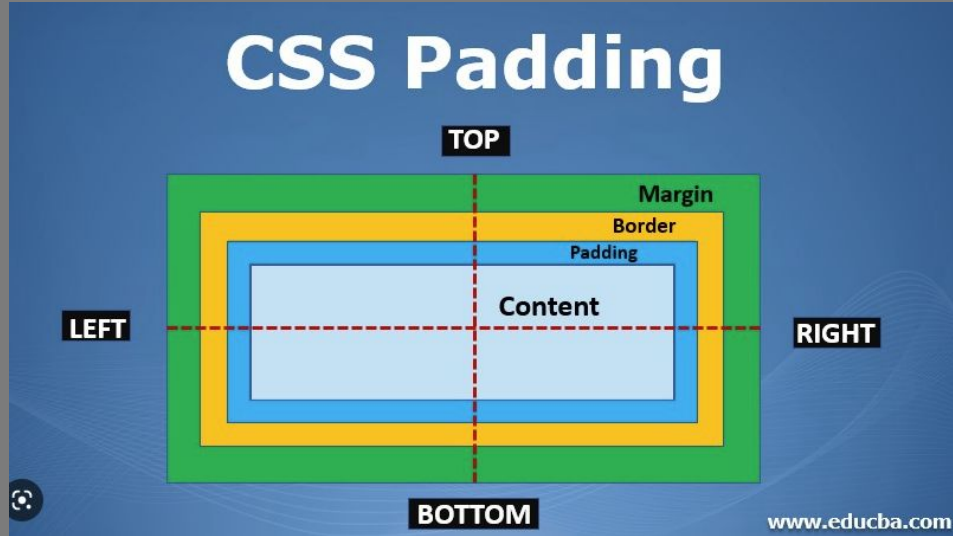
[Valeurs et unités CSS - Apprendre le développement web | MDN](#)

CSS

Padding / Margin

Propriétés CSS utilisées pour ajouter de l'espace autour d'un élément

- *padding* est l'espace situé à l'intérieur de la bordure d'un élément
- *margin* est l'espace situé à l'extérieur de la bordure d'un élément
- Peuvent être définies pour chaque côté de l'élément individuellement en utilisant les propriétés *padding-top*
-




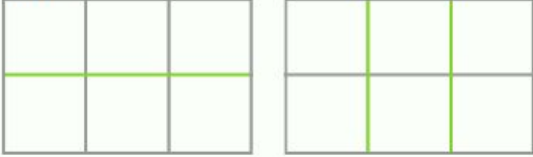
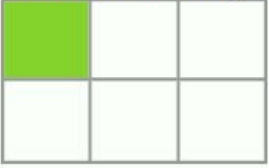
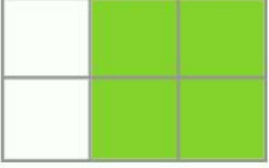
CSS

Display

Propriété CSS utilisée pour définir comment un élément HTML doit être affiché.

- *float*
- *block*
- *inline*
- *inline-block*
- *none*
- *flex*
- *grid*

[Découverte du CSS : Float, Flex ou grid ?](#)

Grid Track	<p>Can be either the column or row of the grid.</p> 
Grid Line	<p>Lines that define the structure of the grid. Think of them as the lines between the grid tracks.</p> 
Grid Cell	<p>An individual grid unit, the space enclosed by adjacent horizontal and vertical grid lines.</p> 
Grid Area	<p>Now this is the cool part. Grid allows you to define an area made up of multiple grid cells.</p> 





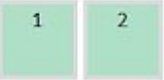
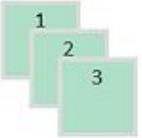
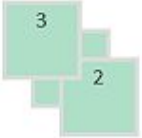
CSS

Display

Propriété CSS utilisée pour définir comment un élément HTML doit être affiché.

- *float*
- *block*
- *inline*
- *inline-block*
- *none*
- *flex*
- ***grid***

[Découverte du CSS : Float, Flex ou grid ?](#)

 Default position	 1 Moved across  2 Moved down	 Centered horizontally automatically
	position:relative; 1. left:100px; 2. top:50px;	margin:0 auto;
 1 2 Side-by-side	 1 2 3 Overlapping	 3 2 Out of order
1: position:absolute; top:40px;left:40px; 2: position:absolute; top:40px;left:100px;	position:absolute; 1: top:10px;left:30px; 2: top:30px;left:50px; 3: top:50px;left:70px;	position:absolute; 1: top:30px;left:50px; 2: top:50px;left:70px; 3: top:10px;left:30px;

CSS

Position

Propriétés CSS définir le type de positionnement d'un élément HTML par rapport à son conteneur

- *Static*
- *Relative*
- *Absolute*
- *Fixed*
- *Sticky*

[La position en CSS](#)

```
/* Utilisation des mots clés */
h1 {
  color: red;
}

/* Utilisation des codes hexadécimaux */
p {
  color: #00FF00;
}

/* Utilisation des codes RGB */
span {
  color: rgb(255, 0, 0);
}

/* Utilisation des codes RGBA */
a {
  color: rgba(0, 0, 255, 0.5);
}

/* Utilisation des noms de couleurs */
div {
  color: blueviolet;
}

/* Utilisation des couleurs HSL */
section {
  color: hsl(120, 100%, 50%);
}

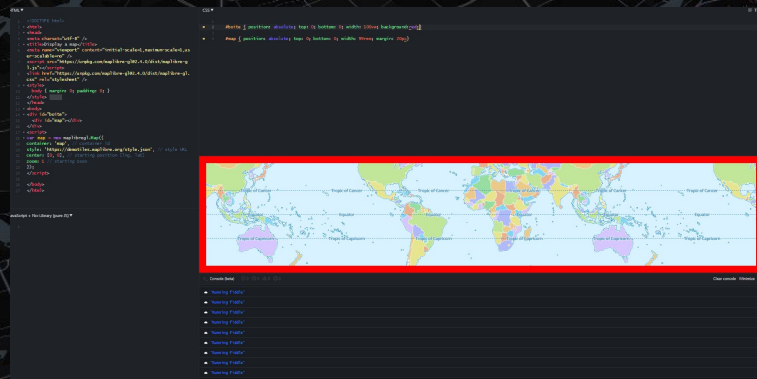
/* Utilisation des couleurs HSLA */
article {
  color: hsla(240, 100%, 50%, 0.7);
}
```

CSS

Couleurs

- *simple keyword*
- *complex keyword*
- *HEX*
- *rgb*
- *rgba*
- *hsl (Teinte saturation lumière)*
- *hsla*

Mise en pratique



<https://jsfiddle.net>

Ajoutez du CSS dans le jsfiddle précédent

```
#boite { position: absolute; top: 0; bottom: 0; width: 100vw; background:red;}
```

```
#map { position: absolute; top: 0; bottom: 0; width: 99rem; margin: 20px}
```

Rétro / Pause





Javascript

- C'est quoi
- Variable
- Types
- Opérateurs
- Structure de contrôle
- `console.log()`



Javascript

C'est quoi ?

JavaScript est un langage de programmation utilisé pour créer des pages Web interactives et dynamiques, avec des effets visuels et des interactions.

Javascript

Variables

keyword	const	let	var
global scope	NO	NO	YES
function scope	YES	YES	YES
block scope	YES	YES	NO
can be reassigned	NO	YES	YES

VAR : à portée de fonction et peut être mise à jour et redéclarée.

LET : à portée de bloc, peut être mise à jour mais ne peut pas être redéclarée.

CONST : à portée de bloc, ne peut pas être mise à jour ni redéclarée.



Javascript

Types

- **Number** : 1 / 1.1
- **String** : 'text' / "text" / `text`
- **Boolean** : true / false ; 1 / 0 ...
- **Array** : [1,2,3] / ['text','text','text']
- **Object** : { key:value }
- **Function** : myFunction (payload) {code}

javascript

Copy code

```
let isTrue = true;

if (isTrue) {
  console.log("This will be executed");
} else {
  console.log("This will not be executed");
}
```

javascript

Copy code

```
let bool1 = true;
let bool2 = false;
let bool3 = Boolean(1);
let bool4 = Boolean(0);
let bool5 = Boolean("");
let bool6 = Boolean("Hello");

console.log(bool1); // true
console.log(bool2); // false
console.log(bool3); // true
console.log(bool4); // false
console.log(bool5); // false
console.log(bool6); // true
```

Javascript

Types

Boolean :

- true ou false
- 1 ou 0
- vide ou null

javascript

Copy code

```
let numbers = [1, 2, 3, 4, 5];
```

scss

Copy code

```
numbers[2] = 10; // remplace le 3ème élément par 10  
console.log(numbers); // output: [1, 2, 10, 4, 5]
```

scss

Copy code

```
numbers.push(6, 7); // ajoute les éléments 6 et 7 à la fin de l'array  
console.log(numbers); // output: [1, 2, 10, 4, 5, 6, 7]
```

scss

Copy code

```
numbers.pop(); // supprime le dernier élément de l'array  
console.log(numbers); // output: [1, 2, 10, 4, 5, 6]
```

yaml

Copy code

```
const line = [  
  { lat: 48.8566, lon: 2.3522 }, // Paris  
  { lat: 45.7640, lon: 4.8357 }, // Lyon  
  { lat: 43.6045, lon: 1.4442 }, // Toulouse  
  { lat: 51.5098, lon: -0.1180 } // Londres  
];
```

Javascript

Types

Array :

- [1, 2, 3]
- ['text', 'text', 'text']
- [[], [], ...]
- [{}, {}, ...]
- [[], {}, ...]

Javascript

Types

Array :

- [1, 2, 3]
- ['text', 'text', 'text']
- [[], [], ...]
- [{}, {}, ...]
- [[], {}, true ,...]

javascript

```
let myArray = [  
  [],  
  {},  
  "Hello",  
  [1, 2, 3],  
  { name: "John", age: 30 },  
  true,  
  null  
];
```


Javascript

Types

Object : { key:value }

javascript [Copy code](#)

```
let person = {
  firstName: "John",
  lastName: "Doe",
  age: 30,
  address: {
    street: "123 Main St",
    city: "Anytown",
    state: "CA",
    zip: "12345"
  },
  hobbies: ["reading", "swimming", "traveling"],
  getFullName: function() {
    return this.firstName + " " + this.lastName;
  }
};
```

lua [Copy code](#)

```
console.log(person.firstName); // output: "John"
console.log(person.address.city); // output: "Anytown"
console.log(person.getFullName()); // output: "John Doe"
```

javascript [Copy code](#)

```
user.age = 31; // met à jour la propriété age de l'objet user
console.log(user.age); // output: 31
```

json [Copy code](#)

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [2.3522, 48.8566]
  },
  "properties": {
    "name": "Paris",
    "country": "France"
  }
}
```

Javascript

Types

Object : { key:value }

json

```
{  
  "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [2.3522, 48.8566]  
  },  
  "properties": {  
    "name": "Paris",  
    "country": "France"  
  }  
}
```

json

Copy code

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [2.3522, 48.8566]
      },
      "properties": {
        "name": "Paris",
        "country": "France"
      }
    },
    {
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [2.3522, 48.8566],
          [4.8357, 45.7640],
          [1.4442, 43.6045]
        ]
      },
      "properties": {
        "name": "Route des vacances",
        "distance": 1023.6
      }
    }
  ]
}
```

Javascript

Types

Object : { key:value }

Javascript

Types

Functions

javascript

```
function addNumbers(num1, num2) {  
  return num1 + num2;  
}
```

scss

Copy code

```
let sum = addNumbers(5, 10); // call the function with arguments 5 and 10  
console.log(sum); // output: 15
```

javascript

Copy code

```
function calculateDistance(lat1, lon1, lat2, lon2) {  
  const R = 6371; // rayon de la Terre en kilomètres  
  const dLat = (lat2 - lat1) * Math.PI / 180; // différence de latitude en radians  
  const dLon = (lon2 - lon1) * Math.PI / 180; // différence de longitude en radians  
  const a =  
    Math.sin(dLat / 2) * Math.sin(dLat / 2) +  
    Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *  
    Math.sin(dLon / 2) * Math.sin(dLon / 2);  
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));  
  const d = R * c; // distance en kilomètres  
  return d;  
}  
  
const distance = calculateDistance(48.8566, 2.3522, 51.5072, -0.1276);  
console.log(`La distance entre Paris et Londres est de ${distance.toFixed(2)} km`);
```

Javascript

Structures de contrôle

Conditionnelle :

- if - else
- switch - case

javascript

Copy

```
let lat = 48.8566;
let lng = 2.3522;

if (lat >= 48.853 && lat <= 48.857 && lng >= 2.349 && lng <= 2.353) {
  console.log("La coordonnée se trouve dans la zone géographique.");
} else {
  console.log("La coordonnée ne se trouve pas dans la zone géographique.");
}
```


javascript

Copy code

```
let lat = 48.8566;
let lng = 2.3522;
let locationType;

switch (true) {
  case lat >= 48.853 && lat <= 48.857 && lng >= 2.349 && lng <= 2.353:
    locationType = "Quartier général";
    break;
  case lat >= 48.860 && lat <= 48.864 && lng >= 2.305 && lng <= 2.309:
    locationType = "Bureau";
    break;
  default:
    locationType = "Emplacement inconnu";
    break;
}

console.log(`L'emplacement est de type "${locationType}".`);
```

Javascript

Structures de contrôle

Conditionnelle :

- if - else
- switch - case

```
javascript Copy code

// Supposons que nous avons un objet avec des noms de villes et leurs coordonnées
const cities = {
  Paris: [2.3522, 48.8566],
  Lyon: [4.8357, 45.7640],
  Toulouse: [1.4442, 43.6045]
};

// Boucle for...in pour parcourir chaque ville et afficher son nom et ses coordonnées
for (const city in cities) {
  console.log(`Ville: ${city}, Coordonnées: (${cities[city][0]}, ${cities[city][1]})`);
}
```

Javascript

Structures de contrôle

Boucles :

- for
- while

javascript

Copy code

```
// Supposons que nous avons un objet avec des noms de villes et leurs coordonnées
const cities = {
  Paris: [2.3522, 48.8566],
  Lyon: [4.8357, 45.7640],
  Toulouse: [1.4442, 43.6045]
};

// Boucle for...in pour parcourir chaque ville et afficher son nom et ses coordonnées
for (const city in cities) {
  // Si le nom de la ville contient la lettre "u", on ignore cette ville et on passe à la suivante
  if (city.includes('u')) {
    continue;
  }

  // Si le nom de la ville est "Lyon", on arrête la boucle ici
  if (city === 'Lyon') {
    break;
  }

  console.log(`Ville: ${city}, Coordonnées: (${cities[city][0]}, ${cities[city][1]})`);
}
```

Javascript

Structures de contrôle

Saut:

- break
- continue

Javascript

Opérateurs

Opérateurs d'affectation :

- = (affectation simple)
- += (ajout et affectation)
- -= (soustraction et affectation)
- *= (multiplication et affectation)
- /= (division et affectation)
- %= (modulo et affectation)

javascript  Copy code

```
let lat = 48.8566;  
let lng = 2.3522;
```

javascript  Copy code


```
let x = 10;  
let y = 20;  
x += y; // x est maintenant égal à 30
```

javascript  Copy code

```
let lat1 = 48.8566;  
let lat2 = 45.7640;  
let distance = 0;  
distance = lat1 - lat2; // distance est maintenant égale à 3.0926
```

javascript  Copy code

```
let radius = 6371;  
let distance = 10;  
let circumference = 0;  
circumference = 2 * Math.PI * radius * distance; // circumference est maintenant égale à 400000
```

javascript  Copy code

```
let distance = 100;  
let time = 2;  
let speed = 0;  
speed = distance / time; // speed est maintenant égal à 50
```

javascript  Copy code

```
let latitude = 48.8566;  
let longitude = 2.3522;  
let remainder = 0;  
remainder = latitude % longitude; // remainder est maintenant égal à 0.1522
```

Javascript

Opérateurs

Opérateurs d'affectation :

- = (affectation simple)
- += (ajout et affectation)
- -= (soustraction et affectation)
- *= (multiplication et affectation)
- /= (division et affectation)
- %= (modulo et affectation)

javascript  Copy code

```
let lat = 48.8566;  
let lng = 2.3522;
```

javascript  Copy code


```
let x = 10;  
let y = 20;  
x += y; // x est maintenant égal à 30
```

javascript  Copy code

```
let lat1 = 48.8566;  
let lat2 = 45.7640;  
let distance = 0;  
distance = lat1 - lat2; // distance est maintenant égale à 3.0926
```

javascript  Copy code

```
let radius = 6371;  
let distance = 10;  
let circumference = 0;  
circumference = 2 * Math.PI * radius * distance; // circumference est maintenant égale à 4000
```

javascript  Copy code

```
let distance = 100;  
let time = 2;  
let speed = 0;  
speed = distance / time; // speed est maintenant égal à 50
```

javascript  Copy code

```
let latitude = 48.8566;  
let longitude = 2.3522;  
let remainder = 0;  
remainder = latitude % longitude; // remainder est maintenant égal à 0.1522
```

Javascript

Opérateurs

Opérateurs de comparaison :

- == (égalité)
- === (égalité stricte)
- != (différence)
- !== (différence stricte)
- < (inférieur à)
- > (supérieur à)
- <= (inférieur ou égal à)
- >= (supérieur ou égal à)

javascript

Copy code

```
let latitude1 = 48.8566;  
let latitude2 = 48.8566;  
console.log(latitude1 == latitude2); // Affiche true
```

javascript

Copy code

```
let longitude1 = 2.3522;  
let longitude2 = 3.7038;  
console.log(longitude1 != longitude2); // Affiche true
```

javascript

Copy code

```
let city1 = { name: "Paris", latitude: 48.8566, longitude: 2.3522 };  
let city2 = { name: "Paris", latitude: 48.8566, longitude: 2.3522 };  
console.log(city1 === city2); // Affiche false car city1 et city2 sont deux objets
```

yaml

Copy code

```
let city3 = { name: "Paris", latitude: 48.8566, longitude: 2.3522 };  
let city4 = { name: "Lyon", latitude: 45.7640, longitude: 4.8357 };  
console.log(city3 !== city4); // Affiche true
```

javascript

Copy code

```
let distance1 = 100;  
let distance2 = 50;  
console.log(distance1 > distance2); // Affiche true
```


javascript

Copy code

```
let city = {  
  name: "Paris",  
  lat: 48.8566,  
  lng: 2.3522  
};  
  
let locationStatus = city.lat > 0 ? "Northern Hemisphere" : "Southern Hemisphere";  
  
console.log(`The city of ${city.name} is located in the ${locationStatus}.`);
```

Javascript

Opérateurs

Opérateurs ternaires:

- condition ? exprSiVrai : exprSiFaux

javascript

Copy code

```
let nbPoints = 10;
console.log("Nombre de points avant l'incrémentation : " + nbPoints); // affiche 10
nbPoints++;
console.log("Nombre de points après l'incrémentation : " + nbPoints); // affiche 11
```

javascript

Copy code

```
let nbPoints = 10;
console.log("Nombre de points avant la décrémentation : " + nbPoints); // affiche 10
nbPoints--;
console.log("Nombre de points après la décrémentation : " + nbPoints); // affiche 9
```

javascript

Copy code

```
// Tableau de coordonnées géographiques (latitude, longitude)
const coords = [
  [48.8566, 2.3522],
  [51.5072, -0.1276],
  [41.9028, 12.4964],
  [40.4168, -3.7038]
];
```

```
function decrementCoords() {
  for (let i = coords.length - 1; i >= 0; i--) {
    console.log(`Coordonnées avant : ${coords[i]}`);
    coords[i][0]--; // Décrémente la latitude de 1
    coords[i][1]--; // Décrémente la longitude de 1
    console.log(`Coordonnées après : ${coords[i]}`);
  }
}
```

```
decrementCoords();
```

Javascript

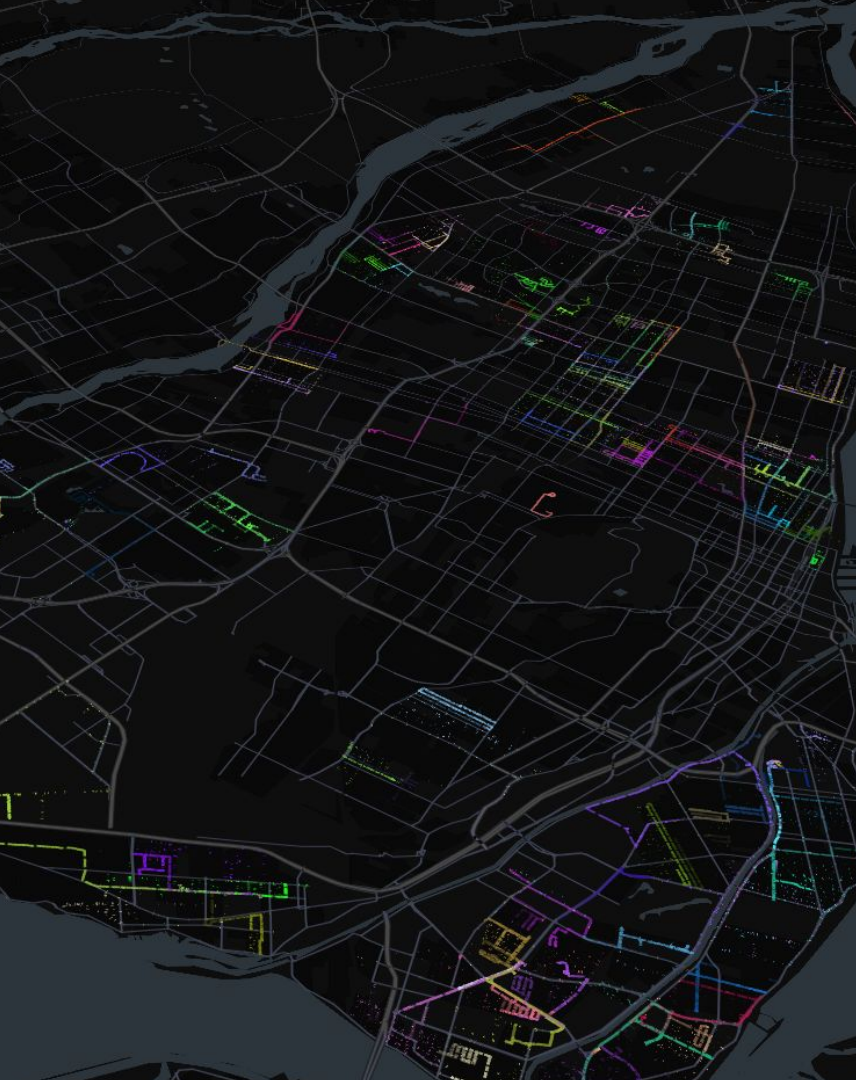
Opérateurs

Opérateurs d'incrément / décrémentation:

- ++ (incrément)
- -- (décrément)

Rétro / Pause





Introduction au format JSON/GeoJSON

JSON (JavaScript Object Notation) est un format de données léger et facile à lire qui est utilisé pour échanger des données entre les applications.

Il est souvent utilisé dans les API Web pour stocker et transférer des données structurées sous forme de paires clé-valeur.

JSON

```
json
{
  "city": "New York",
  "state": "NY",
  "latitude": 40.7128,
  "longitude": -74.0060
}
```

JSON (JavaScript Object Notation) est un format de données léger et facile à lire qui est utilisé pour échanger des données entre les applications.

Il est souvent utilisé dans les API Web pour stocker et transférer des données structurées sous forme de paires clé-valeur.


```
{
  "store": {
    "name": "My Store",
    "location": "New York City",
    "inventory": {
      "products": [
        {
          "id": "p1",
          "name": "Product 1",
          "description": "A great product",
          "price": 10.99,
          "available_sizes": ["S", "M", "L"],
          "reviews": [
            {
              "user": "John Doe",
              "rating": 4,
              "comment": "Great product!"
            },
            {
              "user": "Jane Smith",
              "rating": 3,
              "comment": "It's okay"
            }
          ]
        },
        {
          "id": "p2",
          "name": "Product 2",
          "description": "Another great product",
          "price": 19.99,
          "available_sizes": ["M", "L", "XL"],
          "reviews": [
            {
              "user": "Bob Johnson",
              "rating": 5,
              "comment": "Amazing product!"
            }
          ]
        }
      ]
    }
  }
}
```

JSON

JSON (JavaScript Object Notation) est un format de données léger et facile à lire qui est utilisé pour échanger des données entre les applications.

Il est souvent utilisé dans les API Web pour stocker et transférer des données structurées sous forme de paires clé-valeur.

json

Copy code

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [
      -73.9857,
      40.7484
    ]
  },
  "properties": {
    "name": "Empire State Building",
    "address": "350 Fifth Ave, New York, NY 10118"
  }
}
```

GeoJSON

GeoJSON est une extension de JSON qui permet de stocker et d'échanger des données géographiques, telles que des coordonnées de points, des lignes et des polygones.

Il utilise la même structure clé-valeur que JSON, mais ajoute des types de géométrie pour représenter les données spatiales.

[OGC EO Dataset Metadata](#)
[GeoJSON\(-LD\) Encoding Standard](#)
[- Open Geospatial Consortium](#)

.json

Copy code

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [2.3522, 48.8566]
      },
      "properties": {
        "name": "Paris",
        "country": "France"
      }
    },
    {
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [2.3522, 48.8566],
          [4.8357, 45.7640],
          [1.4442, 43.6045]
        ]
      },
      "properties": {
        "name": "Route des vacances",
        "distance": 1023.6
      }
    }
  ]
}
```

GeoJSON

GeoJSON est une extension de JSON qui permet de stocker et d'échanger des données géographiques, telles que des coordonnées de points, des lignes et des polygones.

Il utilise la même structure clé-valeur que JSON, mais ajoute des types de géométrie pour représenter les données spatiales.

[OGC EO Dataset Metadata](#)
[GeoJSON\(-LD\) Encoding Standard](#)
[- Open Geospatial Consortium](#)

json  Copy code

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [2.3522, 48.8566]
  },
  "properties": {
    "name": "Paris",
    "country": "France",
    "population": 2148000
  },
  "crs": {
    "type": "name",
    "properties": {
      "name": "EPSG:4326"
    }
  },
  "bbox": [2.2269, 48.8156, 2.4699, 48.9021]
}
```

GeoJSON-OGC

- Type
- Coordinates
- Properties
- Crs
- Bbox
- Features

json

Copy code

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [2.3522, 48.8566]
  },
  "properties": {
    "name": "Paris",
    "country": "France",
    "population": 2148000
  },
  "crs": {
    "type": "name",
    "properties": {
      "name": "EPSG:4326"
    }
  },
  "bbox": [2.2269, 48.8156, 2.4699, 48.9021]
}
```

GeoJSON-OGC

Type :

Le type de géométrie, comme

- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon

json

Copy code

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [2.3522, 48.8566]
  },
  "properties": {
    "name": "Paris",
    "country": "France",
    "population": 2148000
  },
  "crs": {
    "type": "name",
    "properties": {
      "name": "EPSG:4326"
    }
  },
  "bbox": [2.2269, 48.8156, 2.4699, 48.9021]
}
```

GeoJSON-OGC

Coordinates :

Les coordonnées géographiques de la géométrie, stockées sous forme d'array.

json

Copy code

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [2.3522, 48.8566]
  },
  "properties": {
    "name": "Paris",
    "country": "France",
    "population": 2148000
  },
  "crs": {
    "type": "name",
    "properties": {
      "name": "EPSG:4326"
    }
  },
  "bbox": [2.2269, 48.8156, 2.4699, 48.9021]
}
```

GeoJSON-OGC

Crs :

Le système de référence de coordonnées spatiales utilisé pour les coordonnées géographiques. Cela peut être utilisé pour convertir les coordonnées en un système de référence différent.

json

Copy code

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [2.3522, 48.8566]
  },
  "properties": {
    "name": "Paris",
    "country": "France",
    "population": 2148000
  },
  "crs": {
    "type": "name",
    "properties": {
      "name": "EPSG:4326"
    }
  },
  "bbox": [2.2269, 48.8156, 2.4699, 48.9021]
}
```

GeoJSON-OGC

Bbox :

La boîte englobante (bounding box) de la géométrie, stockée sous forme d'array de quatre valeurs (xmin, ymin, xmax, ymax). Cela peut être utilisé pour déterminer l'emplacement et l'échelle de la géométrie.

json

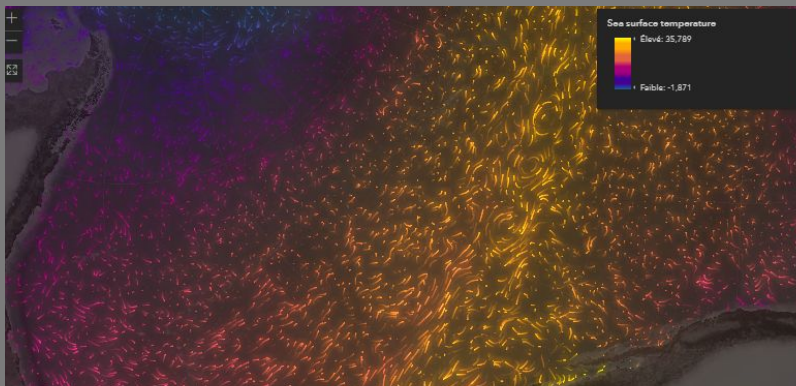
Copy code

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [2.3522, 48.8566]
  },
  "properties": {
    "name": "Paris",
    "country": "France",
    "population": 2148000
  },
  "crs": {
    "type": "name",
    "properties": {
      "name": "EPSG:4326"
    }
  },
  "bbox": [2.2269, 48.8156, 2.4699, 48.9021]
}
```

GeoJSON-OGC

Features :

Pour les objets GeoJSON de type FeatureCollection, il s'agit d'un array d'objets Feature qui représentent chacun une géométrie individuelle avec ses propriétés associées.



How it works

Normally, the FlowRenderer only displays its streamlines in one solid color. To create a multivariate visualization with the FlowRenderer, you can use layer blending. In this application, we're blending together an ocean currents layer (visualized by the flow renderer - which gives us the streamlines) and a sea surface temperature layer (visualized by a [RasterStretchRenderer](#) - which gives us the color). When initializing the currents layer, we give it a [destination-in](#) [blendMode](#) so that the temperature layer only draws where it overlaps with the currents layer.

```
// ocean currents, visualized with flow renderer
const currentsLayer = new ImageryTileLayer({
  url: "https://tiledimageservices.arcgis.com/31l9msH9012080Cb/arcgis/rest/services/Spilhaus_UV_ocean_currents/ImageServer",
  renderer: {
    type: "flow", // autocasts to FlowRenderer
    density: 1,
    maxPathLength: 10, // max length of a streamline will be 10
    trailWidth: "2px"
  },
  blendMode: "destination-in", // temperature layer will only display on top of this layer
});
```

Then, we add both layers to a [GroupLayer](#). We apply the bloom [layer effect](#) to the group layer to make the colors glow on the dark basemap.

```
const groupLayer = new GroupLayer({
  effect: "bloom(2, 0.5px, 0.0)", // apply bloom effect to make the colors pop
  layers: [temperatureLayer, currentsLayer]
});
```

Introduction aux Interfaces de programmation applicatives (APIs)

- Interfaces pour appeler du code écrit par d'autre
- Exposées par des URL / HTTP => JSON
- Exposées par des librairies (web)
- Exposées par des SDK (logiciel)
- Publiques ou privées (authentification)

[Overview | ArcGIS Maps SDK for JavaScript 4.26](#)

Revue des APIs cartographiques



MapLibre GL JS

MapLibre GL JS is an open-source library for publishing maps on your websites or webview based apps. Fast displaying of maps is possible thanks to GPU-accelerated vector tile rendering.

It originated as an open-source fork of `mapbox-gl-js`, before their switch to a non-OSS license in December 2020. The library's initial versions (1.x) were intended to be a drop-in replacement for the Mapbox's OSS version (1.x) with additional functionality, but have evolved a lot since then.

License `BSD 3-Clause` npm `v2.4.0` CI `passing` PRs `welcome`

Getting Started

Include the JavaScript and CSS files in the `<head>` of your HTML file.

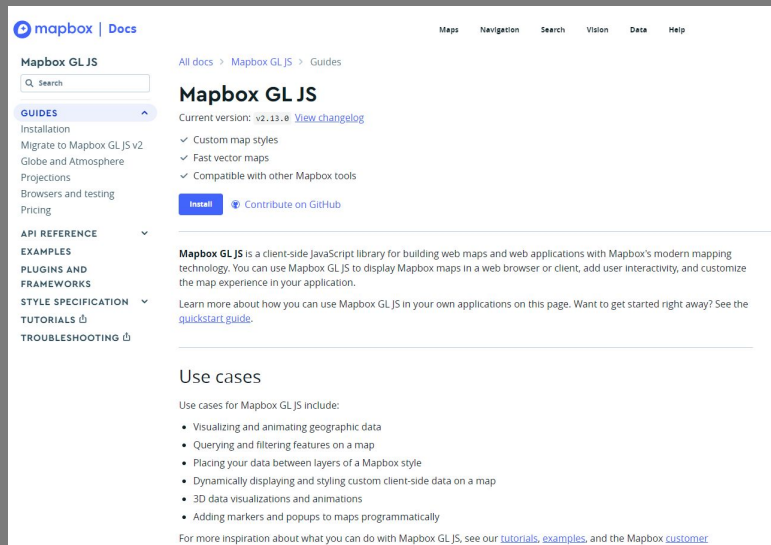
```
<script src='https://unpkg.com/maplibre-gl@latest/dist/maplibre-gl.js'></script>
<link href='https://unpkg.com/maplibre-gl@latest/dist/maplibre-gl.css' rel='stylesheet' />
```

Include the following code in the `<body>` of your HTML file.

```
<div id='map' style='width: 400px; height: 300px;'></div>
<script>
var map = new maplibregl.Map({
  container: 'map',
  style: 'https://demotiles.maplibre.org/style.json', // stylesheet location
  center: [-74.5, 40], // starting position [lng, lat]
  zoom: 9 // starting zoom
});
</script>
```

- [Maplibre](#)
- Mapbox
- Leaflet
- OpenLayer
- DeckGL
- ESRI ArcGIS API for Javascript
- GoogleMaps API
- Turfs

Revue des APIs cartographiques



The screenshot shows the Mapbox GL JS documentation page. The header includes the Mapbox logo and 'Docs'. A navigation bar contains links for Maps, Navigation, Search, Vision, Data, and Help. The left sidebar lists various sections: Mapbox GL JS, GUIDES (with a sub-menu for Installation, Migration, Projections, etc.), API REFERENCE, EXAMPLES, PLUGINS AND FRAMEWORKS, STYLE SPECIFICATION, TUTORIALS, and TROUBLESHOOTING. The main content area is titled 'Mapbox GL JS' and shows the current version (v2.13.0) with a link to the changelog. It lists features like custom map styles, fast vector maps, and compatibility with other Mapbox tools. There are buttons for 'Install' and 'Contribute on GitHub'. Below this, a section titled 'Use cases' lists various applications of the library, such as visualizing geographic data, querying features, and adding markers.

mapbox | Docs

Mapbox GL JS

Search

GUIDES

Installation

Migrate to Mapbox GL JS v2

Globe and Atmosphere

Projections

Browsers and testing

Pricing

API REFERENCE

EXAMPLES

PLUGINS AND FRAMEWORKS

STYLE SPECIFICATION

TUTORIALS

TROUBLESHOOTING

All docs > Mapbox GL JS > Guides

Mapbox GL JS

Current version: v2.13.0 [View changelog](#)

- ✓ Custom map styles
- ✓ Fast vector maps
- ✓ Compatible with other Mapbox tools

[Install](#) [Contribute on GitHub](#)

Mapbox GL JS is a client-side JavaScript library for building web maps and web applications with Mapbox's modern mapping technology. You can use Mapbox GL JS to display Mapbox maps in a web browser or client, add user interactivity, and customize the map experience in your application.

Learn more about how you can use Mapbox GL JS in your own applications on this page. Want to get started right away? See the [quickstart guide](#).

Use cases

Use cases for Mapbox GL JS include:

- Visualizing and animating geographic data
- Querying and filtering features on a map
- Placing your data between layers of a Mapbox style
- Dynamically displaying and styling custom client-side data on a map
- 3D data visualizations and animations
- Adding markers and popups to maps programmatically

For more inspiration about what you can do with Mapbox GL JS, see our [tutorials](#), [examples](#), and the Mapbox [customer](#).

- MapLibre GL
- [Mapbox GL](#)
- Leaflet
- OpenLayer
- DeckGL
- ESRI ArcGIS API for Javascript
- GoogleMaps API
- Turfs

Revue des APIs cartographiques



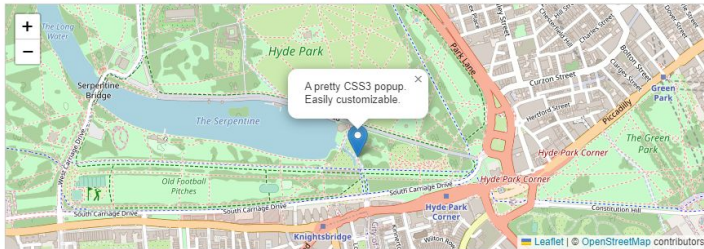
an open-source JavaScript library
for mobile-friendly interactive maps

[Overview](#) [Tutorials](#) [Docs](#) [Download](#) [Plugins](#) [Blog](#)

Sep 21, 2022 — [Leaflet 1.9](#) has been released!

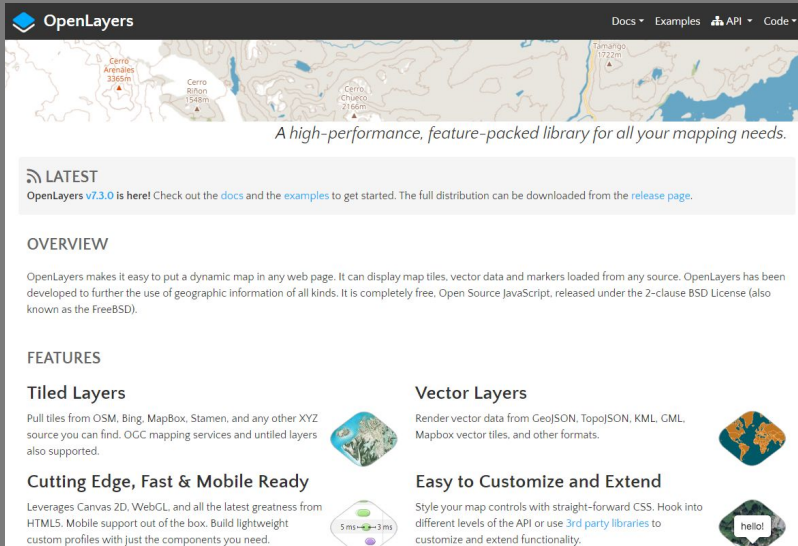
Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 42 KB of JS, it has all the mapping [features](#) most developers ever need.

Leaflet is designed with *simplicity*, *performance* and *usability* in mind. It works efficiently across all major desktop and mobile platforms, can be extended with lots of [plugins](#), has a beautiful, easy to use and [well-documented API](#) and a simple, readable [source code](#) that is a joy to [contribute](#) to.



- MapLibre GL
- Mapbox GL
- [Leaflet](#)
- OpenLayer
- DeckGL
- ESRI ArcGIS API for Javascript
- GoogleMaps API
- Turfs

Revue des APIs cartographiques



The screenshot shows the OpenLayers website. At the top is the OpenLayers logo and navigation links: Docs, Examples, API, and Code. Below the navigation is a map of a mountainous region with labels for Cerro Amaltes (3365m), Cerro El Tor (1548m), Cerro Chupico (2166m), and Tanagero (1720m). Below the map is the tagline: "A high-performance, feature-packed library for all your mapping needs." Below this is a "LATEST" section announcing OpenLayers v7.3.0, with links to docs, examples, and a release page. The "OVERVIEW" section describes OpenLayers as a dynamic map library that can display map tiles, vector data, and markers from any source, and is released under the 2-clause BSD License. The "FEATURES" section is divided into three columns: "Tiled Layers" (pulling tiles from OSM, Bing, MapBox, Stamen, etc.), "Vector Layers" (rendering vector data from GeoJSON, TopoJSON, KML, GML, etc.), and "Cutting Edge, Fast & Mobile Ready" (leveraging Canvas 2D, WebGL, and HTML5). Each feature column includes a small icon: a globe for tiled layers, a map for vector layers, and a speedometer for the cutting-edge features.

OpenLayers

Docs ▾ Examples API ▾ Code ▾

A high-performance, feature-packed library for all your mapping needs.

LATEST

OpenLayers **v7.3.0** is here! Check out the [docs](#) and the [examples](#) to get started. The full distribution can be downloaded from the [release page](#).

OVERVIEW

OpenLayers makes it easy to put a dynamic map in any web page. It can display map tiles, vector data and markers loaded from any source. OpenLayers has been developed to further the use of geographic information of all kinds. It is completely free. Open Source JavaScript, released under the 2-clause BSD License (also known as the FreeBSD).

FEATURES

Tiled Layers

Pull tiles from OSM, Bing, MapBox, Stamen, and any other XYZ source you can find. OGC mapping services and untiled layers also supported.

Vector Layers

Render vector data from GeoJSON, TopoJSON, KML, GML, Mapbox vector tiles, and other formats.

Cutting Edge, Fast & Mobile Ready

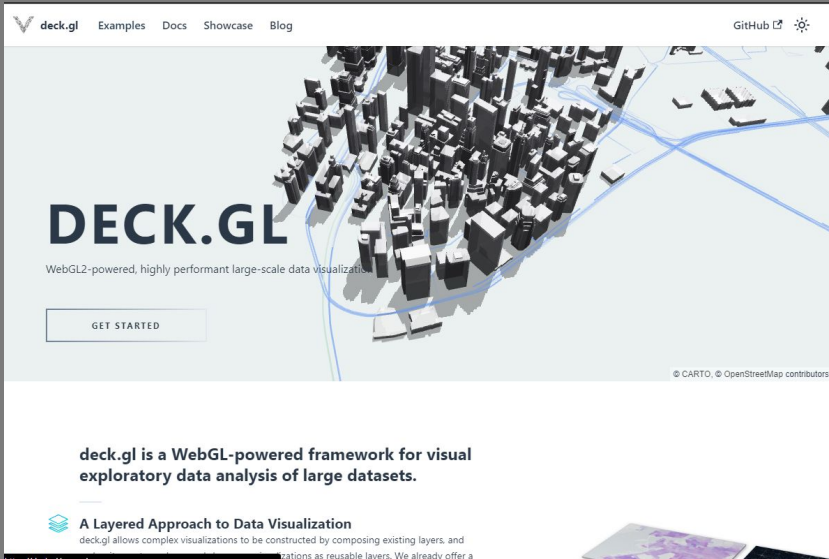
Leverages Canvas 2D, WebGL, and all the latest greatness from HTML5. Mobile support out of the box. Build lightweight custom profiles with just the components you need.

Easy to Customize and Extend

Style your map controls with straight-forward CSS. Hook into different levels of the API or use [3rd party libraries](#) to customize and extend functionality.

- MapLibre GL
- Mapbox GL
- Leaflet
- [OpenLayer](#)
- DeckGL
- ESRI ArcGIS API for Javascript
- GoogleMaps API
- Turfs

Revue des APIs cartographiques



- MapLibre GL
- Mapbox GL
- Leaflet
- OpenLayer
- [DeckGL](#)
- ESRI ArcGIS API for Javascript
- GoogleMaps API
- Turfs

Revue des APIs cartographiques

ArcGIS Developers Documentation Features Pricing Support

ArcGIS Maps SDK for JavaScript Overview Sample Code API

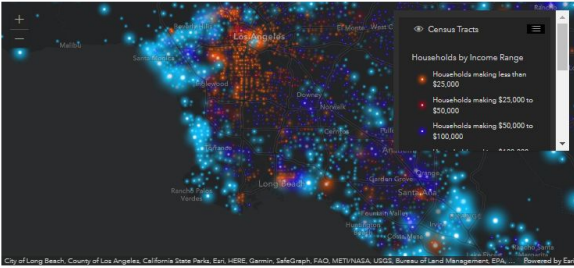
Find page...

- Overview
- Key features
- Get started
- Install and set up
- Release notes
- FAQ
- Community
- Tutorials
- Core concepts
- Visualization
- Building your UI
- Working with ArcGIS Online and Enterprise
- Developer tooling
- Migrating from 3.x
- Reference

Overview

Current version: [4.26](#) (February 2023)

This guide describes how to use ArcGIS Maps SDK for JavaScript to build compelling web apps that unlock your data's potential with interactive user experiences and stunning 2D and 3D visualizations.



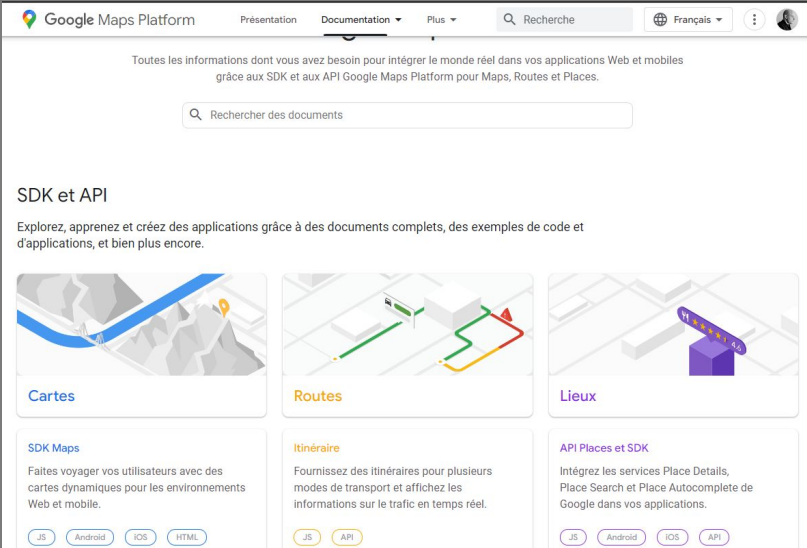
City of Long Beach, County of Los Angeles, California State Parks, Esri, HERE, Garmin, SafeGraph, FAO, METNADA, USGS, Bureau of Land Management, EPA, Powered by Esri

Where to start

- 1 If you are new to ArcGIS start with the [mapping APIs and location services guide](#).
- 2 Follow the [get started](#) instructions to get the API and start developing.
[Start building your app](#)
- 3 Review the key features of the ArcGIS Maps SDK for JavaScript.
[Learn about key features](#)

- MapLibre GL
- Mapbox GL
- Leaflet
- OpenLayer
- DeckGL
- [ESRI ArcGIS API for Javascript](#)
- GoogleMaps API
- Turfs

Revue des APIs cartographiques



The screenshot shows the Google Maps Platform documentation page. At the top, there's a navigation bar with 'Google Maps Platform', 'Présentation', 'Documentation' (selected), 'Plus', a search bar, and a language dropdown set to 'Français'. Below the navigation bar, a message states: 'Toutes les informations dont vous avez besoin pour intégrer le monde réel dans vos applications Web et mobiles grâce aux SDK et aux API Google Maps Platform pour Maps, Routes et Places.' A search bar with the placeholder 'Rechercher des documents' is present. The main content area is titled 'SDK et API' and includes a sub-header: 'Explorez, apprenez et créez des applications grâce à des documents complets, des exemples de code et d'applications, et bien plus encore.' Below this, there are three columns of cards. The first column is titled 'Cartes' and features an image of a map with a blue line and a location pin. The second column is titled 'Routes' and features an image of a 3D map with a green line and a red location pin. The third column is titled 'Lieux' and features an image of a 3D map with a purple location pin. Each card has a description and a set of tags for different platforms: 'JS', 'Android', 'iOS', and 'HTML' for 'Cartes'; 'JS' and 'API' for 'Routes'; and 'JS', 'Android', 'iOS', and 'API' for 'Lieux'.

SDK et API

Explorez, apprenez et créez des applications grâce à des documents complets, des exemples de code et d'applications, et bien plus encore.

Cartes

SDK Maps

Faites voyager vos utilisateurs avec des cartes dynamiques pour les environnements Web et mobile.

JS Android iOS HTML

Routes

Itinéraire

Fournissez des itinéraires pour plusieurs modes de transport et affichez les informations sur le trafic en temps réel.

JS API

Lieux

API Places et SDK

Intégrez les services Place Details, Place Search et Place Autocomplete de Google dans vos applications.

JS Android iOS API

- MapLibre GL
- Mapbox GL
- Leaflet
- OpenLayer
- DeckGL
- ESRI ArcGIS API for Javascript
- [GoogleMaps API](#)
- Turfs

Revue des APIs cartographiques


TURF

GETTING STARTED

MEASUREMENT
along
area
bbox
blobPolygon
bearing
center
centerOfMass
centroid
destination
distance
envelope
length
midpoint
pointOnFeature
polygonTangents
pointToLineDistance
rumbBearing
rumbDestination
rumbDistance
square
greatCircle

COORDINATE MUTATION

Welcome to Turf.js



Advanced geospatial analysis for browsers and Node.js

Simple	Modular	Fast
Modular, simple-to-understand JavaScript functions that speak GeoJSON	Turf is a collection of small modules, you only need to take what you want to use	Takes advantage of the newest algorithms and doesn't require you to send data to a server

- MapLibre GL
- Mapbox GL
- Leaflet
- OpenLayer
- DeckGL
- ESRI ArcGIS API for Javascript
- GoogleMaps API
- [Turf.js](https://turfjs.org/)