**Node Express**

1. **Clients and Servers**
2. **Ip Adress & Domains**

Une adresse IP est un identifiant numérique unique attribué à chaque appareil connecté à Internet. Elle est utilisée pour identifier et communiquer avec d'autres appareils sur un réseau, ainsi que pour acheminer des paquets de données entre les appareils.

Une image contenant texte, logiciel, Logiciel multimédia, diagramme

Description générée automatiquement

* **C’est quoi un http**

Une image contenant texte, logiciel, Logiciel multimédia, Police

Description générée automatiquement

HTTP signifie Hypertext Transfer Protocol. C'est un protocole d'application utilisé pour transférer des données sur Internet entre un client (tel qu'un navigateur Web) et un serveur. HTTP est le fondement du World Wide Web, et il est utilisé pour accéder et transférer des données sur des sites Web et des applications Web.

HTTP fonctionne selon un modèle client-serveur, où le client envoie une demande au serveur, et le serveur répond avec un message contenant les données demandées ou un message d'erreur si la demande ne peut être satisfaite. Les requêtes et les réponses HTTP sont composées d'en-têtes et d'un corps de message.

Les requêtes HTTP sont constituées de plusieurs parties, notamment :

1. Ligne de requête : Cela comprend la méthode HTTP (GET, POST, etc.), l'URL de la ressource demandée et la version HTTP utilisée.
2. En-têtes de requête : Ils fournissent des informations supplémentaires sur la requête, telles que l'agent utilisateur effectuant la demande, le type de données demandées et toute information d'authentification.
3. Corps de la requête : Il contient toutes les données envoyées au serveur, telles que des données de formulaire ou des téléchargements de fichiers.

Les réponses HTTP ont également plusieurs parties, notamment :

1. Ligne d'état : Cela comprend la version HTTP, un code d'état indiquant si la requête a été réussie ou non, et un bref message expliquant l'état.
2. En-têtes de réponse : Ils fournissent des informations supplémentaires sur la réponse, telles que le type de contenu des données renvoyées et des instructions de mise en cache.
3. Corps de la réponse : Il contient les données renvoyées par le serveur, telles qu'une page HTML ou un objet JSON.

HTTP est un protocole sans état, ce qui signifie que chaque requête et réponse est indépendante de toutes les requêtes ou réponses précédentes. Pour maintenir l'état entre les requêtes, les applications Web utilisent souvent des cookies ou d'autres mécanismes pour stocker des informations sur la session du client.

* **Différents types de requêtes http**

Il existe plusieurs types de requêtes HTTP, chacun servant un objectif spécifique lors de la communication entre un client et un serveur. Les méthodes de requête HTTP les plus courantes sont les suivantes :

1. GET : La méthode GET est utilisée pour récupérer des données du serveur. Lorsqu'un client envoie une requête GET, il demande au serveur de renvoyer une représentation de la ressource spécifiée. Les requêtes GET ne doivent récupérer que des données et ne doivent avoir aucun autre effet sur le serveur.
2. POST : La méthode POST est utilisée pour envoyer des données au serveur afin de créer une nouvelle ressource. Elle est couramment utilisée pour soumettre des données à partir de formulaires sur des pages Web et peut également être utilisée à d'autres fins où des données doivent être envoyées au serveur.
3. PUT : La méthode PUT est utilisée pour mettre à jour une ressource existante sur le serveur. Elle envoie des données au serveur, qui sont ensuite utilisées pour mettre à jour la ressource spécifiée.
4. DELETE : La méthode DELETE est utilisée pour demander la suppression d'une ressource spécifiée sur le serveur.
5. PATCH : La méthode PATCH est utilisée pour appliquer des modifications partielles à une ressource sur le serveur. Elle envoie les modifications spécifiées au serveur, qui les applique à la ressource existante.
6. HEAD : La méthode HEAD est similaire à GET, mais elle demande uniquement les en-têtes de la ressource, sans le corps du message. Cela permet au client de vérifier les en-têtes (comme la taille ou le type de contenu) avant de récupérer la ressource complète avec une requête GET.
7. OPTIONS : La méthode OPTIONS est utilisée pour obtenir les options de communication disponibles pour une ressource spécifiée sur le serveur. Cela permet au client de découvrir quelles méthodes de requête et quels en-têtes sont pris en charge par le serveur pour cette ressource.

Ces différentes méthodes de requête HTTP permettent aux clients et aux serveurs de communiquer de manière flexible et d'effectuer diverses opérations sur les ressources du serveur.

1. **Création d’un serveur**

Ce code crée une variable constante appelée "http" et lui assigne le module http. Vous pouvez ensuite utiliser cette variable pour créer un serveur HTTP, effectuer des requêtes HTTP et gérer les réponses HTTP dans votre application Node.js. Par exemple, pour créer un serveur HTTP en utilisant le module http, vous pouvez utiliser le code suivant :

const http = require("http");

const server = http.createServer((req, res) => {

  res.writeHead(200, { "Content-Type": "text/plain" });

  res.end("Hello, world!");

});

server.listen(3000, "localhost", () => {

  console.log("Server running on port 3000");

});

Ce code crée un serveur HTTP qui écoute sur le port 3000 et répond à toutes les requêtes avec un message en texte brut disant "Hello, World!". Lorsqu'un client fait une requête à ce serveur, la fonction createServer crée une nouvelle instance de serveur HTTP, et la fonction de rappel gère la requête entrante et envoie une réponse de retour au client.

1. **L’objet Response(Repose)**

En Node.js, vous pouvez envoyer différents types de contenus au navigateur en utilisant la méthode **response** de l'objet **http.ServerResponse**. Les types de contenus les plus courants sont :

• HTML : Vous pouvez renvoyer une page HTML en utilisant la méthode **response.write()** pour écrire le contenu de la page et **response.end()** pour terminer la réponse.

• JSON : Vous pouvez envoyer des données au format JSON en utilisant la méthode **response.json()** de certains frameworks comme Express.js.

• Fichiers : Vous pouvez envoyer des fichiers tels que des images, des fichiers CSS ou JavaScript en utilisant la méthode **response.sendFile()** ou **response.write()** pour écrire le contenu du fichier et **response.end()** pour terminer la réponse.

Il est également possible d'envoyer d'autres types de contenus tels que des textes bruts, des fichiers PDF ou des fichiers audio ou vidéo en spécifiant correctement l'en-tête **Content-Type** de la réponse.

const http = require("http")

const server = http.createServer((req, res) => {

  console.log(req.url,req.method);

  res.setHeader('Content-Type','text/text')

res.write(‘Hello Ninja’)

  res.end()

});

server.listen(3000, "localhost", () => {

  console.log("listening for request on port 3000");

});

1. **Retourner une page HTML**

On crée note fichier Html (Dans ce cas **index.html**)

const http = require("http");

const fs=require("fs")

const server = http.createServer((req, res) => {

  console.log(req.url,req.method);

  res.setHeader('Content-Type','text/html')

  // send an html file

  fs.readFile("./views/index.html",(err,data)=>{

    if(err){

   console.log(err)

   res.end()

    } else{

      res.write(data)

      res.end()

    }

  })

});

server.listen(3000, "localhost", () => {

  console.log("listening for request on port 3000");

});

1. **Statuts code**

En Node.js, il existe plusieurs codes de statut HTTP que vous pouvez utiliser pour indiquer le résultat d'une requête HTTP. Les codes de statut HTTP sont des nombres à trois chiffres qui indiquent si une requête a été traitée avec succès ou s'il y a eu une erreur.

Voici quelques-uns des codes de statut HTTP les plus couramment utilisés en Node.js :

* 200 OK : la requête a été traitée avec succès.
* 201 Created : la ressource a été créée avec succès.
* 204 No Content : la requête a été traitée avec succès, mais il n'y a pas de contenu à renvoyer.
* 301 Moved Permanently :est utilisé pour indiquer qu'une ressource a été déplacée de façon permanente vers une autre adresse URL. Cela signifie que toute demande future pour la même ressource doit être dirigée vers la nouvelle adresse URL.
* 400 Bad Request : la requête n'a pas pu être comprise ou traitée en raison d'une syntaxe invalide.
* 401 Unauthorized : l'utilisateur n'est pas autorisé à accéder à la ressource demandée.
* 403 Forbidden : l'utilisateur est authentifié mais n'a pas les droits d'accès nécessaires pour accéder à la ressource.
* 404 Not Found : la ressource demandée n'a pas été trouvée sur le serveur.
* 500 Internal Server Error : une erreur interne s'est produite sur le serveur lors du traitement de la requête.

Vous pouvez utiliser ces codes de statut HTTP dans la méthode **response.writeHead()** de l'objet **http.ServerResponse** pour définir le code de statut et les en-têtes de la réponse HTTP.

1. **Basic Routing et la redirection**

En Node.js, la méthode **res.setHeader()** peut être utilisée pour définir l'en-tête **Location** dans une réponse HTTP. L'en-tête **Location** est utilisé pour spécifier une nouvelle adresse URL vers laquelle le client doit être redirigé.

 let path = "./views/";

  switch (req.url) {

    case "/":

      path += "index.html";

      res.statusCode=200

      break;

    case "/about":

      path += "about.html";

      res.statusCode=200

      break;

    case '/about-me':

      res.statusCode=301

      res.setHeader('location','/about')

      res.end()

    default:

      path += "404.html";

      res.statusCode=404

      break;

  }

  fs.readFile(path, (err, data) => {

    if (err) {

      console.log(err);

      res.end();

    } else {

      res.write(data);

      res.end();

    }

  });

});

server.listen(3000, "localhost", () => {

  console.log("listening for request on port 3000");

});